Electronic Colloquium on Computational Complexity, Report No. 9 (2015)

Aggregate Pseudorandom Functions and Connections to Learning

Aloni Cohen* Shafi Goldwasser† Vinod Vaikuntanathan‡

Abstract

In the first part of this work, we introduce a new type of pseudo-random function for which "aggregate queries" over exponential-sized sets can be efficiently answered. An example of an aggregate query may be the product of all function values belonging to an exponential-sized interval, or the sum of all function values on points for which a polynomial time predicate holds. We show how to use algebraic properties of underlying classical pseudo random functions, to construct aggregatable pseudo random functions for a number of classes of aggregation queries under cryptographic hardness assumptions. On the flip side, we show that certain aggregate queries are impossible to support.

In the second part of this work, we show how various extensions of pseudo-random functions considered recently in the cryptographic literature, yield impossibility results for various extensions of machine learning models, continuing a line of investigation originated by Valiant and Kearns in the 1980s and 1990s. The extended pseudo-random functions we address include constrained pseudo random functions, aggregatable pseudo random functions, and pseudo random functions secure under related-key attacks.

*MIT.

†MIT and the Weizmann Institute of Science.

‡MIT.

ISSN 1433-8092

Contents

1 Introduction

Pseudo-random functions (PRF), introduced by Goldreich, Goldwasser and Micali [GGM86], are a family of indexed functions for which there exists a polynomial-time algorithm that, given an index (which can be viewed as a secret key) for a function, can evaluate it, but no probabilistic polynomial-time algorithm without the secret key can distinguish the function from a truly random function – even if allowed oracle query access to the function. Pseudo-random functions have been shown over the years to be useful for numerous cryptographic applications. Interestingly, aside from their cryptographic applications, PRFs have also been used to show impossibility of computational learning in the membership queries model [Val84], and served as the underpinning of the proof of Razborov and Rudich [RR97] that natural proofs would not suffice for unrestricted circuit lower bounds.

Since their inception in the mid eighties, various *augmented* pseudo random functions with *extra* properties have been proposed, enabling more sophisticated forms of access to PRFs and more structured forms of PRFs. In this vein, we mention the work of [GGN10] on constructing "huge random objects" which designs PRFs that are guaranteed to maintain some global combinatorial property; the recent works on constrained PRFs[1] [KPTZ13a, BGI14a, BW13a] which can release auxiliary secret keys whose knowledge enables computing the PRF in a restricted number of locations without compromising pseudo-randomness elsewhere; the construction of a PRF [BLMR13] family which is homomorphic with respect to its key; and the construction of related key secure PRFs [BC10, ABPP14]. These constructions yield fundamental objects with often surprising applications to cryptography and elsewhere. A case in point is the truly surprising use of constrained PRFs [SW14], to show that indistinguishability obfuscation can be used to resolve a long-standing problem of deniable encryption, among many others.

In the first part of this paper, we introduce a new type of augmented PRF which we call *aggregate pseudo random functions* (AGG-PRF). An AGG-PRF is a family of indexed functions each associated with a secret key, such that *given the secret key*, one can compute aggregates of the values of the function *over super-polynomially large sets* in *polynomial time*; and yet without the secret key, access to such aggregated values cannot enable a polynomial time adversary (distinguisher) to distinguish the function from random, even when the adversary can make *aggregate queries*. Note that the distinguisher can request and receive an aggregate of the function values over sets (of possibly super-polynomial size) that she can specify. Examples of aggregate queries can be the sum/product of all function values belonging to an exponential-sized interval, or more generally, the sum/product of all function values on points for which some polynomial time predicate holds. Since the sets over which our function values are aggregated are super-polynomial in size, they cannot be directly computed by simply querying the function on individual points.

We show AGG-PRFs under various cryptographic hardness assumptions (one-way functions and DDH) for a number of types of aggregation operators such as sums and products and for a number of set systems including intervals, hypercubes, and (the supports of) restricted computational models such as decision trees and read-once Boolean formulas. We also show negative results: there are no AGG-PRFs for more expressive set systems such as (the supports of) CNF formulas. For a detailed description of our results, see Section 1.1.

In the second part of this paper, we embark on a study of the connection between the new augmented PRF constructions of recent years (constrained, related-key, aggregate) and the theory

[1] Constrained PRFs are also known as Functional PRFs and as Delegatable PRFs.

of computational learning. We recall at the outset that the fields of cryptography and machine learning share a curious historical relationship. The goals are in complete opposition and at the same time the aesthetics of the models, definitions and techniques bear a striking similarity. For example, a cryptanalyst can attack a cryptosystem using a range of powers from only seeing ciphertext examples to requesting to see decryptions of ciphertexts of her choice. Analogously, machine learning allows different powers to the learner such as random examples versus membership queries and shows that certain powers allow learners to learn concepts in polynomial time whereas others will fail. Even more directly, problems which pose challenges for machine learning such as Learning Parity with Noise (LPN) have been used as the underpinning for building secure cryptosystems, and as mentioned above [Val84] observes that the existence of PRFs in a complexity class $\mathcal{C}$ implies the existence of concept classes in $\mathcal{C}$ which can not be learned under membership queries, and [KV94] extends this direction to some public key constructions.

In the decades since the introduction of PAC learning, new computational learning models have been proposed, such as the recent "restriction access" model [DRWY12] which allows the learner to interact with the target concept by asking membership queries, but also to obtain an entire circuit that computes the concept on a random subset of the inputs. For example, in one shot, the learner can obtain a circuit that computes the concept class on all n-bit inputs that start with $n/2$ zeros. At the same time, the cryptographic research landscape has been swiftly moving in the direction of augmenting traditional PRFs and other cryptographic primitives to include higher functionalities. This brings to mind natural questions:

- *Can one leverage augmented pseudo-random function constructions to establish limits on what can and cannot be learned in augmented machine learning models*?
- *Going even further afield, can augmented cryptographic constructs suggest interesting learning models?*

We address these questions in the second part of this paper. For a detailed description of our findings, see Section 1.2.

1.1 Our Results: Aggregate Pseudo Random Functions

Aggregate Pseudo Random Functions (AGG-PRF) are indexed families of pseudo-random functions for which a distinguisher (who runs in time polynomial in the security parameter) can request and receive the value of an aggregate (for example, the sum or the product) of the function values over certain large sets and yet cannot distinguish oracle access to the function from oracle access to a truly random function. At the same time, given the function index (in other words, the secret key), one can compute such aggregates over potentially super-polynomial size sets in polynomial time. Such an efficent aggregation algorithm cannot possibly exist for random functions. Thus, this is a PRF family that is very unlike random functions (in the sense of being able to efficiently aggregate over superpolynomial size sets), and yet is computationally indistinguishable from random functions.

To make this notion precise, we need two ingredients. Let $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda>0}$ where each $\mathcal{F}_\lambda = \{f_K : \mathcal{D}_\lambda \to \mathcal{R}_\lambda\}_{K\in\mathcal{K}_\lambda}$ is a collection of functions on a domain $\mathcal{D}_\lambda$ to a range $\mathcal{R}_\lambda$, computable in time $\mathsf{poly}(\lambda)$.[2] The first ingredient is a collection of sets (also called a set system) $\mathcal{S} = \{S \subseteq \mathcal{D}\}$ over

[2] In this informal exposition, for the sake of brevity, we will sometimes omit the security parameter and refrain from referring to ensembles.

which the aggregates can be efficiently computed given the index K of the function. The second ingredient is an aggregation function $\Gamma : \mathcal{R}^* \to \{0,1\}^*$ which takes as input a tuple of function values $\{f(x) : x \in S\}$ for some set $S \in \mathcal{S}$ and outputs the aggregate $\Gamma(f(x_1), \ldots, f(x_{|S|}))$.

The sets are typically super-polynomially large, but are efficiently recognizable. That is, for each set S, there is a corresponding $\mathsf{poly}(\lambda)$-size circuit C_S that takes as input an $x \in \mathcal{D}$ and outputs 1 if and only if $x \in S$. [3] Throughout this paper, we will consider relatively simple aggregate functions, namely we will treat the range of the functions as an Abelian group, and will let Γ denote the group operation on its inputs. Note that the input to Γ is super-polynomially large (in the security parameter λ), making the aggregate computation non-trivial.

This family of functions, equipped with a set system $\mathcal{S}$ and an aggregation function Γ is called an aggregate PRF family (AGG-PRF) if the following two requirements hold:

1. *Aggregatability:* There exists a polynomial (in the security parameter λ) time algorithm that given an index K to the PRF $f_K \in \mathcal{F}$ and a circuit C_S that recognizes a set $S \in \mathcal{S}$, can compute Γ over the PRF values $f_K(x)$ for all $x \in S$. That is, it can compute

$$AGG_{K,\Gamma}(S) := \Gamma_{x \in S}\ f_K(x)$$

2. *Pseudorandomness:* No polynomial-time distinguisher which can specify a set $S \in \mathcal{S}$ as a query and can receive as an answer either $AGG_{K,\Gamma}(S)$ for a random function $f_K \in \mathcal{F}$ or $AGG_{h,\Gamma}(S)$ for a truly random functions h, can distinguish between the two cases.

We remark that our notion of aggregate PRFs bears some resemblance to the notion of "algebraic PRFs" defined in the work of Benabbas, Gennaro and Vahlis [BGV11]. In a nutshell, there are two main differences. First, algebraic PRFs support efficient aggregation over very specific subsets, whereas our constructions of aggregate PRFs support expressive subset classes, such as subsets recognized by hypercubes, decision trees and read-once Boolean formulas. Secondly, in the security notion for aggregate PRFs, the adversary obtains access to an oracle that computes the function as well as one that computes the aggregate values over super-polynomial size sets, whereas in algebraic PRFs, the adversary is restricted to accessing the function oracle alone. Our constructions from DDH use an algebraic property of the Naor-Reingold PRF in a similar manner as in [BGV11].

We show a number of constructions of AGG-PRF for various set systems under different cryptographic assumptions. We describe our constructions below, starting from the least expressive set system.

Interval Sets. We first construct AGG-PRFs over interval set systems with respect to aggregation functions that compute any group operation. The construction can be based on any (standard) PRF family.

Theorem 1.1 (Intervals from one-way functions)**.** *Assume one-way functions exist. Then, there exists an AGG-PRF family that maps $\mathbb{Z}_p$ to a group G, with respect to a collection of sets defined by intervals $[a,b] \subseteq \mathbb{Z}_p$ and the aggregation function computing the group operation on G.*

[3] All the sets we consider are efficiently recognizable, and we use the corresponding circuit as the representation of the set. We occasionally abuse notation and use S and C_S interchangeably.

The construction works as follows. Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be a (standard) pseudorandom function family based on the existence of one-way functions [GGM86, HILL99]. Construct an AGG-PRF family G supporting efficient computation of group aggregation functions. Define

$$G(k, x) = F(k, x) - F(k, x-1)$$

To aggregate G, set

$$\sum_{x \in [a,b]} G(k, x) = F(k, b) - F(k, a-1)$$

Given k, this can be efficiently evaluated. This construction can also be easily constrained to allow evaluation on any point or sub-interval of any interval $[a, b]$.

Hypercubes. As a warmup, we next construct AGG-PRFs over hypercube set systems. Throughout this section, we take $\mathcal{D}_\lambda = \{0,1\}^\ell$ for some polynomial $\ell = \ell(\lambda)$. A hypercube $S_\mathbf{y}$ is defined by a vector $\mathbf{y} \in \{0, 1, \star\}^\ell$ as

$$S_\mathbf{y} = \{\mathbf{x} \in \{0,1\}^\ell : \ \forall i, y_i = \star \text{ or } x_i = y_i\}$$

We present a construction under the sub-exponential DDH assumption.

Theorem 1.2 (Hypercubes from DDH)**.** *Let $\mathcal{HC} = \{\mathcal{HC}_{\ell(\lambda)}\}_{\lambda>0}$ where $\mathcal{HC}_\ell = \{0,1,\star\}^\ell$ be the set of hypercubes on $\{0,1\}^\ell$. Then, there is a construction of AGG-PRF supporting the set system $\mathcal{HC}$ with the product aggregation function, assuming the subexponential DDH assumption.*

We sketch the construction from DDH below. Our DDH construction is the Naor-Reingold PRF [NR04]. Namely, the function is parametrized by an ℓ-tuple $\vec{k} = (k_1, \dots, k_\ell)$ and is defined as

$$F(\vec{k}, x) = g^{\prod_{i:x_i=1} k_i}$$

Let us illustrate aggregation over the hypercube $\mathbf{y} = (1, 0, \star, \star, \dots, \star)$. To aggregate the function F, observe that

$$\begin{aligned}\prod_{\{x:\ x_1=1, x_2=0\}} F(\vec{k}, x) &= \prod_{\{x:\ x_1=1, x_2=0\}} g^{\prod_{i:x_i=1} k_i} \\ &= g^{\sum_{\{x:x_1=1,x_2=0\}} \prod_{i:x_i=1} k_i} \\ &= g^{(k_1)(1)(k_2+1)(k_3+1)\cdots(k_\ell+1)}\end{aligned}$$

which can be efficiently computed given $\vec{k}$.

Decision Trees. A decision tree T on ℓ variables is a binary tree where each internal node is labeled by a variable x_i, the leaves are labeled by either 0 or 1, one of the two outgoing edges of an internal node is labeled 0, and the other is labeled 1. Computation of a decision tree on an input $(x_1, \dots, x_\ell)$ starts from the root, and at each internal node n, proceeds by taking either the 0-outgoing edge or 1-outgoing edge depending on whether $x_n = 0$ or $x_n = 1$, respectively. Finally, the output of the computation is the label of the leaf reached through this process. The size of a decision tree is the number of nodes in the tree.

A decision tree T defines a set $S = S_T = \{x \in \{0,1\}^\ell : T(x) = 1\}$. We show how to compute product aggregates over sets defined by polynomial size decision trees, under the subexponential DDH assumption.

The construction is simply a result of the observation that the set $S = S_T$ can be written as a disjoint union of polynomially many hypercubes. Computing aggregates over each hypercube and multiplying the results together gives us the decision tree aggregate.

Theorem 1.3 (Decision Trees from DDH). *Assuming the sub-exponential hardness of the decisional Diffie-Hellman assumption, there is an AGG-PRF that supports aggregation over sets recognized by polynomial-size decision trees.*

Read-Once Boolean Formulas. Finally, we show a construction of AGG-PRF over read-once Boolean formulas, the most expressive of our set systems, under the subexponential DDH assumption. A read-once Boolean formula a Boolean circuit composed of AND, OR and NOT gates with fan-out 1, namely each input literal feeds into at most one gate, and each gate output feeds into at most one other gate. Thus, a read-once formula can be written as a binary tree where each internal node is labeled with an AND or OR gate, and each literal (variable or its negation) appears in at most one leaf.

Theorem 1.4 (Read-Once Boolean Formulas from DDH). *Under the subexponential decisional Diffie-Hellman assumption, there is an AGG-PRF that supports aggregation over sets recognized by read-once Boolean formulas.*

Our aggregate PRF is, once again, the Naor-Reingold PRF. The index of the PRF consists of a $(\ell+1)$-tuple of integers in $\mathbb{Z}_p$, namely $\vec{K} = (K_0, \ldots, K_\ell) \in \mathbb{Z}_p^{\ell+1}$. The function is defined as

$$f_{\vec{K}}(x) = g^{K_0 \prod_{i \in [\ell]} K_i^{x_i}}$$

We compute aggregates by recursion on the levels of the formula. We start by noting that it is enough to compute

$$A(C,1) := \sum_{x:C(x)=1} \prod_{i \in [1\ldots\ell]} K_i^{x_i}$$

because once this is done, it is easy to compute

$$\prod_{x:C(x)=1} f_{\vec{k}}(x) = g^{K_0 \cdot A(C,1)}$$

For the purposes of this informal exposition, assume that ℓ is a power of two. Let C be the formula, with either $C = C_L \wedge C_R$ or $C = C_L \vee C_R$ for subformula C_L and C_R. We show how to recursively compute $A(C,1)$ for these sub-circuits and thus for C.

Limits of Aggregation. A natural question to ask is whether one can support aggregation over sets defined by general circuits. It is however easy to see that you cannot support any class of circuits for which deciding satisfiability is hard (for example, AC^0), or even ones for which counting the number of SAT assignments is hard (DNFs, for example) as follows. Suppose C is a circuit which is either unsatisfiable or has a unique SAT assignment. Solving satisfiability for such circuits is known to be sufficient to solve SAT in general [VV86]. The algorithm for SAT simply runs the aggregator with a random PRF key K, and outputs YES if and only if the aggregator returns a non-zero value. Note that if the formula is unsatisfiable, we will always get 0 from the aggregator. Otherwise, we get $f_k(x)$, where x is the (unique) satisfying assignment. Now, this might end up being 0 accidentally, but cannot be 0 always since otherwise, we will turn it into a PRF distinguisher. The distinguisher has the satisfying assignment hardcoded into it non-uniformly, and it simply checks if $PRF_K(x)$ is 0.

Theorem 1.5 (Impossibility for General Set Systems)**.** *Suppose there is an efficient algorithm which on an index for $f \in \mathcal{F}$, a set system defined by $\{x : C(x) = 1\}$ for a polynomial size Boolean circuit C, and an aggregation function Γ, outputs the $\Gamma_{x:C(x)=1} f(x)$. Then, there is efficient algorithm that takes circuits C as input and w.h.p. over its coins, decides satisfiability for C.*

1.2 Our Results: Augmented PRFs and Computational Learning

As discussed above, connections between PRFs and learning theory date back to the 80's in the pioneering work of [Val84] showing that PRF in a complexity class C implies the existence of concept classes in C which can not be learned with membership queries. In the second part of this work, we study the implications of the slew of augmented PRF constructions of recent years [BW13a, BGI14a, KPTZ13b, BC10, ABPP14] and our new aggregate PRF to computational learning.

1.2.1 Constrained PRFs and limits on Restriction Access learnability

Recently, Dvir, Rao, Wigderson, and Yehudayoff [DRWY12] introduced a new learning model where the learner is allowed non-black-box information on the computational device (such as circuits, DNF,formulas) that decides the concept; their learner receives a simplified device resulting from partial assignments to input variables (i.e. restrictions). These partial restrictions lie somewhere in between function evaluation (full restrictions) which correspond to learning with membership queries and the full description of the original device (the empty restriction). The work of [DRWY12] studies a PAC version of restriction access, called PAC_{RA}, where the learner receives the circuit restricted with respect to random partial assignments. They show that both decision trees and DNF formulas can be learned efficiently in this model. Indeed, the PAC_{RA} model seems like quite a powerful generalization, if not too unrealistic, of the traditional PAC learning model, as it returns to the learner a computational description of the simplified concept.

Yet, in this section we will show limitations of this computational model under cryptographic assumptions. We show that the *constrained pseudo-random function families* introduced recently in [BW13b, BGI14b, KPTZ13a] naturally define a concept class which is not learnable by an even stronger variant of the restriction access learning model which we define. In the stronger variant, which we name *membership queries with restriction access (MQ_{RA})* the learner can adaptively specify any restriction of the circuit from a specified class of restrictions $\mathcal{S}$ and receive the simplified device computing the concept on this restricted domain in return. As this setting requires substantial notation, we define this new model very informally, and defer the formal definitions and theorems to the full version.

Definition 1.1 (Membership queries with restriction access (MQ_{RA}))**.** *Let $\mathcal{C} : X \to \{0,1\}$ be a concept class, and $\mathcal{S} = \{S \subseteq X\}$ be a collection of subsets of the domain. $\mathcal{S}$ is the set of allowable* *restrictions for concepts $f \in \mathcal{C}$. Let* Simp *be "simplification rule" which, for a concept f and restriction S outputs a "simplification" of f restricted to $\mathcal{S}$.*

An algorithm $\mathcal{A}$ is an $(\epsilon, \delta, \alpha)$-$\mathrm{MQ}_{\mathrm{RA}}$learning algorithm *for representation class $\mathcal{C}$ with respect to a restrictions in $\mathcal{S}$ and simplification rule* Simp *if, for every $f \in \mathcal{C}$,* $\Pr[\mathcal{A}^{\mathsf{Simp}(f,\cdot)} = h] \geq 1 - \delta$ *where h is an ϵ-approximation to f – and furthermore, $\mathcal{A}$ only requests restrictions for an α-fraction of the whole domain X.*

Informally, constrained PRFs are PRFs with two additional properties: 1) for any subset S of the domain in a specified collection $\mathcal{S}$, a *constrained key* K_S can be computed, knowledge of which enables efficient evaluation of the PRF on S; and 2) even with knowledge of constrained keys $K_{S_1}, \ldots, K_{S_m}$ for the corresponding subsets, the function retains pseudo-randomness on all points not covered by any of these sets. Connecting this to restriction access, the constrained keys will allow for generation of restriction access examples (restricted implementations with fixed partial assignments) and the second property implies that those examples do not aid in the learning of the function.

Theorem 1.6 (Informal)**.** *Suppose $\mathcal{F}$ is a family of constrained PRFs which can be constrained to sets in $\mathcal{S}$. If $\mathcal{F}$ is computable in circuit complexity class $\mathcal{C}$, then $\mathcal{C}$ is hard to MQ_{RA}-learn with restrictions in $\mathcal{S}$.*

Corollary 1.7 (Informal)**.** *Existing constructions of constrained PRFs [BW13a] yield the following corollaries:*

- *If one-way functions exist, then poly-sized circuits can not be learned with restrictions on sub-intervals of the input-domain; and*
- *Assuming the sub-exponential hardness of the multi-linear Diffie-Hellman problem, NC^1 cannot be learned with restriction on hypercubes.*

1.2.2 New Learning Models Inspired by the Study of PRFs

We proceed to define two new learning models inspired by recent directions in cryptography. The first model is the *related concept* model inspired by work into related-key attacks in cryptography. While we have cryptography and lower bounds in mind, we argue that this model is in some ways natural. The second model, learning with *aggregate queries*, is directly inspired by our development of aggregate pseudo-random functions in this work; rather than being a natural model in its own right, this model further illustrates how cryptography and learning are duals in many senses.

The Related Concept Learning Model The idea that some functions or concepts are related to one another is quite natural. For a DNF formula, for instance, related concepts may include formulas where a clause has been added or formulas where the roles of two variables are swapped. For a decision tree, we could consider removing some accepting leaves and examining the resulting behavior. For a circuit, a related circuit might alter internal gates or fix the values on some wires. A similar phenomena occurs in cryptography, where secret keys corresponding to different instances of the same cryptographic primitive or even secret keys of different cryptographic primitives are related (if, for example, they were generated by a pseudo random process on the same seed).

We propose a new computational learning model where the learner is explicitly allowed to specify membership queries not only for the concept to be learned, but also for "related" concepts, given by a class of allowed transformations on the concept. We will show both a separation from membership queries, and a general negative result in the new model. Based on recent constructions of related-key secure PRFs by Bellare and Cash [BC10] and Abdalla et al [ABPP14], we demonstrate concept classes for which access to these related concepts is of no help.

To formalize the related concept learning model, we will consider *keyed concept classes* – classes indexed by a set of keys. This will enable the study of related concepts by instead considering concepts whose keys are related in some way. Most generally, we think of a key as a succinct representation of the computational device which decides the concept. This is a general framework; for example, we may consider the bit representation of a particular log-depth circuit as a key for a concept in the concept class NC^1. For a concept f_k in concept class $\mathcal{C}$, we allow the learner to query a membership oracle for f_k and also for 'related' concepts $f_{\phi(k)} \in \mathcal{C}_K$ for ϕ in a specified class of allowable functions Φ. For example: let $K = \{0,1\}^\lambda$ and let $\Phi^{\oplus} = \{\phi_\Delta : k \mapsto k \oplus \Delta\}_{\Delta \in \{0,1\}^\lambda}$. Informally:

Definition 1.2 (Φ-Related-Concept Learning Model (Φ-RC))**.** *For $\mathcal{C}_K$ a keyed concept class, let $\Phi = \{\phi : K \to K\}$ be a set of functions on K that contains the identity function* id*. A related-concept oracle RC_k, on query (ϕ, x), responds with $f_{\phi(k)}(x)$, for all $\phi \in \Phi$ and $x \in X$.*

An algorithm A is an (ϵ, δ)-Φ-RK learning algorithm for a $\mathcal{C}_k$ if, for every $k \in K$, when given access to the oracle $RK_k(\cdot)$, the algorithm A outputs with probability at least $1 - \delta$ a function $h : \{0,1\}^n \to \{0,1\}$ that ϵ-approximates f_k.

Yet again, we are able to demonstrate the limitations of this model using the power of a strong type of pseudo-random function. We show that *related-key secure PRF families (RKA-PRF)* defined and instantiated in [BC10] and [ABPP14] give a natural concept class which is not learnable with related key queries. RKA-PRFs are defined with respect to a set Φ of functions on the set of PRF keys. Informally, the security notion guarantees that for a randomly selected key k, no efficient adversary can distinguish oracle access to f_k and $f_{\phi(k)}$ (for many adaptively chosen functions $\phi \in \Phi$) from an oracle that returns completely random values. We leverage this strong pseudo-randomness property to show hard-to-learn concepts in the related concept model.

Theorem 1.8 (Informal)**.** *Suppose $\mathcal{F}$ is a family of RKA-PRFs with respect to related-key functions Φ. If $\mathcal{F}$ is computable in circuit complexity class $\mathcal{C}$, then $\mathcal{C}$ is hard to learn in the Φ'-RC model for some Φ'.*

Existing constructions of RKA-PRFs [ABPP14] yield the following corollary:

Corollary 1.9 (Informal)**.** *Assuming the hardness of the DDH problem, and collision-resistant hash functions, NC^1 is hard to Φ-RC-learn for an class of affine functions Φ.*

The Aggregate Learning Model The other learning model we propose is inspired by our aggregate PRFs. Here, we consider a new extension to the power of the learning algorithm. Whereas membership queries are of the form "What is the label of an example x?", we grant the learner the power to request the evaluation of simple functions on tuples of examples $(x_1, ..., x_n)$ such as "How many of $x_1, \ldots, x_n$ are in C?" or "Compute the product of the labels of $x_1, ..., x_n$?". Clearly, if n is polynomial then this will result only a polynomial gain in the query complexity of a learning algorithm in the best case. Instead, we propose to study cases when n may be super-polynomial, but the description of the tuples is succinct. For example, the learning algorithm might query the number of x's in a large interval that are positive examples in the concept.

As with the restriction access and related concept models – and the aggregate PRFs we define in this work – the Aggregate Queries (AQ) learning model will be considered with restrictions to both the types of aggregate functions Γ the learner can query, and the sets $\mathcal{S}$ over which the learner may request these functions to be evaluated on. We now present the AQ learning model informally:

Definition 1.3 (($\Gamma, \mathcal{S}$)-Aggregate Queries (AQ) Learning)**.** *Let $\mathcal{C} : X \to \{0,1\}$ be a concept class, and let $\mathcal{S}$ be a collection of subsets of X. Let $\Gamma : \{0,1\}^* \to V$ be an aggregation function. For $f \in \mathcal{C}$, let AGG_f be an "aggregation" oracle, which for $S \in \mathcal{S}$, returns $\Gamma_{x \in S} f(x)$. Let MEM_f be the membership oracle, which for input x returns $f(x)$.*

An algorithm $\mathcal{A}$ is an (ϵ, δ)-$(\Gamma, \mathcal{S})$-AQ learning algorithm for $\mathcal{C}$ if for every $f \in \mathcal{C}$,

$$\Pr[\mathcal{A}^{MEM_f(\cdot), \mathsf{AGG}_f(\cdot)} = h] \geq 1 - \delta$$

where h is an ϵ-approximation to f.

Initially, AQ learning is reminiscent of learning with statistical queries (SQ). In fact, this apparent connection inspired this portion of our work. But the AQ setting is in fact incomparable to SQ learning, or even the weaker "statistical queries that are independent of the target" as defined in [BF02]. On the one hand, AQ queries provide a sort of noiseless variant of SQ, giving more power to the AQ learner; on the other hand, the AQ learner is restricted to aggregating over sets in $\mathcal{S}$, whereas the SQ learner is not restricted in this way, thereby limiting the power of the AQ learner. The AQ setting where $\mathcal{S}$ contains every subset of the domain is indeed a noiseless version of "statistical queries independent of the target," but even this model is a restricted version of SQ. This does raise the natural question of a noiseless version of SQ and its variants; hardness results in such models would be interesting in that they would suggest that the hardness comes not from the noise but from an inherent loss of information in statistics/aggregates.

We will show both a simple separation from learning with membership queries (in the full version), and under cryptographic assumptions, a general lower bound on the power of learning with aggregate queries. The negative examples will use the results in Section 1.1.

Theorem 1.10. *Let $\mathcal{F}$ be a boolean-valued aggregate PRF with respect to set system $\mathcal{S}$ and aggregation function Γ. If $\mathcal{F}$ is computable in complexity class $\mathcal{C}$, then $\mathcal{C}$ is hard to $(\Gamma, \mathcal{S})$-AQ learn.*

Corollary 1.11. *Using the results from Section 3, we get the following corollaries:*

- *The existence of one way functions implies that P/poly is hard to $(\sum, \mathcal{S}_{[a,b]})$-AQ learn, with $\mathcal{S}_{[a,b]}$ the set of sub-intervals of the domain as defined in Section 3.*
- *The DDH assumption implies that NC^1 is hard to $(\sum, \mathcal{S}_{[a,b]})$-AQ learn, with $\mathcal{S}_{[a,b]}$ being the set of sub-intervals of the domain as defined in Section 3.*
- *The subexponential DDH Assumption implies that NC^1 is hard to $(\prod, \mathcal{R})$-AQ learn, with $\mathcal{R}$ the set of read-once boolean formulas defined in Section 3.*

Open Questions. As discussed in the introduction, augmented pseudo-random functions often have powerful and surprising applications, perhaps the most recent example being constrained PRFs [BW13a, KPTZ13a, BGI14a]. Perhaps the most obvious open question that emerges from this work is to find applications for aggregate PRFs. We remark that a primitive similar to aggregate PRFs was used in [BGV11] to construct delegation protocols.

Perhaps a more immediate concern is that all our aggregate PRF constructions (except for intervals) requires sub-exponential hardness assumptions. We view it as an important open question to base these constructions on polynomial assumptions.

In this work we restricted our attention to particular types of aggregation functions and subsets over which the aggregation takes place, although our definition captures more general scenarios. We looked at aggregation functions that compute group operations over Abelian groups. Can we support more general aggregation functions that are not restricted to group operations, for example the majority aggregation function, or even non-symmetric aggregation functions? We show positive results for intervals, hypercubes, and sets recognized by read-once formulas and decision trees. On the other hand, we show that it is unlikely that we can support general sets, for example sets recognized by CNF formulas. This almost closes the gap between what is possible and what is hard. A concrete open question in this direction is to construct an aggregate PRF computing summation over an Abelian group for sets recognized by DNFs, or provide evidence that this cannot be done.

Organization. This paper is organized into two parts that can be read essentially independently of each other. In the first part (Sections 2 and 3), we present the definition and constructions of aggregate pseudo-random functions. In the second part (Section 4), we show connections between various notions of augmented PRFs and their applications to augmented learning models.

2 Aggregate PRF

We will let λ denote the security parameter throughout this paper.

Let $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda>0}$ be a function family where each function $f \in \mathcal{F}_\lambda$ maps a domain $\mathcal{D}_\lambda$ to a range $\mathcal{R}_\lambda$. An *aggregate function* family is associated with two objects:

1. an ensemble of sets $\mathcal{S} = \{\mathcal{S}_\lambda\}_{\lambda>0}$ where each $\mathcal{S}_\lambda$ is a collection of subsets of the domain $S \subseteq \mathcal{D}_\lambda$; and

2. an "aggregation function" $\Gamma_\lambda : (\mathcal{R}_\lambda)^* \to \mathcal{V}_\lambda$ that takes a tuple of values from the range $\mathcal{R}_\lambda$ of the function family and "aggregates" them to produce a value in an output set $\mathcal{V}_\lambda$.

Let us now make this notion formal. To do so, we will impose restrictions on the set ensembles and the aggregation function. First, we require set ensemble $\mathcal{S}_\lambda$ to be *efficiently recognizable*. That is, there is a polynomial-size Boolean circuit family $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda>0}$ such that for any set $S \in \mathcal{S}_\lambda$ there is a circuit $C = C_S \in \mathcal{C}_\lambda$ such that $x \in S$ if and only if $C(x) = 1$. Second, we require our aggregation functions Γ to be efficient in the length of its inputs, and symmetric; namely the output of the function does not depend on the order in which the inputs are fed into it. Summation over an Abelian group is an example of a possible aggregation function. Third and finally, elements in our sets $\mathcal{D}_\lambda$, $\mathcal{R}_\lambda$, and $\mathcal{V}_\lambda$ are all representable in $\mathsf{poly}(\lambda)$ bits, and the functions $f \in \mathcal{F}_\lambda$ are computable in $\mathsf{poly}(\lambda)$ time.

Define the aggregate function $\mathsf{AGG} = \mathsf{AGG}^{\lambda}_{f,\mathcal{S}_\lambda,\Gamma_\lambda}$ that is indexed by a function $f \in \mathcal{F}_\lambda$, takes as input a set $S \in \mathcal{S}_\lambda$ and "aggregates" the values of $f(x)$ for all $x \in \mathcal{S}_\lambda$. That is, $\mathsf{AGG}(S)$ outputs

$$\Gamma\big(f(x_1), f(x_2), \ldots, f(x_{|S|})\big)$$

where $S = \{x_1, \ldots, x_{|S|}\}$. More precisely, we have

$$\begin{aligned} \mathsf{AGG}^{\lambda}_{f,\mathcal{S}_\lambda,\Gamma_\lambda} : & \mathcal{S}_\lambda \to \mathcal{V}_\lambda \\ & S \mapsto \Gamma_{x_i \in S}\big(f(x_1), \ldots, f(x_{|S|})\big) \end{aligned}$$

We will furthermore require that the AGG can be computed in $\mathsf{poly}(\lambda)$ time. We require this in spite of the fact that the sets over which the aggregation is done can be exponentially large! Clearly, such a thing is impossible for a random function f but yet, we will show how to construct *pseudo-random* function families that support efficient aggregate evaluation. We will call such a pseudo-random function (PRF) family an *aggregate PRF* family. In other words, our objective is two fold:

1. Allow anyone who knows the (polynomial size) function description to efficiently compute the aggregate function values over exponentially large sets; but at the same time,
2. Ensure that the function family is indistinguishable from a truly random function, even given an oracle that computes aggregate values.

A simple example of aggregates is that of computing the summation of function values over sub-intervals of the domain. That is, let domain and range be $\mathbb{Z}_p$ for some $p = p(\lambda)$, let the family of subsets be $\mathcal{S}_\lambda = \{[a,b] \subseteq \mathbb{Z}_p : a, b \in \mathbb{Z}_p; a \leq b\}$, and the aggregation function be $\Gamma_\lambda(y_1, \ldots, y_k) = \sum_{i=1}^{k} y_i \pmod p$. In this case, we are interested in computing

$$\mathsf{AGG}^{\lambda}_{f,\mathcal{S}_\lambda,\mathsf{sum}}([a,b]) = \sum_{a \leq x \leq b} f(x)$$

We will, in due course, show both constructions and impossibility results for aggregate PRFs, but first let us start with the formal definition.

Definition 2.1 (Aggregate PRF)**.** *Let $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda>0}$ be a function family where each function $f \in \mathcal{F}_\lambda$ maps a domain $\mathcal{D}_\lambda$ to a range $\mathcal{R}_\lambda$, $\mathcal{S}$ be an efficiently recognizable ensemble of sets $\{\mathcal{S}_\lambda\}_{\lambda>0}$, and $\Gamma_\lambda : (\mathcal{R}_\lambda)^* \to \mathcal{V}_\lambda$ be an aggregation function. We say that $\mathcal{F}$ is an* $(\mathcal{S}, \Gamma)$-aggregate pseudorandom function family (also denoted $(\mathcal{S}, \Gamma)$-AGG-PRF) *if there exists an efficient algorithm* $\mathsf{Aggregate}_{k,\mathcal{S},\Gamma}(S)$*: On input a subset $S \in \mathcal{S}$ of the domain, outputs $v \in \mathcal{V}$, such that*

- **Efficient aggregation:** *For every $S \in \mathcal{S}$,* $\mathsf{Aggregate}_{k,\mathcal{S},\Gamma}(S) = \mathsf{AGG}_{k,\mathcal{S},\Gamma}(S)$ *where* $\mathsf{AGG}_{k,\mathcal{S},\Gamma}(S) := \Gamma_{x \in S}\ F_k(x)$.[4][5]
- **Pseudorandomness:** *For all probabilistic polynomial-time (in security parameter λ) algorithms A, and for randomly selected key $k \in K$:*

$$\left| \Pr_{f \leftarrow \mathcal{F}_\lambda}[A^{f_k, AGG_{f_k,\mathcal{S},\Gamma}}(1^\lambda)] - \Pr_{h \leftarrow \mathcal{H}_\lambda}[A^{h, AGG_{h,\mathcal{S},\Gamma}}(1^\lambda)] \right| \leq \mathsf{negl}(\lambda)$$

where $\mathcal{H}_\lambda$ is the set of all functions $D_\lambda \to R_\lambda$.

[4] We omit subscripts on AGG and $\mathsf{Aggregate}$ when clear from context.

[5] AGG is defined to be the correct aggregate value, while $\mathsf{Aggregate}$ is the algorithm by which we compute the value AGG. We make this distinction because while a random function cannot be efficiently aggregated, the aggregate value is still well-defined.

Remark. In this work, we restrict our attention to aggregation functions that treat the range $\mathcal{V}_\lambda = \mathcal{R}_\lambda$ as an Abelian group and compute the group sum (or product) of its inputs. We denote this setting by $\Gamma = \sum$ (or $\prod$, respectively). Supporting other types of aggregation functions (ex: max, a hash) is a direction for future work.

2.1 A General Security Theorem for Aggregate PRFs

How does the security of a function family in the AGG-PRF game relate to security in the normal PRF game (in which A uses only the oracle f and not AGG_f)?

In this section, we show a general security theorem for aggregate pseudo-random functions. Namely, we show that any "sufficiently secure" PRF is also aggregation-secure (for any collection of efficiently recognizable sets and any group-aggregation operation), in the sense of Definition 2.1, by way of an *inefficient* reduction (with overhead polynomial in the size of the domain). In Section 3, we will use this to construct AGG-PRFs from a subexponential-time hardness assumption on the DDH problem. We also show that no such *general* reduction can be efficient, by demonstrating a PRF family that is not aggregation-secure. As a general security theorem cannot be shown without the use of complexity leveraging, this suggests a natural direction for future study: to devise constructions for similarly expressive aggregate PRFs from polynomial assumptions.

Lemma 2.1. *Let $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda>0}$ be a pseudo-random function family where each function $f \in \mathcal{F}_\lambda$ maps a domain $\mathcal{D}_\lambda$ to a range $\mathcal{R}_\lambda$. Suppose there is an adversary A that runs in time $t_A = t_A(\lambda)$ and achieves an advantage of $\epsilon_A = \epsilon_A(\lambda)$ in the aggregate PRF security game for the family $\mathcal{F}$ with an efficiently recognizable set system $\mathcal{S}_\lambda$ and an aggregation function Γ_λ that is computable in time polynomial in its input length. Then, there is an adversary B that runs in time $t_B = t_A + \mathsf{poly}(\lambda, |\mathcal{D}_\lambda|)$ and achieves an advantage of $\epsilon_B = \epsilon_A$ in the standard PRF game for the family $\mathcal{F}$.*

Proof. Let $f_K \leftarrow \mathcal{F}_\lambda$ be a random function from the family $\mathcal{F}_\lambda$. We construct the adversary B which is given access to an oracle $\mathcal{O}$ which is either f_K or a uniformly random function $h : \mathcal{D}_\lambda \to \mathcal{R}_\lambda$.

B works as follows: It queries the PRF on all inputs $x \in \mathcal{D}_\lambda$, builds the function table T_K of f_K and runs the adversary A, responding to its queries as follows:

1. Respond to its PRF query $x \in \mathcal{D}_\lambda$ by returning $T_K[x]$; and
2. Respond to its aggregate query (Γ, S) by (a) going through the table to look up all x such that $x \in S$; and (b) applying the aggregation function honestly to these values.

Finally, when A halts and returns a bit b, B outputs the bit b and halts.
B takes $O(|\mathcal{D}_\lambda|)$ time to build the truth table of the oracle. For each aggregate query (Γ, S), B first checks for each $x \in \mathcal{D}_\lambda$ whether $x \in S$. This takes $|\mathcal{D}_\lambda| \cdot \text{poly}(\lambda)$ time, since S is efficiently recognizable. It then computes the aggregation function Γ over $f(x)$ such that $x \in S$, taking $\text{poly}(|\mathcal{D}_\lambda|)$ time, since Γ is computable in time polynomial in its input length. The total time, therefore, is

$$t_B = t_A + \mathsf{poly}(\lambda, |\mathcal{D}_\lambda|)$$

Clearly, when $\mathcal{O}$ is the pseudo-random function f_K, B simulates an aggregatable PRF oracle to A, and when $\mathcal{O}$ is a random function, B simulates an aggregate random oracle to A. Thus, B has the same advantage in the PRF game as A does in the aggregate PRF game. $\square$

The above gives an inefficient reduction from the PRF security of a function family $\mathcal{F}$ to the AGG-PRF security of the same family running in time polynomial in the size of the domain. Can this reduction be made efficient; that is, can we replace $t_B = t_A + \mathsf{poly}(\lambda)$ into the Lemma 2.1?

This is not possible. Such a reduction would imply that every PRF family that supports efficient aggregate functionality AGG is AGG-PRF secure; this is clearly false. Take for example a pseudorandom function family $\mathcal{F}_0 = \{f : \mathbb{Z}_{2p} \to \mathbb{Z}_p\}$ such that for all f, there is no x with $f(x) = 0$. It is possible to construct such a pseudorandom function family $\mathcal{F}_0$ (under the standard definition). While 0 is not in the image of any $f \in \mathcal{F}_0$, a random function with the same domain and range will, with high probability, have 0 in the image. For an aggregation oracle AGG_f computing *products* over $\mathbb{Z}_p$: $\mathsf{AGG}_f(\mathbb{Z}_{2p}) \neq 0$ if $f \in \mathcal{F}_0$, while $\mathsf{AGG}_f(\mathbb{Z}_{2p}) = 0$ with high probability for random f.

Thus, access to aggregates for products over $\mathbb{Z}_p$[6] would allow an adversary to trivially distinguish $f \in \mathcal{F}_0$ from a truly random map.

2.2 Impossibility of Aggregate PRF for General Sets

It is natural to ask whether whether an aggregate PRF might be constructed for more general sets than we present in Section 3. There we constructed aggregate PRF for the sets of all satisfying assignments for read-once boolean formula and decision trees. As we show in the following, it is impossible to extend this to support the set of satisfying assignmnets for more general circuits.

Theorem 2.2. *Suppose there is an algorithm that has a PRF description K, a circuit C, and a fixed aggregation rule (sum over a finite field, say), and outputs the aggregate value*

$$\sum_{x:C(x)=1} f_K(x)$$

Then, there is an algorithm that takes circuits C as input and w.h.p. over it coins, decides the satisfiability of C.

Proof. The algorithm for SAT simply runs the aggregator with a randomly chosen K, and outputs YES if and only if the aggregator returns 1. The rationale is that if the formula is unsatisfiable, you will always get 0 from the aggregator.[7] Otherwise, you will get $f_K(x)$, where x is the satisfying assignment. (More generally, $\sum_{x:C(x)=1} f_K(x)$). Now, this might end up being 0 accidentally, but cannot be 0 always since otherwise, you will get a PRF distinguisher. The distinguisher has the satisfying assignment hardcoded into it non-uniformly,[8] and it simply checks if $f_K(x) = 0$. □

This impossibility result can be generalized for efficient aggregation of functions that are not pseudo-random. For instance, if $f(x) \equiv 1$ was the constant function 1, the same computing the aggregate over f satisfying inputs to C would not only reveal the satisfiability of C, but even the number of satisfying assignments! In the PRF setting though, it seems that aggregates only reveal the (un)satisfiability of a circuit C, but not the number of satisfying assignments. Further studying

[6]Taken with respect to a set ensemble $\mathcal{S}$ containing, as an element, the whole domain $\mathbb{Z}_{2p}$. While this is not necessary (a sufficiently large subset would suffice), it is the case for the ensembles $\mathcal{S}$ we consider in this work.

[7]This proof may be extended to the case when the algorithm's output is not restricted to be 0 when the input circuit C is unsatisfiable, and even arbitrary outputs for sufficiently expressive classes of circuits.

[8]As pointed out by one reviewer, for sufficiently expressive classes of circuits C, this argument can be made uniform. Specifically, we use distinguish the challenge y from a pseudo-random generator from random by choosing $C := C_y$ that is satisfiable if and only if y is in the PRG image, and modify the remainder of the argument accordingly.

the relationship between the (not necessarily pseudo-random) function f, the circuit representation of C, and the tractability of computing aggregates is an interesting direction. A negative result for a class for which satisfiability (or even counting assignments) is tractable would be very interesting.

3 Constructions of aggregate PRF

In this section, we show several constructions of aggregate PRFs. In Section 3.1, we show a generic construction of aggregate PRFs for intervals (where the aggregation is any group operation). This construction is black-box: given any PRF with the appropriate domain and range, we construct a related family of aggregate PRFs and with no loss in security. In Section 3.2, we show a construction of aggregate PRFs for products over bit-fixing sets (hypercubes), from a strong decisional Diffie-Hellman assumption. We then generalize the DDH construction: in Section 3.3, to the class of sets recognized by polynomial-size decision trees; and in Section 3.4, to sets recognized by read-once Boolean formulas. In these last three constructions, we make use of Lemma 2.1 to argue security.

3.1 Generic Construction for Interval Sets

Our first construction is adapted from [GGN10][9]. The construction is entirely black-box: from any appropriate PRF family $\mathcal{G}$, we construct a related AGG-PRF family $\mathcal{F}$. Unlike the proofs in the sequel, this reduction exactly preserves the security of the starting PRF.

Let $\mathcal{G}_\lambda = \{g_K : \mathbb{Z}_{n(\lambda)} \to R_\lambda\}_{K \in \mathcal{K}_\lambda}$ be a PRF family, with $R = R_\lambda$ being a group where the group operation is denoted by $\oplus$[10]. We construct an aggregatable PRF $\mathcal{F}_\lambda = \{f_K\}_{K \in \mathcal{K}_\lambda}$ for which we can efficiently compute summation of $f_K(x)$ for all x in an interval $[a, b]$, for any $a \leq b \in \mathbb{Z}_n$. Let $\mathcal{S}_{[a,b]} = \{[a, b] \subseteq \mathbb{Z}_n : a, b \in \mathbb{Z}_n; a \leq b\}$ be the set of all interval subsets of $\mathbb{Z}_n$, $[a, b] = \{x \in \mathbb{Z}_n : a \leq x \leq b\}$. Define $\mathcal{F} = \{f_K : \mathbb{Z}_n \to R\}_{K \in \mathcal{K}}$ as follows:

$$f_K(x) = \begin{cases} g_K(0) & : x = 0 \\ g_K(x) \ominus g_K(x-1) & : x \neq 0 \end{cases}$$

Lemma 3.1. *Assuming that $\mathcal{G}$ is a pseudo-random function family, $\mathcal{F}$ is a $(\mathcal{S}_{[a,b]}, \oplus)$-aggregate pseudo-random function family.*

Proof. It follows immediately from the definition of f_K that one can compute the summation of $f_K(x)$ over any interval $[a, b]$. Indeed, rearranging the definition yields

$$\sum_{x \in [0,b]} f_K(x) = g_K(b) \quad \text{and} \quad \sum_{x \in [a,b]} f_K(x) = g_K(b) \oplus -g_K(a-1)$$

We reduce the pseudo-randomness of $\mathcal{F}$ to that of $\mathcal{G}$. The key observation is that each query to the f_K oracle as well as the aggregation oracle for f_K can be answered using at most two black-box calls to the underlying function g_K. By assumption on $\mathcal{G}$, replacing the oracle for g_K with a uniformly random function $h : \mathbb{Z}_n \to R$ is computationally indistinguishable. Furthermore, the function f defined by replacing g by h, namely

$$f'(x) = \begin{cases} h(0) & : x = 0 \\ h(x) \ominus h(x-1) & : x \neq 0 \end{cases}$$

[9]See Example 3.1 and Footnote 18

[10]The only structure of $\mathbb{Z}_n$ we us is the *total order*. Our construction directly applies to any finite, totally-ordered domain D by first mapping D to $\mathbb{Z}_n$, preserving order.

is a truly random function. Thus, the simulated oracle with g_K replaced by h implements a uniformly random function that supports aggregate queries. Security according to Definition 2.1 follows immediately. $\square$

3.2 Bit-Fixing Aggregate PRF from DDH

We now construct an aggregate PRF computing products for bit-fixing sets. Informally, our PRF will have domain $\{0,1\}^{poly(\lambda)}$, and support aggregation over sets like $\{x : x_1 = 0 \wedge x_2 = 1 \wedge x_7 = 0\}$. We will naturally represent such sets by a string in $\{0,1,\star\}^{\text{poly}(\lambda)}$ with 0 and 1 indicating a fixed bit location, and $\star$ indicating a free bit location. We call each such set a 'hypercube.' The PRF will have a multiplicative group $\mathcal{G}$ as its range, and the aggregate functionality will compute group products.

Our PRF is exactly the Naor-Reingold PRF [NR04], for which we demonstrate efficient aggregation and security. We begin by stating the decisional Diffie-Hellman assumption.

Let $\mathcal{G} = \{\mathcal{G}_\lambda\}_{\lambda>0}$ be a family of groups of order $p = p(\lambda)$. The decisional Diffie-Hellman assumption for $\mathcal{G}$ says that the following two ensembles are computationally indistinguishable:

$$\begin{aligned}&\left\{(\mathcal{G}_\lambda, g, g^a, g^b, g^{ab}) : \ G \leftarrow \mathcal{G}_\lambda;\ g \leftarrow G;\ a, b \leftarrow \mathbb{Z}_p\right\}_{\lambda>0}\\ &\qquad\qquad \approx_c \left\{(G, g, g^a, g^b, g^c) : \ G \leftarrow \mathcal{G}_\lambda;\ g \leftarrow G;\ a, b, c \leftarrow \mathbb{Z}_p\right\}_{\lambda>0}\end{aligned}$$

We say that the $(t(\lambda), \epsilon(\lambda))$-DDH assumption holds if for every adversary running in time $t(\lambda)$, the advantage in distinguishing between the two distributions above is at most $\epsilon(\lambda)$.

3.2.1 Construction

Let $\mathcal{G} = \{\mathcal{G}_\lambda\}_{\lambda>0}$ be a family of groups of order $p = p(\lambda)$, each with a canonical generator g, for which the decisional Diffie Hellman (DDH) problem is hard. Let $\ell = \ell(\lambda)$ be a polynomial function. We will construct a PRF family $\mathcal{F}_\ell = \{\mathcal{F}_{\ell,\lambda}\}_{\lambda>0}$ where each function $f \in \mathcal{F}_{\ell,\lambda}$ maps $\{0,1\}^{\ell(\lambda)}$ to $\mathcal{G}_\lambda$. Our PRF family is exactly the Naor-Reingold PRF [NR04]. Namely, each function f is parametrized by $\ell + 1$ numbers $\vec{K} := (K_0, K_1, \ldots, K_\ell)$, where each $K_i \in \mathbb{Z}_p$.

$$f_{\vec{K}}(x_1, \ldots, x_\ell) = g^{K_0 \prod_{i=1}^{\ell} K_i^{x_i}} = g^{K_0 \prod_{i:x_i=1} K_i} \qquad \in \mathcal{G}_\lambda$$

The aggregation algorithm $\mathsf{Aggregate}$ for bit-fixing functions gets as input the PRF key $\vec{K}$ and a bit-fixing string $y \in \{0,1,\star\}^\ell$ and does the following:

- Define the strings K'_i as follows:

$$K'_i = \begin{cases} 1 & \text{if } y_i = 0 \\ K_i & \text{if } y_i = 1 \\ 1 + K_i & \text{otherwise} \end{cases}$$

- Output $g^{K_0 \prod_{i=1}^{\ell} K'_i}$ as the answer to the aggregate query.

Letting $\mathcal{HC} = \{\mathcal{HC}_{\ell(\lambda)}\}_{\lambda>0}$ where $\mathcal{HC}_\ell = \{0,1,\star\}^\ell$ is the set of hypercubes on $\{0,1\}^\ell$, we now prove the following:

Theorem 3.2. *Let $\epsilon > 0$ be a constant, choose the security parameter $\lambda = \Omega(\ell^{1/\epsilon})$, and assume the $(2^{\lambda^\epsilon}, 2^{-\lambda^\epsilon})$-hardness of DDH over the group $\mathcal{G}$. Then, the collection of functions $\mathcal{F}$ defined above is a secure aggregate PRF with respect to the subsets $\mathcal{HC}$ and the product aggregation function over $\mathcal{G}$.*

Correctness. We show that the answer we computed for an aggregate query $y \in \{0, 1, \star\}^\lambda$ is correct. Define the sets

$$\mathsf{Match}(y) := \{x \in \{0,1\}^\lambda : \ \forall i, y_i = \star \text{ or } x_i = y_i\} \text{ and } \mathsf{Fixed}(y) := \{i \in [\lambda] : y_i \in \{0,1\}\}$$

Thus, $\mathsf{Match}(y)$ is the set of all 0-1 strings x that match all the fixed locations of y, but can take any value on the wildcard locations of y. $\mathsf{Fixed}(y)$ is the set of all locations i where the bit y_i is fixed. Note that:

$$\begin{aligned}
\mathsf{AGG}(\vec{K}, y) &= \textstyle\prod_{x \in \mathsf{Match}(y)} f_{\vec{K}}(x) && \text{(by definition of } \mathsf{AGG}\text{)} \\
&= \textstyle\prod_{x \in \mathsf{Match}(y)} g^{K_0 \prod_{i=1}^{\ell} K_i^{x_i}} && \text{(by definition of } f_{\vec{K}}\text{)} \\
&= g^{K_0 \sum_{x \in \mathsf{Match}(y)} \prod_{i=1}^{\ell} K_i^{x_i}} && \\
&= g^{K_0 \left(\prod_{i \in \mathsf{Fixed}(y)} K_i^{y_i}\right) \cdot \left(\prod_{i \in [\ell] \setminus \mathsf{Fixed}(y)} (1+K_i)\right)} && \text{(inverting sums and products)} \\
&= g^{K_0 \prod_{i=1}^{\ell} K_i'} && \text{(by definition of } K_i'\text{)} \\
&= \mathsf{Aggregate}(\vec{K}, y) && \text{(by definition of } \mathsf{Aggregate}\text{)}
\end{aligned}$$

Security. We will rely on the following theorem from [NR04].

Theorem 3.3 (Theorem 4.1, [NR04])**.** *Suppose there is an adversary A that runs in time $t(\lambda)$ and has an advantage of $\gamma(\lambda)$ in the (regular) PRF game. Then, there is an adversary B that runs in time $\mathsf{poly}(\lambda) \cdot t(\lambda)$ and breaks the DDH assumption with advantage $\gamma(\lambda)/\lambda$.*

The aggregate PRF security proof proceeds as follows. First, we choose the security parameter $\lambda = \Omega(\ell^{1/\epsilon})$ as in the theorem statement. We use Lemma 2.1 to conclude that if there is an adversary distinguisher D breaking the aggregate PRF security of $\mathcal{F}$ in $\mathsf{poly}(\lambda)$ time with $1/\mathsf{poly}(\lambda)$ advantage, then there is an adversary A that breaks the regular PRF security of $\mathcal{F}$ in $\mathsf{poly}(\lambda) \cdot 2^{O(\ell)} = \mathsf{poly}(\lambda) \cdot 2^{\lambda^\epsilon} = 2^{O(\lambda^\epsilon)}$ time with $1/\mathsf{poly}(\lambda)$ advantage. Using Theorem 3.3 now tells us that there is an adversary B that wins the DDH distinguishing game in $2^{O(\lambda^\epsilon)}$ time with $1/\mathsf{poly}(\lambda)$ advantage, breaking the subexponential DDH assumption. This establishes the aggregate security of the PRF and thus Theorem 3.2.

Obtaining a security proof based on polynomial assumptions is an interesting open question.

3.3 Decision Trees

We generalize the previous construction from DDH to support sets specified by polynomial-sized decision trees by observing that such decision trees can be written as disjoint unions of hypercubes.

A decision tree family $\mathcal{T}_\lambda$ of size $p(\lambda)$ over $\ell(\lambda)$ variables consists of binary trees with at most $p(\lambda)$ nodes, where each internal node is labeled with a variable x_i for $i \in [\ell]$, the two outgoing edges of an internal node are labeled 0 and 1, and the leaves are labeled with 0 or 1. On input an $x \in \{0, 1\}^\ell$, the computation of the decision tree starts from the root, and upon reaching an internal node n labeled by a variable x_i, takes either the 0-outgoing edge or the 1-outgoing edge out of the node n, depending on whether x_i is 0 or 1, respectively.

We now show how to construct a PRF family $\mathcal{F}_\ell = \{\mathcal{F}_{\ell,\lambda}\}_{\lambda>0}$ where each $\mathcal{F}_{\ell,\lambda}$ consists of functions that map $\mathcal{D}_\lambda := \{0,1\}^\ell$ to a group $\mathcal{G}_\lambda$, that supports aggregation over sets recognized by decision trees. That is, let $\mathcal{S}_\lambda = \{S \subseteq \{0,1\}^\ell : \exists$ a decision tree $T_S \in \mathcal{T}_\lambda$ that recognizes $S\}$.

Our construction uses a hypercube-aggregate PRF family $\mathcal{F}'_\ell$ as a sub-routine. First, we need the following simple lemma.

Lemma 3.4 (Decision Trees as Disjoint Unions of Hypercubes)**.** *Let $S \subseteq \{0,1\}^\ell$ be recognized by a decision tree T_S of size $p = p(\lambda)$. Then, S is a disjoint union of at most p hybercubes $H_{y_1}, \ldots, H_{y_p}$, where each $y_i \in \{0,1,\star\}^\ell$ and $H_{y_i} = \mathsf{Match}(y_i)$. Furthermore, given T_S, one can in polynomial time compute these hypercubes.*

Given the lemma, $\mathsf{Aggregate}$ is simple: on input a set S represented by a decision tree T_S, compute the disjoint hypercubes $H_{y_1}, \ldots, H_{y_p}$. Run the hypercube aggregation algorithm to compute

$$g_i \leftarrow \mathsf{Aggregate}_{\mathcal{F}}(K, y_i)$$

and outputs $g := \prod_{i=1}^{p} g_i$.

Basing the construction on the hypercube-aggregate PRF scheme from Section 3.2, we get a decision tree-aggregate PRF based on the sub-exponential DDH assumption. The security of this PRF follows from Lemma 2.1 by an argument identical to the one in Section 3.2.

3.4 Read-once formulas

Read-once boolean formula provide a different generalization of hypercubes and they too admit an efficient aggregation algorithm for the Naor-Reingold PRF, with a similar security guarantee.

A boolean formula on ℓ variables is a circuit on $x = (x_1, \ldots, x_\ell) \in \{0,1\}^\ell$ composed of only AND, OR, and NOT gates. A *read-once boolean formula* is a boolean formula with fan-out 1, namely each input literal feeds into at most one gate, and each gate output feeds into at most one other gate.[11] Let R_λ be the family of all read-once boolean formulas over $\ell(\lambda)$ variables. Without loss of generality, we restrict these circuits to be in a standard form: namely, composed of fan-in 2 and fan-out 1 AND and OR gates, and any NOT gates occurring at the inputs.

In this form, the circuit for any read-once boolean formula can be identified with a labelled binary tree; we identify a formula by the label of its root C_ϕ. Nodes with zero children are variables or their negation, labelled by x_i or $\bar{x}_i$, while all other nodes have 2 children and represent gates with fan-in 2. For such a node with label C, its children have labels C_L and C_R. Note that each child is itself a read-once boolean formula on fewer inputs, and their inputs are disjoint Let the gate type of a node C be $\mathsf{type}(C) \in \{AND, OR\}$.

We describe a recursive aggregation algorithm for computing products of PRF values over all accepting inputs for a given read-once boolean formula C_ϕ. Looking forward, we require the formula to be read-once in order for the recursion to be correct. The algorithm described reduces to that of Section 3.2 in the case where ϕ describes a hypercube.

3.4.1 Construction

The aggregation algorithm for read-once Boolean formulas takes as input the PRF key $\vec{K} = (K_0, \ldots, K_\ell)$ and a formula $C_\phi \in R_\lambda$ where C_ϕ only reads the variables $x_1, \ldots, x_m$ for some $m \leq \ell$. We abuse notation and interpret C_ϕ to be a formula on both $\{0,1\}^\ell$ and $\{0,1\}^m$ in the natural way.

[11] We allow a formula to ignore some inputs variables; this enables the model to express hypercubes directly.

$$
\begin{aligned}
\mathsf{AGG}_{k,\prod}(C_\phi) &= \prod_{x:C_\phi(x)=1} g^{K_0 \prod_{i\in[\ell]} K_i^{x_i}} && (1)\\
&= g^{K_0 \sum_{x:C_\phi(x)=1} \prod_{i\in[\ell]} K_i^{x_i}} && (2)\\
&= g^{K_0 \cdot A(C_\phi,1)\cdot \prod_{m<j\le \ell}(1+K_i)} && (3)
\end{aligned}
$$

where we define $A(C,1) := \sum_{\{x\in\{0,1\}^m : C(x)=1\}} \prod_{i\in[m]} K_i^{x_i}$. If $A(C,1)$ is efficiently computable, then $\mathsf{Aggregate}$ will simply compute it and return (3). To this end, we provide a recursive procedure for computing $A(C,1)$.

Generalizing the definition for any sub-formula C with variables named x_1 to x_m, define the values $A(C,0)$ and $A(C,1)$:

$$A(C,b) := \sum_{\{x\in\{0,1\}^m :\ C(x)=b\}} \quad \prod_{i\in[m]} K_i^{x_i}.$$

Recursively compute $A(C,b)$ as follows:

- If C is a literal for variable x_i, then by definition:

$$A(C,b) = \begin{cases} K_i & \text{if } C = x_i \\ 1 & \text{if } C = \bar{x}_i \end{cases}$$

- Else, if $\mathsf{type}(C) = AND$: Let C_L and C_R be the children of C. By hypothesis, we can recursively compute $A(C_L,b)$ and $A(C_R,b)$ for $b\in\{0,1\}$. Compute $A(C,b)$ as:

$$
\begin{aligned}
&A(C,1) = A(C_L,1)\cdot A(C_R,1)\\
&A(C,0) = A(C_L,0)\cdot A(C_R,0) + A(C_L,1)\cdot A(C_R,0) + A(C_L,0)\cdot A(C_R,1)
\end{aligned}
$$

- Else, $\mathsf{type}(C) = OR$: Let C_L and C_R be the children of C. By hypothesis, we can recursively compute $A(C_L,b)$ and $A(C_R,b)$ for $b\in\{0,1\}$. Compute $A(C,b)$ as:

$$
\begin{aligned}
&A(C,1) = A(C_L,1)\cdot A(C_R,1) + A(C_L,1)\cdot A(C_R,0) + A(C_L,0)\cdot A(C_R,1)\\
&A(C,0) = A(C_L,0)\cdot A(C_R,0))
\end{aligned}
$$

Lemma 3.5. *$A(C,b)$ as computed above is equal to $\sum_{\{x\in\{0,1\}^m :\ C(x)=b\}} \ \prod_{i\in[m]} K_i^{x_i}$*

Proof. For C a literal, the correctness is immediate. We must check the recursion for each $\mathsf{type}(C)\in\{AND,OR\}$ and $b\in\{0,1\}$. We only show the case for $b=1$ when C is an OR gate; the other three cases can be shown similarly.

Let $S_{b_L,b_R} = \{x = (x_L,x_R) : (C_L(x_L), C_R(x_R) = (b_L,b_R)\}$ be the set of inputs (x_L,x_R) to C such that $C_L(x_L) = b_L$ and $C_R(x_R) = b_R$. The set $\{x : C(x) = 1\}$ can be decomposed into the disjoint union $S_{0,1}\sqcup S_{1,0}\sqcup S_{1,1}$. Furthermore,

$$A(C,1) = \sum_{x\in S_{0,1}} \prod_{i\in[m]} K_i^{x_i} + \sum_{x\in S_{1,0}} \prod_{i\in[m]} K_i^{x_i} + \sum_{x\in S_{1,1}} \prod_{i\in[m]} K_i^{x_i}$$

Because C is read-once, the sets of inputs on which C_L and C_R depend are disjoint; this implies that $A(C_L,b_L)\cdot A(C_R,b_R) = \sum_{x\in S_{b_L,b_R}} \prod_{i\in[m]} K_i^{x_i}$, yielding the desired recursion. □

Theorem 3.6. *Let $\epsilon > 0$ be a constant, choose the security parameter $\lambda = \Omega(\ell^{1/\epsilon})$, and assume $(2^{\lambda^\epsilon}, 2^{-\lambda^\epsilon})$-hardness of the DDH assumption. Then, the collection of functions $\mathcal{F}_\lambda$ defined above is a secure aggregate PRF with respect to the subsets R_λ and the product aggregation function over the group $\mathcal{G}$.*

Proof. Correctness is immediate from Lemma 3.5, and Equation (3). Security follows from the decisional Diffie-Hellman assumption in much the same way it did in the case of bit-fixing functions. □

4 Connection to Learning

4.1 Preliminaries

Notation: For a probability distribution D over a set X, we denote by $x \leftarrow D$ to mean that x is sampled according to D, and $x \leftarrow X$ to denote uniform sampling form X. For an algorithm A and a function $\mathcal{O}$, we denote that A has oracle access to $\mathcal{O}$ by $A^{\mathcal{O}(\cdot)}$.

We recall the definition of a "concept class". In this section, we will often need to explicitly reason about the representations of the concept classes discussed. Therefore we make use of the notion of a "representation class" as defined by [KV94] alongside that of concept classes. This unified formalization enables us to discuss both these traditional learning models (namely, PAC and learning with membership queries) as well as the new models we present below. Our definitions are parametrized by $\lambda \in \mathbb{N}$.[12]

Definition 4.1 (Representation class [KV94])**.** *Let $K = \{K_\lambda\}_{\lambda\in\mathbb{N}}$ be a family of sets, where each $k \in K_\lambda$ has description in $\{0,1\}^{s_k(\lambda)}$ for some polynomial $s_k(\cdot)$. Let $X = \{X_\lambda\}_{\lambda\in\mathbb{N}}$ be a set, where each X_λ is called a* domain *and each $x \in X_\lambda$ has description in $\{0,1\}^{s_x(\lambda)}$ for some polynomial $s_x(\cdot)$. With each λ and each $k \in K_\lambda$, we associate a Boolean function $f_k : X_\lambda \to \{0,1\}$.*[13] *We call each such function f_k a* concept*, and k its* index *or its* description*. For each λ, we define the* concept class *$C_\lambda = \{f_k : k \in K_\lambda\}$ to be the set of all concepts with index in K_λ. We define the* representation class *$C = \{C_\lambda\}$ to be the union of all concept classes C_λ.*

This formalization allows us to easily associate complexity classes with concepts in learning theory. For example, to capture the set of all DNF formulas on λ inputs with size at most $p(\lambda)$ for a polynomial p, we will let $X_\lambda = \{0,1\}^\lambda$, and $K_\lambda^{p(\lambda)}$ be the set of descriptions of all DNF formulas on λ variables with size at most $p(\lambda)$ under some reasonable representation. Then a concept $f_k(x)$ evaluates the formula k on input x. Finally, $\mathsf{DNF}_\lambda^{p(\lambda)} = \{f_k : k \in K_\lambda^{p(\lambda)}\}$ is the concept class, and $\mathsf{DNF}^{p(\lambda)} = \{DNF_\lambda^{p(\lambda)}\}_{\lambda\in\mathbb{N}}$. $DNF^{p(\lambda)}$ is the representation class that computes all DNF formulas on λ variables with description of size at most $p(\lambda)$ in the given representation.

As a final observation, note that a Boolean-valued PRF family $\mathcal{F} = \{\mathcal{F}_\lambda\}$ where $\mathcal{F}_\lambda = \{f_k : X_\lambda \to \{0,1\}\}$ with keyspace $K = \{K_\lambda\}$ and domain $X = \{X_\lambda\}$ satisfies the syntax of a representation class as defined above. This formalization is useful precisely because it captures both PRF families and complexity classes, enabling lower bounds in various learning models.

In proving lower bounds for learning representation classes, it will be convenient to have a notion of containment for two representation classes.

[12] When clear from the context, we will omit the subscript λ.

[13] This association is an efficient procedure for evaluating f_k. Concretely, we might consider that there is a universal circuit F_λ such that for each λ, $f_k(\cdot) = F_\lambda(k, \cdot)$.

Definition 4.2 ($\subseteq$)**.** *For two representation classes $\mathcal{F} = \{\mathcal{F}_\lambda\}$ and $\mathcal{G} = \{\mathcal{G}_\lambda\}$ on the same domain $X = \{X_\lambda\}$, and with indexing sets $I = \{I_\lambda\}$ and $K = \{K_\lambda\}$ respectively, we say $\mathcal{F} \subseteq \mathcal{G}$ if for all sufficiently large λ, for all $i \in I_\lambda$, there exists $k \in K_\lambda$ such that $g_k \equiv f_i$.*

Informally, if a representation class contains a PRF family, then this class is hard to MQ-learn (as in [Val84]). We apply similar reasoning to more powerful learning models. For example, if $\mathcal{G}$ is the representation class $DNF^{p(\lambda)}$ as defined above, then $\mathcal{F} \subseteq DNF^{p(\lambda)}$ is equivalent to saying that for all sufficiently large λ, the concept class $\mathcal{F}_\lambda$ can be decided by a DNF on λ inputs of $p(\lambda)$ size.

We now recall some standard definitions.

Definition 4.3 (ϵ-approximation)**.** *Let $f, h : X \to \{0,1\}$ be arbitrary functions. We say h ϵ-*approximates *f if $\Pr_{x \leftarrow X}[h(x) \neq f(x)] \leq \epsilon$.*

In general, ϵ-approximation is considered under a general distribution on X, but we will consider only the uniform distribution in this work.

Definition 4.4 (PAC learning)**.** *For a concept $f : X_\lambda \to \{0,1\}$, and a probability distribution D_λ over X_λ, the example oracle $EX(f, D_\lambda)$ takes no input and returns $(x, f(x))$ for $x \leftarrow D_\lambda$. An algorithm $\mathcal{A}$ is an (ϵ, δ)-PAC learning algorithm for representation class $\mathcal{C}$ if for all sufficiently large λ, $\epsilon = \epsilon(\lambda) > 0$, $\delta = \delta(\lambda) > 0$ and $f \in \mathcal{C}_\lambda$,*

$$\Pr[\mathcal{A}^{EX(f,D_\lambda)} = h : \ h \text{ is an } \epsilon\text{-approximation to } f] \geq 1 - \delta$$

Definition 4.5 (MQ learning)**.** *For a concept $f : X_\lambda \to \{0,1\}$, the membership oracle $MEM(f)$ takes as input a point $x \in X_\lambda$ and returns $f(x)$. An algorithm $\mathcal{A}$ is an (ϵ, δ)-MQ learning algorithm for representation class $\mathcal{C}$ if for all sufficiently large λ, $\epsilon = \epsilon(\lambda) > 0, \delta = \delta(\lambda) > 0$, and $f \in \mathcal{C}_\lambda$,*

$$\Pr[\mathcal{A}^{MEM(f)} = h : \ h \text{ is an } \epsilon\text{-approximation to } f] \geq 1 - \delta$$

We consider only PAC learning *with uniform examples*, where D_λ is the uniform distribution over X_λ. In this case, MQ is strictly stronger than PAC: everything that is PAC learnable is MQ learnable.

Observe that for any $f : X_\lambda \to \{0,1\}$, either $h(x) = 0$ or $h(x) = 1$ will $\frac{1}{2}$-approximate f. Furthermore, if $\mathcal{A}$ is inefficient, f may be learned exactly. For a learning algorithm to be non-trivial, we require that it is *efficient* in λ, and that it at least *weakly* learns $\mathcal{C}$.

Definition 4.6 (Efficient- and weak- learning)**.**

- *$\mathcal{A}$ is said to be* efficient *if the time complexity of $\mathcal{A}$ and h are polynomial in $1/\epsilon, 1/\delta$, and λ.*
- *$\mathcal{A}$ is said to* weakly *learn $\mathcal{C}$ if there exist some polynomials $p_\epsilon(\lambda), p_\delta(\lambda)$ for which $\epsilon \leq \frac{1}{2} - \frac{1}{p_\epsilon(\lambda)}$ and $\delta \leq 1 - \frac{1}{p_\delta(\lambda)}$.*
- *We say a representation class is* learnable *if it is both efficiently and weakly learnable. Otherwise, it is* hard to learn*.*

Lastly, we recall the efficiently recognizable ensembles of sets as defined in Section 2. We occasionally call such ensembles indexed, or succinct. Throughout this section, we require this property of our set ensembles $\mathcal{S}$. Both the $\mathrm{MQ}_{\mathrm{RA}}$ and AQ learning models that we present are defined with respect to $\mathcal{S} = \{\mathcal{S}_\lambda\}$, an efficiently recognizable ensemble of subsets of the domain X_λ.

4.2 Membership queries with restriction access

In the PAC-with-Restriction Access model of learning of Dvir, et al [DRWY12], a powerful generalization of PAC learning is studied: rather than receiving random examples of the form $(x, f(x))$ for the concept f, the learning algorithm receives a random "restriction" of f - an implementation of the concept for a subset of the domain. Given this implementation of the restricted concept, the learning algorithm can both evaluate f on many related inputs, and study the properties of the restricted implementation itself. We consider an even stronger setting: instead of receiving random restrictions, the learner can adaptively request any restriction from a specified class $\mathcal{S}$. We call this model *membership queries with restriction access (MQ_{RA})*.

As a concrete example to help motivate and understand the definitions, we consider DNF formulas. For a DNF formula ϕ, a natural restriction might set the values of some of the variables. Consequently, some literals and clauses may have their values determined, yielding a simpler DNF formula ϕ' which agrees with ϕ on this restricted domain. This is the 'restricted concept' that the learner receives.

This model is quite powerful; indeed, decision trees and DNFs are efficiently learnable in the PAC-with-restriction-access learning model whereas neither is known to be learnable in plain PAC model [DRWY12]. Might this access model be too powerful or are there concepts that cannot be learned?

Looking forward, we will show that constrained PRFs correspond to hard-to-learn concepts in the $\mathrm{MQ_{RA}}$learning model. In the remainder, we will formally define the learning model, define constrained PRFs, and prove the main lower bound of this section.

4.2.1 $\mathbf{MQ_{RA}}$learning

While the original restriction access model only discusses restrictions fixing individual input bits for a circuit, we consider more general notions of restrictions.

Definition 4.7 (Restriction)**.** *For a concept $f : X_\lambda \to \{0,1\}$, a* restriction *$S \subseteq X_\lambda$ is a subset of the domain. The* restricted concept *$f|_S : S \to \{0,1\}$ is equal to f on S.*

While general restrictions can be studied, we consider the setting in which all restrictions S are in a specified set of restrictions $\mathcal{S}$. For a DNF formula ϕ, a restriction might be $S = \{x : x_1 = 1 \wedge x_4 = 0\}$. This restriction is contained in the set of 'bit-fixing' restrictions in which individual input bits are fixed. In fact, this class of restrictions is all that is considered in [DRWY12]; we generalize their model by allowing more general classes of restrictions.

In the previous example, a restricted DNF can be naturally represented as another DNF. More generally, we allow a learning algorithm to receive representations of restricted concepts. These representations are computed according to a Simplification Rule.[14]

Definition 4.8 (Simplification Rule)**.** *For each λ, let $\mathcal{C}_\lambda = \{f_k : X_\lambda \to \{0,1\}\}_{k \in K_\lambda}$ be a concept class, $\mathcal{S}_\lambda$ an efficiently recognizable ensemble of subsets of X_λ, and $S \in \mathcal{S}_\lambda$ be a restriction. A* simplification *of $f_k \in C_\lambda$ according to S is the description $k_S \in K_\lambda$ of a concept f_{K_S} such that $f_{k_S} = f_k|_S$. A* simplification rule *for $\mathcal{C} = \{\mathcal{C}_\lambda\}$ and $\mathcal{S} = \{\mathcal{S}_\lambda\}$ is a mapping $\mathsf{Simp}_\lambda : (k, S) \mapsto k_S$ for all $k \in K_\lambda, S \in \mathcal{S}_\lambda$.*

[14] Whereas a DNF with some fixed input bits is naturally represented by a smaller DNF, wehen considering general representation classes and general restrictions, this is not always the case. Indeed, the simplification of f according to S may be in fact more complex. We use the term "Simplification Rule" for compatibility with [DRWY12].

In the PAC-learning with restriction access (PAC_{RA}) learning model considered in [DRWY12], the learner only receives random restrictions. Instead, we consider the setting where the learner can adaptively request any restriction from a specified class $\mathcal{S}$. This model – which we call *membership queries learning with restriction access (MQ_{RA})* – is a strict generalization of PAC_{RA} for efficiently samplable distributions over restrictions (including all the positive results in [DRWY12]). Further observe that this strictly generalizes the membership oracle of MQ learning if $\mathcal{S}$ is such that for each x, it is easy to find a restriction S covering x.

In traditional learning models (PAC, MQ) it is trivial to output a hypothesis that $\frac{1}{2}$-approximates any concept f; a successful learning algorithm is required to learn substantially more than half of the concept. With restriction queries, the learning algorithm is explicitly given the power to compute on some fraction α of the domain. Consequently, outputting an $\epsilon \geq (\frac{1-\alpha}{2})$-approximation to f is trivial; we require a successful learning algorithm to do substantially better. This reasoning is reflected in the definition of *weak* $\mathrm{MQ_{RA}}$learning below.

Definition 4.9 (Membership queries with restriction access ($\mathrm{MQ_{RA}}$))**.** *In a given execution of an oracle algorithm $\mathcal{A}$ with access to a restriction oracle* Simp*, let $X_S \subseteq X_\lambda$ be the union of all restrictions $S \in \mathcal{S}_\lambda$ queried by $\mathcal{A}$. $\mathcal{S}$ is an efficiently recognizable ensemble of subsets of the domain X_λ.*

An algorithm $\mathcal{A}$ is an $(\epsilon, \delta, \alpha)$-$\mathrm{MQ_{RA}}$learning algorithm *for representation class $\mathcal{C}$ with respect to a restrictions in $\mathcal{S}$ and simplification rule* Simp *if, for all sufficiently large λ, for every $f_k \in \mathcal{C}_\lambda$,* $\Pr[\mathcal{A}^{\mathsf{Simp}(k,\cdot)} = h] \geq 1 - \delta$ *where h is an ϵ-approximation to f, – and furthermore – $|X_S| \leq \alpha |X_\lambda|$.*

$\mathcal{A}$ is said to weakly $\mathrm{MQ_{RA}}$-learn *if $\alpha \leq 1 - \frac{1}{p_\alpha(\lambda)}$, $\epsilon \leq (1-\alpha)(\frac{1}{2} - \frac{1}{p_\epsilon(\lambda)})$, $\delta \leq 1 - \frac{1}{p_\delta(\lambda)}$ for some polynomials $p_\alpha, p_\epsilon, p_\delta$.*

4.2.2 Constrained PRFs

We look to constrained pseudorandom functions for hard-to-learn concepts in the restriction access model. To support the extra power of the restriction access model, our PRFs will need to allow efficient evaluation on restrictions of the domain while maintaining some hardness on the remainder. Constrained PRFs [KPTZ13a, BGI14a, BW13a] provide just this power. For showing hardness of restriction access learning, the constrained keys will correspond to restricted concepts; the strong pseudorandomness property will give the hardness result.

Definition: Syntax A family of functions $\mathcal{F} = \{F_\lambda : K_\lambda \times X_\lambda \to Y_\lambda\}$ is said to be *constrained* with respect to a set system $\mathcal{S}$, if it supports the additional efficient algorithms:

- $Constrain_\lambda(k, S)$: A randomized algorithm, on input $(k, S) \in K_\lambda \times \mathcal{S}_\lambda$, outputs a *constrained key* k_S. We $\tilde{K}_\lambda \triangleq Support(Constrain(k, S))$ the set of all constrained keys.
- $Eval_\lambda(k_S, x)$: A deterministic algorithms taking input $(k_S, x) \in \tilde{K}_\lambda \times X_\lambda$, and satisfying the following correctness guarantee:

$$Eval(Constrain(k, S), x) = \begin{cases} F(k, x) & \text{if } x \in S \\ \perp \notin Y & \text{otherwise.} \end{cases}$$

Definition: Security Game

- C picks a random key $k \in K_\lambda$ and initializes two empty subsets of the domain: $C, V = \emptyset$. C and V are subsets of X_λ which must satisfy the invariant that $C \cap V = \emptyset$. C will keep track the inputs $x \in X_\lambda$ to the Challenge oracle, and V will be the union of all sets S queries to Constrain plus all points $x \in X_\lambda$ to the Eval oracle.

- C picks $b \in \{0,1\}$ to run EXP(b), and exposes the following three oracles to A :

  $\mathsf{Eval}(x)$: On input $x \in X_\lambda$, outputs $F(k,x)$. $V \leftarrow V \cup \{x\}$.

  Constrain(S): On input $S \in \mathcal{S}_\lambda$, outputs k_S. $V \leftarrow W \cup S$.

  Challenge(x): On input $x \in X_\lambda$, outputs:

$$\begin{array}{ll} F(k,x) & \text{in EXP(0)} \\ y \leftarrow Y_\lambda & \text{in EXP(1)} \end{array}.$$

  In EXP(1), the responses to Challenge are selected uniformly at random from the range, with the requirement that the responses be consistent for identical inputs x.

- The adversary queries the oracles with the requirement that $C \cap V = \emptyset$, and outputs a bit $b' \in \{0,1\}$.

  Definition 4.10. *The advantage is defined as* $ADV_\lambda^{cPRF}(\mathsf{A}) := \Pr[b' = b]$ *in the above security game.*

Definition 4.11 (Constrained PRF (cPRF))**.** *A family of functions* $\mathcal{F} = \{F_\lambda : K_\lambda \times X_\lambda \to Y_\lambda\}$ *constrained with respect to* $\mathcal{S}$ *is a* constrained PRF *if for all probabilistic polynomial-time adversaries* A *and for all sufficiently large* λ *and all polynomials* $p(n)$*:*

$$ADV_\lambda^{cPRF}(\mathsf{A}) < \frac{1}{2} + \frac{1}{p(n)},$$

over the randomness of C *and* A*.*

4.2.3 Hardness of restriction access Learning

We will now prove that if a constrained PRF $\mathcal{F}$ with respect to set system $\mathcal{S}$ is computable in representation class $\mathcal{C}$, then $\mathcal{C}$ hard to $\mathrm{MQ}_{\mathrm{RA}}$-learn with respect to $\mathcal{S}$ and some simplification rule.

Theorem 4.1. *Let* $\mathcal{F} = \{\mathcal{F}_\lambda\}$ *be a Boolean-valued constrained PRF (also interpreted as a representation class) with respect to sets* $\mathcal{S}$ *and key-space* K*. Let* $EVAL = \{EVAL_\lambda\}$ *be a representation class where each* $EVAL_\lambda$ *is defined as:*

$$EVAL_\lambda = \big\{ g_{k_S}(\cdot) : \; g_{k_S}(x) = \mathsf{PRF.Eval}(k_S, x) \big\}.$$

Namely, each concept in the class $EVAL_\lambda$ *is indexed by* $k_S \in \tilde{K}_\lambda$ *and has* X_λ *as its domain. For any representation class* $\mathcal{C} = \{\mathcal{C}_\lambda\}$ *such that* $\mathcal{F} \subseteq \mathcal{C}$ *and* $EVAL \subseteq \mathcal{C}$*, there exists a simplification rule* Simp *such that* $\mathcal{C}$ *is hard to* MQ_{RA}*-learn with respect to the set of restrictions* $\mathcal{S}$ *and the simplification rule* Simp*.*

Existing constructions of constrained PRFs [BW13a] yield the following corollaries:

Corollary 4.2. *Let $n = n(\lambda)$ be a polynomial, and assume that for the $n+1$-MDDH problem, every adversary time* $\mathrm{poly}(\lambda)$ *the advantage is at most $\epsilon(\lambda)/2^n$. Then there is a simplification rule such that NC^1 is hard to MQ_{RA}-learn with respect to restrictions in $\mathcal{HC}$*[15].

Corollary 4.3. *Assuming the existence of one-way functions, there is a simplification rule such that $P/poly$ is hard to MQ_{RA}-learn with respect to restrictions in $\mathcal{S}_{[a,b]}$* [16].

Remarks: The Simplification Rule here is really the crux of the issue. In our theorem, *there exists* a simplification rule under which we get a hardness result. This may seem somewhat artificial. On the other hand, this implies that the restriction-access learnability (whether PAC- or MQ-RA) of a concept class crucially depends on the simplification rule, as the trivial simplification rule of $Simp(k, S) = k$ admits a trivial learning-algorithm in either setting. This work reinforces that the choice simplification rule can affect the learnability of a given representation class. Positive results for restriction access learning that were independent of the representation would be interesting.

Proof of Theorem 4.1. We interpret $\mathcal{F} = \{\mathcal{F}_\lambda\}$ as a representation class. For each λ, the concepts $f_k \in \mathcal{F}_\lambda$ are indexed by K_λ and have domain X_λ. Let $EVAL = \{EVAL_\lambda\}$ be a representation class defined as in the theorem statement. The indexing set for $EVAL_\lambda$ is $\tilde{K}_\lambda$, the set of constrained keys k_S for $k \in K_\lambda, S \in \mathcal{S}_\lambda$.

Let $\mathcal{C} = \{\mathcal{C}_\lambda\}$ be a representation class, with domain X_λ and indexing set I_λ. For $i \in I_\lambda$, c_i is a concept in $\mathcal{C}_\lambda$.

By hypothesis, $\mathcal{F} \subseteq \mathcal{C}$: for sufficiently large λ, for all $k \in K_\lambda$ there exists $i \in I_\lambda$ such that $c_i \equiv f_k$. Similarly, for all $k_S \in \tilde{K}_\lambda$ there exists $i \in I_\lambda$ such that $c_i \equiv Eval_\lambda(k_S, \cdot)$. For concreteness, let M_λ be this map from $K_\lambda \cup \tilde{K}_\lambda$ to I_λ.[17]

We can now specify the simplification rule $\mathsf{Simp}_\lambda : I_\lambda \times \mathcal{S}_\lambda \to I_\lambda$. Letting $M_\lambda(K_\lambda) \subseteq I_\lambda$ be the image of K_λ under M_λ:

$$\mathsf{Simp}_\lambda(i, S) = \begin{cases} M_\lambda(Constrain_\lambda(M_\lambda^{-1}(i), S)) & \text{if } i \in M_\lambda(K_\lambda) \\ i & \text{otherwise.} \end{cases}$$

For example, i may be a circuit computing the PRF f_k for some $k = M^{-1}(i)$. The simplification computes the circuit corresponding to a constrained PRF key, if the starting circuit already computes a member of the PRF family $\mathcal{F}_\lambda$.[18]

Reduction: Suppose, for contradiction, that there exists an such an efficient learning algorithm $\mathcal{A}$ for $\mathcal{C}$ as in the statement of the theorem. We construct algorithm $\mathcal{B}$ breaking the constrained PRF security. In the PRF security game, $\mathcal{B}$ is presented with the oracles $f_k(\cdot)$, $Constrain_\lambda(k, \cdot)$, and $Challenge_\lambda(\cdot)$, for some $k \leftarrow K_\lambda$. Run $\mathcal{A}$, and answer queries $S \in \mathcal{S}_\lambda$ to the restriction oracle by querying $Constrain_\lambda(k, S)$, receiving k_S, and returning $M_\lambda(k_S)$. Once $\mathcal{A}$ terminates, it outputs hypothesis h. By assumption on $\mathcal{A}$, with probability at least $1 - \delta > \frac{1}{p_\delta(\lambda)}$, the hypothesis h is an ϵ-approximation of $c_{M(k)} \equiv f_k$ with $\epsilon \leq \frac{1-\alpha}{2}$ and $\alpha < 1 - \frac{1}{p_\alpha(\lambda)}$.

After receiving hypothesis h, $\mathcal{B}$ estimates the probability $\Pr_{x \leftarrow X \setminus X_S}[h(x) = Challenge_\lambda(x)]$. In EXP(0), this probability is at least $1 - \epsilon$ with probability at least $1 - \delta$; in EXP(1), it is exactly $1/2$.

[15] as defined in Section 3.

[16] as defined in Section 3.

[17] This is a non-uniform reduction.

[18] Note that while the inverse map M_λ^{-1} may be inefficient, in our reduction, the concept in question is represented by a PRF key k. Thus $\mathcal{B}$ must only compute the forward map M_λ.

To sample uniform $x \in X \setminus X_S$, we simply take a uniform $x \in X$: with probability $1-\alpha \geq 1/p_\alpha(n)$, $x \in X \setminus X_S$. Thus, $\mathcal{B}$ runs in expected polynomial time. If the estimate is close to ϵ, guess EXP(0); otherwise, flip an fair coin $b' \in \{0,1\}$ and guess EXP(b'). The advantage ADV_λ^{cPRF} of $\mathcal{B}$ in the PRF security game is at least $\frac{1}{3p_\delta(\lambda)}$ for all sufficiently large λ (see Analysis for details), directly violating the security of $\mathcal{F}$.

Analysis: Let $p_b \triangleq \Pr_{x\in X\setminus X_S}[h(x) \neq Challenge_\lambda(x)|EXP(b)]$ be the probability taken with respect to experiment EXP(b). In EXP(1), $Challenge_\lambda$ is a uniformly random function. Thus, $p_1 = \frac{1}{2}$. With high probability, $\mathcal{B}$ will output a random bit $b' \in \{0,1\}$, guessing correctly with probability 1/2.

In EXP(0), h is an ϵ-approximation to f_k, and thus to $Challenge_\lambda$, with probability at least $1-\delta$. In this case, $p_0 \geq 1-\epsilon \geq \frac{1}{2} + \frac{1}{p_\epsilon(\lambda)}$. By a Hoeffding bound, $\mathcal{B}$ will guess $b' = 0$ with high probability by estimating p using only polynomial in $\lambda, p_\epsilon(\lambda)$ samples. On the other hand, if h is not an ϵ-approximation, $\mathcal{B}$ will $b' = 0$ with probability at least 1/2.

Let $negl(\lambda)$ be the error probability from the Hoeffding bound, which can be made exponentially small in λ. The success probability is: $\Pr[b = b'|b = 0] \geq (1-\delta)(1-negl(\lambda)) + \frac{\delta}{2}$ which, for $1-\delta \geq \frac{1}{p_\delta(\lambda)}$ is at least $\frac{1}{3p_{\delta(\lambda)}} + \frac{1}{2}$ for sufficiently large λ. Thus $\mathcal{B}$ a non-negligible advantage of $1/3p_{\delta(\lambda)}$ in the constrained PRF security game. □

4.3 Learning with related concepts

The idea that some functions or concepts are related to one another is very natural. For a DNF formula, for instance, related concepts may include formulas where a clause has been added or formulas where the roles of two variables are swapped. For a decision tree, we could consider removing some accepting leaves and examining the resulting behavior. We might consider a circuit; related circuits might alter internal gates or fix the values of specific input or internal wires.

Formally, we consider indexed representation classes. As discussed in the preliminaries, general classes of functions are easily represented as a indexed family. For example, we may consider the bit representation of a function (say, a log-depth circuit) as an index into a whole class (NC^1). This formalism enables the study of related concepts by instead considering concepts whose keys are related in some way. The related concept setting shares an important property with the restriction access setting: different representations of the same functions might have very different properties. Exploring the properties of different representations – and perhaps their RC learnability as defined below – is a direction for future work.

In our model of learning with related concepts, we allow the learner to query a membership oracle for the concept $f_k \in \mathcal{C}_\lambda$ and also for some 'related' concepts $f_{\phi(k)} \in \mathcal{C}_\lambda$ for some functions ϕ. The *related-concept deriving (RCD)* function ϕ is restricted to be from a specified class, Φ_λ. For each $\phi \in \Phi_\lambda$, a learner can access the membership oracle for $f_{\phi(k)}$. For example: let $K_\lambda = \{0,1\}^\lambda$ and let

$$\Phi_\lambda^\oplus = \{\phi_\Delta : k \mapsto k \oplus \Delta\}_{\Delta \in \{0,1\}^\lambda} \tag{4}$$

Definition 4.12 (Φ-Related-Concept Learning Model). *For $\mathcal{C}$ a representation class indexed by $\{K_\lambda\}$, let $\Phi = \{\Phi_\lambda\}$, with each $\Phi_\lambda = \{\phi : K_\lambda \to K_\lambda\}$ a set of functions on K_λ containing the identity function id_λ. The related-concept oracle RC_k, on query (ϕ, x), responds with $f_{\phi(k)}(x)$, for all $\phi \in \Phi_\lambda$ and $x \in X_\lambda$.*

An algorithm A is an (ϵ,δ)-Φ-RC learning algorithm for a $\mathcal{C}$ if, for all sufficiently large λ, for every $k \in K_\lambda$, $\Pr[\mathcal{A}^{RK_k(\cdot,\cdot)} = h] \geq 1-\delta$ where h is an ϵ-approximation f_k.

Studying the related-concept learnability of standard representation classes (ex: DNFs and decision trees) under different RCD classes Φ is an interesting direction for future study.

4.3.1 RKA PRFs

Again we look to pseudorandom functions for hard-to-learn concepts. To support the extra power of the related concept model, our PRFs will need to maintain their pseudorandomness even when the PRF adversary has access to the function computed with related keys. Related-key secure PRFs [BC10, ABPP14] provide just this guarantee. As in the definition of RC learning, the security of related-key PRFs is given with respect to a class Φ of related-key deriving functions. As we describe in the remainder of the section, related-key secure PRFs prove hard to weakly Φ-RC learn.

Definition: Security Game

Let $\Phi_\lambda \subseteq Fun(K_\lambda, K_\lambda)$ be a subset of functions on K_λ. The set $\Phi = \{\Phi_\lambda\}$ is called the *Related-Key Deriving (RKD)* class and each function $\phi \in \Phi_\lambda$ is an RKD function.

- C picks a random key $k \in K_\lambda$, a bit $b \in \{0, 1\}$, and exposes the oracle according to EXP(b):

  $\mathrm{RKFn}_\lambda(\phi, x)$: On input $(\phi, x) \in \Phi_\lambda \times X_\lambda$, outputs:

$$\begin{array}{ll} F(\phi(k), x) & \text{in EXP}(0) \\ y \leftarrow Y_\lambda & \text{in EXP}(1) \end{array}.$$

  In EXP(1), the responses to RKFn_λ are selected uniformly at random from the range, with the requirement that the responses be consistent for identical inputs (ϕ, x).

- The adversary interacts with the oracle, and outputs a bit $b' \in \{0, 1\}$.

  Definition 4.13. *The advantage is defined as* $ADV_\lambda^{\Phi\text{-}RKA}(\mathsf{A}) := \Pr[b' = b]$ *in the above security game.*

Definition 4.14 (Φ Related-key attack PRF (Φ-RKA-PRF))**.** *Let* $\mathcal{F} = \{F_\lambda : K_\lambda \times X_\lambda \to Y_\lambda\}$ *be family of functions and let* $\Phi = \{\Phi_\lambda\}$ *with each* $\Phi_\lambda \subseteq Fun(K_\lambda, K_\lambda)$ *be a set of functions on* K_λ. $\mathcal{F}$ *is a* Φ related-key attack PRF family *if for all probabilistic polynomial-time adversaries* A *and for all sufficiently large* λ *and all polynomials* $p(n)$*:*

$$ADV_\lambda^{\Phi\text{-}RKA}(\mathsf{A}) < \frac{1}{2} + \frac{1}{p(n)},$$

over the randomness of C *and* A*.*

4.3.2 Hardness of related concept learning

In the Appendix C, we present a concept that can be RC-learned under $\Phi^\oplus$ (Equation 4), but is hard to weakly learn with access to membership queries. We construct the concept $\mathcal{F}$ from a PRF $\mathcal{G}$ and a PRP P. Informally, the construction works by hardcoding the the PRF key in the function values *on a related PRF*. With the appropriate related-concept access, a learner can learn the PRF key.

We now present a general theorem relating RKA-PRFs to hardness of RC learning. This connection yields hardness for a class $\mathcal{C}$ with respect to restricted classes of relation functions Φ. More general hardness results will require new techniques.

Theorem 4.4. *Let $\mathcal{F}$ be a boolean-valued Φ-RKA-PRF with respect to related-key deriving class Φ and keyspace K. For a representation class $\mathcal{C}$, if $\mathcal{F} \subseteq \mathcal{C}$, then there exists an related-concept deriving class Ψ such that $\mathcal{C}$ is hard to Ψ-RC.*

As a corollary, we get a lower bound coming from the RKA-PRF literature. For a group $(G, +)$, and $K = G^m$, define the the element-wise addition RKD functions as

$$\Phi_{+}^{m} = \{\phi_\Delta : k[1], \ldots, k[m] \mapsto k[1] + \Delta[1], \ldots, k[m] + \Delta[m]\}_{\Delta \in G^m} \tag{5}$$

Notice that Φ_{+}^{m} directly generalizes $\Phi^{\oplus}$ with $G = \mathbb{Z}_2$. For this natural RKD function family, we are able to provide a strong lower bound based on the hardness of DDH and the existence of collision-resistant hash functions using the RKA-PRF constructions from [ABPP14].

Corollary 4.5 (Negative Result from RKA-PRF)**.** *If the DDH assumption holds and collision-resistant hash functions exist NC^1 is hard to Φ_{+}^{m}-RKA-learn.*

Proof of Theorem 4.4. We interpret $\mathcal{F} = \{\mathcal{F}_\lambda\}$ as a representation class. For each λ, the concepts $f_k \in \mathcal{F}_\lambda$ are indexed by K_λ and have domain X_λ. Let $\mathcal{C} = \{\mathcal{C}_\lambda\}$ be a representation class, with domain X_λ and indexing set I_λ. For $i \in I_\lambda$, c_i is a concept in $\mathcal{C}_\lambda$.

By hypothesis, $\mathcal{F} \subseteq \mathcal{C}$: for sufficiently large λ, for all $k \in K_\lambda$ there exists $i \in I_\lambda$ such that $c_i \equiv f_k$. For concreteness, let M_λ be this map from K_λ to I_λ.[19]

We can now specify the RCD class $\Psi_\lambda : I_\lambda \to I_\lambda$. Let $M_\lambda(K_\lambda) \subseteq I_\lambda$ be the image of K_λ under M_λ. We define $\Psi_\lambda = \{\psi_\phi : \phi \in \Phi_\lambda\}$:

$$\psi_\phi(i) = \begin{cases} M_\lambda \circ \phi \circ M_\lambda^{-1}(i) & \text{if } i \in M_\lambda(K_\lambda) \\ i & \text{otherwise.} \end{cases}$$

Reduction: Suppose, for contradiction, that there exists an efficient Ψ-RC learning algorithm $\mathcal{A}$ for $\mathcal{C}$ as in the statement of the theorem. We construct algorithm $\mathcal{B}$ breaking the Φ-RKA-PRF security of $\mathcal{F}$. In the PRF security game, $\mathcal{B}$ is presented with the oracle $RKFn(\cdot, \cdot)$; $\mathcal{A}$ is presented with the oracle $RC(\cdot, \cdot)$. Run $\mathcal{A}$, and answer queries $(\psi_\phi, x) \in \Psi_\lambda \times X_\lambda$ to RC by querying $RKFn$ on (ϕ, x) and passing the response along to $\mathcal{A}$. Let $X_{\mathcal{A}} = \{x \in X_\lambda : \mathcal{A} \text{ queried } (\psi, x) \text{ for some } \psi\}$. Once $\mathcal{A}$ terminates, it outputs hypothesis h. In EXP(0), $RKFn()$ responds according to f_k for some $k \in K\lambda$; in this case, $\mathcal{B}$ simulates the RC oracle for the concept $c_{M(k)}$.

After receiving hypothesis h, $\mathcal{B}$ estimates the probability $\Pr_{x \leftarrow X \setminus X_{\mathcal{A}}}[h(x) = RKFn_\lambda(x)]$. In EXP(0), this probability is at least $1 - \epsilon$ with probability at least $1 - \delta$; in EXP(1), it is exactly 1/2. To sample uniform $x \in X \setminus X_{\mathcal{A}}$, we simply take a uniform $x \in X$: with high probability $x \in X \setminus X_{\mathcal{A}}$. If the estimate is close to ϵ, guess EXP(0); otherwise, flip an fair coin $b' \in \{0, 1\}$ and guess EXP(b'). The advantage $ADV_\lambda^{\Phi\text{-}RKA}$ of $\mathcal{B}$ in the PRF security game is at least $\frac{1}{3p_\delta(n)}$ (see Analysis for details) for all sufficiently large λ, directly violating the security of $\mathcal{F}$.

Analysis: Let $p_b \triangleq \Pr_{x \in X \setminus X_{\mathcal{A}}}[h(x) \neq RKFn(\mathrm{id}_\lambda, x) | EXP(b)]$ be the probability taken with respect to experiment EXP(b). In EXP(1), $RKFn$ is a uniformly random function. Thus, $p_1 = \frac{1}{2}$. With high probability, $\mathcal{B}$ will output a random bit $b' \in \{0, 1\}$, guessing correctly with probability 1/2.

In EXP(0), h is an ϵ-approximation to $RKFn(\mathrm{id}, \cdot)$ with probability at least $1 - \delta$. In this case, $p_0 \geq 1 - \epsilon \geq \frac{1}{2} + \frac{1}{p_\epsilon(\lambda)}$. By a Hoeffding bound, $\mathcal{B}$ will guess $b' = 0$ with high probability by estimating

[19] This is a non-uniform reduction in general, but in most cases, the map M is known. That is, M_λ is the map that takes a key and outputs a circuit computing the function.

p using only polynomial in $\lambda, p_\epsilon(\lambda)$ samples. On the other hand, if h is not an ϵ-approximation, $\mathcal{B}$ will $b' = 0$ with probability at least $1/2$.

Let $negl(\lambda)$ be the error probability from the Hoeffding bound, which can be made exponentially small in λ. The success probability is: $\Pr[b = b'|b = 0] \geq (1-\delta)(1-negl(\lambda)) + \frac{\delta}{2}$ which, for $1-\delta \geq \frac{1}{p_\delta(\lambda)}$ is at least $\frac{1}{3p_{\delta(\lambda)}} + \frac{1}{2}$ for sufficiently large λ. Thus $\mathcal{B}$ a non-negligible advantage of $1/3p_{\delta(\lambda)}$ in the Φ-RKA-PRF security game. ☐

Proof. For $n \in \mathbb{N}$ let $\mathbb{G} = \langle g \rangle$ be a group of prime order $p = p(n)$, $X_n = \{0,1\}^{m(n)} \setminus \{0^n\}$, $K_n = \mathbb{Z}_p^m(n)$, and define $F_k(x)$ as in Theorem 4.5 of [Abdalla] (). Let ϕ_+^m be as above over K. ☐

4.4 Learning with Aggregate Queries

This computational learning model is inspired by our aggregate PRFs. Rather than being a natural model in its own right, this model further illustrates how cryptography and learning are in some senses duals. Here, we consider a new extension to the power of the learning algorithm. Whereas membership queries are of the form "What is the label of an example x?", we grant the learner the power to request the evaluation of simple functions on tuples of examples $(x_1, ..., x_k)$ such as "How many of $(x_1...x_k)$ are in C?" or "Compute the product of the labels of $(x_1, ..., x_k)$?". Clearly, if k is polynomial then this will result only a polynomial gain in the query complexity of a learning algorithm in the best case. Instead, we propose to study cases when k may be super polynomial, but the description of the tuples is succinct. For example, the learning algorithm might query the number of x's in a large interval that are positive examples in the concept.

As with the restriction access and related concept models – and the aggregate PRFs we define in this work – the Aggregate Queries (AQ) learning model will be considered with restrictions to both the types of aggregate functions Γ the learner can query, and the sets $\mathcal{S}$ over which the learner may request these functions to be evaluated on. We now present the AQ learning model informally:

Definition 4.15 (($\Gamma, \mathcal{S}$)-Aggregate Queries (AQ) Learning)**.** *Let $\mathcal{C}$ be a representation class with domains $X = \{X_\lambda\}$, and $\mathcal{S} = \{\mathcal{S}_\lambda\}$ where each $\mathcal{S}_\lambda$ is a collection of efficiently recognizeable subsets of the X_λ. $\Gamma : \{0,1\}^* \to V_\lambda$ be an aggregation function [as in def:]. Let $\mathsf{AGG}_k^\lambda \triangleq \mathsf{AGG}_{f_k, \mathcal{S}_\lambda, \Gamma_\lambda}^\lambda$ be the aggregation oracle for $f_k \in \mathcal{C}_\lambda$, for $S \in \mathcal{S}_\lambda$ and Γ_λ.*

An algorithm $\mathcal{A}$ is an (ϵ, δ)-$(\Gamma, \mathcal{S})$-AQ learning algorithm for $\mathcal{C}$ if, for all sufficiently large λ, for every $f_k \in \mathcal{C}_\lambda$, $\Pr[\mathcal{A}^{MEM_{f_k}(\cdot), \mathsf{AGG}_{f_k}^\lambda(\cdot)} = h] \geq 1 - \delta$ where h is an ϵ-approximation to f_k.

4.4.1 Hardness of aggregate query learning

Theorem 4.6. *Let $\mathcal{F}$ be a boolean-valued aggregate PRF with respect to set system $\mathcal{S} = \{\mathcal{S}_\lambda\}$ and accumulation function $\Gamma = \{\Gamma_\lambda\}$. For a representation class $\mathcal{C}$, if $\mathcal{F} \subseteq \mathcal{C}$, then $\mathcal{C}$ is hard to $(\Gamma, \mathcal{S})$-AQ learn.*

Looking back to our constructions of aggregate pseudorandom function families from the prequel, we have the following corollaries.

Corollary 4.7. *The existence of one-way functions implies that P/poly is hard to $(\sum, \mathcal{S}_{[a,b]})$-AQ learn, with $\mathcal{S}_{[a,b]}$ the set of sub-intervals of the domain as defined in Section 3.*

Corollary 4.8. *The DDH Assumption implies that NC^1 is hard to $(\sum, \mathcal{S}_{[a,b]})$-AQ learn, with $\mathcal{S}_{[a,b]}$ the set of sub-intervals of the domain as defined in Section 3.*

Corollary 4.9. *The subexponential DDH Assumption implies that NC^1 is hard to $(\prod, \mathcal{R})$-AQ learn, with $\mathcal{R}$ the set of read-once boolean formulas defined in Section 3.*

Proof of Theorem 4.6. Interpreting $\mathcal{F}$ itself as a concept class, we will show an efficient reduction from violating the pseudorandomness property of $\mathcal{F}$ to weakly $(\Gamma, \mathcal{S})$-AQ learning $\mathcal{F}$. By assumption, $\mathcal{F} \subseteq \mathcal{C}$, implying that $\mathcal{C}$ is hard to learn as well.

Reduction: Suppose for contradiction that there exists an efficient weak learning algorithm $\mathcal{A}$ for $\mathcal{F}$. We define algorithm $\mathcal{B}$ violating the aggregate PRF security of $\mathcal{F}$. In the PRF security game, $\mathcal{B}$ is presented with two oracles: $F(\cdot)$ and AGG_F^λ for a function F chosen according to the secret bit $b \in \{0, 1\}$. In EXP(0), $F = f_k$ for random $k \in K_\lambda$; by assumption $f_k \in \mathcal{C}_\lambda$. In EXP(1), F is a uniformly random function from X to $\{0, 1\}$. The learning algorithm $\mathcal{A}$ is presented with precisely the same oracles. $\mathcal{B}$ runs $\mathcal{A}$, simulating its oracles by passing queries and responses to its own oracles. $X_\mathcal{A} = \{x \in X_\lambda : \mathcal{A} \text{ queried } (\psi, x) \text{ for some } \psi\}$. Once $\mathcal{A}$ terminates, it outputs hypothesis h.

After receiving hypothesis h, $\mathcal{B}$ estimates the probability

$$p = \Pr_{x \leftarrow X \setminus X_\mathcal{A}}[h(x) = F(x)]$$

(using polynomial in $\lambda, p_\epsilon(\lambda)$ samples). In EXP(0), this probability is at least $1 - \epsilon$ with probability at least $1 - \delta$; in EXP(1), it is exactly 1/2. To sample uniform $x \in X \setminus X_\mathcal{A}$, we simply take a uniform $x \in X$: with high probability $x \in X \setminus X_\mathcal{A}$. If the estimate is close to ϵ, guess EXP(0); otherwise, flip an fair coin $b' \in \{0, 1\}$ and guess EXP(b'). The advantage ADV_λ^{APRF} of $\mathcal{B}$ in the PRF security game is at least $\frac{1}{3p_\delta(n)}$ for all sufficiently large λ (as shown below), directly violating the security of $\mathcal{F}$.

Let

$$p_b \triangleq \Pr_{x \in X \setminus X_\mathcal{A}}[h(x) \neq F(x) | EXP(b)]$$

be the probability taken with respect to experiment EXP(b). In EXP(1), F is a uniformly random function. Thus, $p_1 = \frac{1}{2}$. With high probability, $\mathcal{B}$ will output a random bit $b' \in \{0, 1\}$, guessing correctly with probability 1/2.

In EXP(0), h is an ϵ-approximation to F with probability at least $1 - \delta$. In this case, $p_0 \geq 1 - \epsilon \geq \frac{1}{2} + \frac{1}{p_\epsilon(\lambda)}$. By a Hoeffding bound, $\mathcal{B}$ will guess $b' = 0$ with high probability by estimating p using only polynomial in $\lambda, p_\epsilon(\lambda)$ samples. On the other hand, if h is not an ϵ-approximation, $\mathcal{B}$ will $b' = 0$ with probability at least 1/2.

Let negl(λ) be the error probability from the Hoeffding bound, which can be made exponentially small in λ. The success probability is:

$$\Pr[b = b' | b = 0] \geq (1 - \delta)(1 - \text{negl}(\lambda)) + \frac{\delta}{2}$$

which, for $1 - \delta \geq \frac{1}{p_\delta(\lambda)}$ is at least $\frac{1}{3p_{\delta(\lambda)}} + \frac{1}{2}$ for sufficiently large λ. Thus $\mathcal{B}$ a non-negligible advantage of $1/3p_{\delta(\lambda)}$ in the $(\Gamma, \mathcal{S})$-aggregate-PRF security game. ☐

4.4.2 Acknowledgements

Aloni Cohen's research was supported in part by NSF Graduate Student Fellowship and NSF grants CNS1347364, CNS1413920 and FA875011-20225.

Shafi Goldwasser's research was supported in part by NSF Grants CNS1347364, CNS1413920 and FA875011-20225.

Vinod Vaikuntanathan's research was supported by DARPA Grant number FA8750-11-2-0225, an Alfred P. Sloan Research Fellowship, an NSF CAREER Award CNS-1350619, NSF Frontier Grant CNS-1414119, a Microsoft Faculty Fellowship, and a Steven and Renee Finn Career Development Chair from MIT.

References

[ABPP14] Michel Abdalla, Fabrice Benhamouda, Alain Passelègue, and Kenneth G. Paterson. Related-key security for pseudorandom functions beyond the linear barrier. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 77–94. Springer, 2014.

[BC10] Mihir Bellare and David Cash. Pseudorandom functions and permutations provably secure against related-key attacks. In Tal Rabin, editor, *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, volume 6223 of *Lecture Notes in Computer Science*, pages 666–684. Springer, 2010.

[BF02] Nader H Bshouty and Vitaly Feldman. On using extended statistical queries to avoid membership queries. *The Journal of Machine Learning Research*, 2:359–395, 2002.

[BGI14a] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In Krawczyk [Kra14], pages 501–519.

[BGI14b] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In Krawczyk [Kra14], pages 501–519.

[BGV11] Siavosh Benabbas, Rosario Gennaro, and Yevgeniy Vahlis. Verifiable delegation of computation over large datasets. In Phillip Rogaway, editor, *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, volume 6841 of *Lecture Notes in Computer Science*, pages 111–131. Springer, 2011.

[BK03] Mihir Bellare and Tadayoshi Kohno. A theoretical treatment of related-key attacks: Rka-prps, rka-prfs, and applications. In *Advances in CryptologyEUROCRYPT 2003*, pages 491–506. Springer, 2003.

[BLMR13] Dan Boneh, Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. Key homomorphic prfs and their applications. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 410–428. Springer, 2013.

[BW13a] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In Sako and Sarkar [SS13], pages 280–300.

[BW13b] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In Sako and Sarkar [SS13], pages 280–300.

[DRWY12] Zeev Dvir, Anup Rao, Avi Wigderson, and Amir Yehudayoff. Restriction access. In Shafi Goldwasser, editor, *Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012*, pages 19–33. ACM, 2012.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986. Extended abstract in FOCS 84.

[GGN10] Oded Goldreich, Shafi Goldwasser, and Asaf Nussboim. On the implementation of huge random objects. *SIAM J. Comput.*, 39(7):2761–2822, 2010.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

[KPTZ13a] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In Sadeghi et al. [SGY13], pages 669–684.

[KPTZ13b] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In Sadeghi et al. [SGY13], pages 669–684.

[Kra14] Hugo Krawczyk, editor. *Public-Key Cryptography - PKC 2014 - 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26-28, 2014. Proceedings*, volume 8383 of *Lecture Notes in Computer Science*. Springer, 2014.

[KV94] Michael J. Kearns and Leslie G. Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *J. ACM*, 41(1):67–95, 1994.

[NR04] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudorandom functions. *J. ACM*, 51(2):231–262, 2004.

[RR97] Alexander A. Razborov and Steven Rudich. Natural proofs. *J. Comput. Syst. Sci.*, 55(1):24–35, 1997.

[SGY13] Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors. *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*. ACM, 2013.

[SS13] Kazue Sako and Palash Sarkar, editors. *Advances in Cryptology - ASIACRYPT 2013 - 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part II*, volume 8270 of *Lecture Notes in Computer Science*. Springer, 2013.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 475–484. ACM, 2014.

[Val84] Leslie G Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[VV86] Leslie G. Valiant and Vijay V. Vazirani. NP is as easy as detecting unique solutions. *Theor. Comput. Sci.*, 47(3):85–93, 1986.

A Simple Positive Results

In the following, we present examples of concept classes separating the Related Concept and Aggregate Query learning models from learning with Membership Queries. We emphasize that the learnability of many traditional concept classes in these models has not been studied, and more general positive results may exist. In order to exhibit separations, we present generic, contrived constructions from simple cryptographic primitives to exhibit our separations. In each case, a MQ learner cannot succeed better than a trivial algorithm, while the stronger model manages to exactly, and properly learn the function.

A.1 Related-concept

While some existing pseudorandom functions are known to suffer from related-key attacks [BK03], these vulnerabilities do not seem directly useful for a proper learning algorithm. Instead we construct a family of PRFs for which the secret key can be recovered under related-key attacks.

We demonstrate a concept that can be RC-learned under additive Φ (defined below), but is hard to weakly learn with access to membership queries. We construct the concept $\mathcal{F}$ from a PRF $\mathcal{G}$ and a PRP P. Informally, the construction works by hardcoding the the PRF key in the function values *under a related PRF key.* With the appropriate related-key access, a learner can learn the PRF key.

Let $\mathcal{G} = \{G_k : \mathbb{Z}_{2^\lambda} \to \{0,1\}\}_{k \in K}$ be a PRF with keyspace $K = \{0,1\}^\lambda$ and let $P = \{\pi : K \to K\}$ be a pseudorandom permutation family on K. For each $g_k \in \mathcal{G}_K$ and $\pi \in P$, we define the following function:

$$F_{k,\pi} = \begin{cases} x\text{th bit of } (\pi(k) \oplus k) & \text{if } x \in [0, \lambda-1] \\ (x-\lambda)\text{th bit of } \pi^{-1}(k) & \text{if } x \in [\lambda, 2\lambda-1] \\ g_k(x) & \text{otherwise} \end{cases}$$

Let $\mathcal{F} = \{F_{k,\pi} : k \in K, \pi \in P\}$. We interpret $\mathcal{F}$ as a keyed concept with elements indexed by a pairs (k, π).

We need to choose a RKD class Φ that will enable recovery of the PRF key k by accessing the PRF for key $\pi(k) \oplus k$. We choose $\Phi = \Phi^{\oplus}$ from Section 4.3.1:

$$\Phi^{\oplus} = \{\phi_\Delta : k \mapsto k \oplus \Delta\}_{\Delta \in K}$$

Note that in that section, we prove a negative result for a strictly stronger RC adversary, but with a different concept class.

Theorem A.1 (Separating RC and MQ)**.** *The keyed concept $\mathcal{F}$ defined above can be (efficiently) exactly $\Phi^{\oplus}$-RC-learned, but is hard to even weakly MQ learn efficiently.*

Proof. Let $F_{k,\pi} \in \mathcal{F}_n$.

$\Phi^{\oplus}$-RC Learning: Let $RC_{k,\pi}$ be the related-concept oracle, taking queries $(\phi, x) \in \Phi^{\oplus} \times \mathbb{Z}_{2^\lambda}$ and returning $F_{\phi(k),\pi}(x)$. Define $\Delta \in K$ such that $\Delta[i] = F_{k,x}(i)$ for all $i \in [\lambda - 1]$; compute the ith bit by querying the oracle at (id, i), where $\mathrm{id} = 0^\lambda$ is the identity function. By construction, $k \oplus \Delta = \pi(k)$. Let $k' \in K$ such that $k'[i] = F_{\pi(k),i+\lambda}$ for all $i \in [\lambda - 1]$; we find bit $k'[i]$ by querying $(\phi_\Delta, i + \lambda)$. By construction, $k' = \pi^{-1}(\pi(k)) = k$. Given the PRF key k, we may compute $F_{k,\pi}$ on all inputs in $X \setminus [2\lambda - 1]$; simply querying those remaining points yields an exact characterization of $F_{k,\pi}$.

MQ Learning: (Informally) Given a weakly-MQ learning algorithm $\mathcal{A}$ for $\mathcal{F}$, an algorithm $\mathcal{B}$ violating the security of the pseudorandom function can be constructed. By assumption, $\mathcal{A}$ is an (ϵ, δ)-MQ learning algorithm with ϵ and $1 - \delta$ both non-negligible in n. First, observe that $\mathcal{A}$ is an (ϵ', δ')-MQ-learning algorithm for the following concept class, indexed by $k \in K$ and uniformly random $r_1 \in \{0, 1\}^\lambda$, with $\epsilon' \geq \epsilon - negl(\lambda)$ and $\delta' \geq \delta - negl(\lambda)$:

$$F^1_{k,r_1} = \begin{cases} x\text{th bit of } r_1 & \text{if } x \in [0, \lambda - 1] \\ (x - \lambda)\text{th bit of } \pi^{-1}(k) & \text{if } x \in [\lambda, 2\lambda - 1] \\ g_k(x) & \text{otherwise} \end{cases}$$

Otherwise, the quality of the hypothesis output by $\mathcal{A}$ would be noticeably different for random functions $F_{k,\pi}$ and F_{k,r_1}. By the security of the pseudorandom permutation, $\pi(k) \oplus k$ should be indistinguishable from uniformly random r_1; this difference could be used to violate the security of the pseudorandom permutation π.

A similar argument will show that $\mathcal{A}$ is an (ϵ'', δ'')-MQ-learning algorithm for the following concept class, indexed by $k \in K$ and $r_1, r_2 \in \{0, 1\}^\lambda$, with $\epsilon'' \geq \epsilon' - negl(\lambda)$ and $\delta'' \geq \delta' - negl(\lambda)$:

$$F^2_{k,r_1} = \begin{cases} x\text{th bit of } r_1 & \text{if } x \in [0, \lambda - 1] \\ (x - \lambda)\text{th bit of } r_2 & \text{if } x \in [\lambda, 2\lambda - 1] \\ g_k(x) & \text{otherwise} \end{cases}$$

Furthermore, weak learning of this concept requires weak learning of this concept even when restricting the domain to require $x \notin [0, 2\lambda - 1]$.

This last oracle can be simulated by $\mathcal{B}$ with only oracle access to a random PRF $g_k \in G_\lambda$. That this concept is weakly learnable violates the security of the PRF G in the usual way. □

A.2 Aggregate queries

We turn to a positive result for learning in the AQ model. Our starting point is the intuition that with aggregate queries, it is easy to distinguish a point function from an everywhere-zero function.

Formally, consider the case when $D = \mathbb{Z}_{2^\lambda}$, $R = \{0, 1\}$, $\Gamma = \sum$ is summation modulo 2, and $\mathcal{S}_{[a,b]} = \{[a, b] : a, b \in \mathbb{Z}_{2^\lambda}; a \leq b\}$ the set of intervals on $\mathbb{Z}_\lambda$. By AQ-learning with respect to *summation over intervals*, we mean $(\sum, \mathcal{S}_{[a,b]})$-AQ learning. Let the concept class $\mathcal{D}_\lambda$ of point functions be defined:

$$\mathcal{D}_\lambda := \{\delta_y : y \in \mathbb{Z}_{2^\lambda}\}$$

where each δ_y is nonzero only at y.

Lemma A.2 (Point functions)**.** *The concept class of point functions $\mathcal{D}_\lambda$ is efficiently, exactly, and properly $(\sum, \mathcal{S}_{[a,b]})$-AQ-learnable.*

Proof. Observe that for $\delta_y \in \mathcal{D}_\lambda$ and interval $[a,b] \subseteq \mathbb{Z}_{2^\lambda}$: $AGG_{\sum,\delta_y}([a,b]) = 1 \iff y \in [a,b]$. This allows us to perform binary search over the domain and find y with at most λ queries to the $AGG_{\sum,\delta_y}(\cdot)$ oracle. $\square$

But if we don't require exact-learning, point functions are trivially learnable with no queries at all; indeed, the hypothesis $h(x) = 0$ agrees with $\delta_y(x)$ at all but a single point! But $\mathcal{D}_\lambda$ is not *exactly* MQ-learnable. More importantly, for two uniformly selected concepts $\delta_y, \delta_w \leftarrow \mathcal{D}_\lambda$, MQ cannot distinguish membership oracle access to δ_y and δ_w. We will leverage this to construct a much stronger separation.

Let $\mathcal{G}_\lambda = \{g_k : \{0,1\}^{\lambda-1} \to \{0,1\}\}_{k \in \{0,1\}^{\lambda-1}}$ be a pseudorandom function family with $(\lambda-1)$-bit keys k and inputs x.

Functions in our concept class $f_k \in \mathcal{F}_\lambda$ will be indexed by an $(\lambda-1)$-bit key, but take inputs from $\{0,1\}^\lambda$. On half the domain, f_k behaves as the PRF g_k, while on the other half it behaves as the point function δ_k. Letting $x[2:\lambda] = (x[2], \ldots, x[\lambda])$:

$$f_k(x) = \begin{cases} \delta_k(x[2:\lambda]) & \text{if } x[0] = 0 \\ G_k(x[2:\lambda]) & \text{if } x[1] = 1 \end{cases}$$

Theorem A.3 (Separating AQ from MQ)**.** *The concept class $\mathcal{F}$ is exactly and (properly) AQ-learnable with respect to summation over intervals. For any polynomials $p_\epsilon(\lambda), p_\delta(\lambda)$, this concept class is hard to (ϵ,δ)-MQ learn for $\epsilon \le \frac{1}{4} - \frac{1}{p_\epsilon(\lambda)}$ and $1-\delta \ge \frac{1}{p_\delta(\lambda)}$.*

Note that it while it easy to $(1/4, 1/4)$-MQ learn $\mathcal{C}$ (for example, outputting the constant 0 function), the theorem above claims that we cannot do appreciably better in ϵ with non-negligible probability $1-\delta$. This has the flavor of a 'hardness of weakly learning' theorem.

Proof. For $\lambda \in \mathbb{N}$, let $f_k \in \mathcal{F}_\lambda$. The first part of the theorem follows as a corollary to the previous lemma. After exactly learning δ_k by binary search, the function f_k is uniquely specified by k.

For the second part, we reduce to the hardness of MQ learning the pseudorandom function, g_k. Suppose for contradiction that there exists an algorithm $\mathcal{A}$ that, when given access to an oracle $\mathcal{O} = g_k(\cdot)$, with probability at least $\frac{1}{p_\delta(\lambda)}$, outputs hypothesis $h : \{0,1\}^\lambda \to \{0,1\}$ with $\Pr_{x \leftarrow \{0,1\}^\lambda}[h(x) = f_k(x)] \ge \frac{3}{4} + \frac{1}{p_\epsilon}$. We describe $\mathcal{B}$ – a weak MQ-learning algorithm for the concept $\mathcal{G} = \{g_k\}_{k \in K}$. Given access to oracles $\mathcal{O}_\delta = \delta_k(\cdot)$ and $\mathcal{O}_G = g_k(\cdot)$, $\mathcal{B}$ can exactly simulate oracle access to $\mathcal{O}$ and thus output hypothesis h with the same distribution. But with only $t(\lambda)$-many queries for any t, the probability (over the random choice of k) of querying a non-zero point in $\mathcal{O}_\delta$ is at most $t(\lambda)/2^{\lambda-1}$; thus, with high probability, all queries to $\mathcal{O}_\delta$ will be zero. Therefore it is computationally infeasible to distinguish between the pair of oracles $(\mathcal{O}_\delta, \mathcal{O}_G)$ and $(\mathcal{O}_0, \mathcal{O}_G)$, where $\mathcal{O}_0$ is the constant zero oracle.

If $\mathcal{B}$ answers $\mathcal{A}$'s oracle queries with $(\mathcal{O}_0, \mathcal{O}_G)$ instead of $(\mathcal{O}_{\delta_k}, \mathcal{O}_G)$, $\mathcal{A}$ will successfully output h which ϵ' approximates f_k with probability $1-\delta'$. By the indistinguishability argument, $\epsilon' \ge \epsilon - \mathrm{negl}(\lambda) \ge \epsilon/2$ and $1-\delta' \ge 1-\delta-\mathrm{negl}(\lambda) \ge 1-\delta/2$.

Let $h|_b$ be the restriction of h to the set $\{x : x[1] = b\}$ for $b \in \{0,1\}$.

$$\Pr_x[h(x) \neq f_k(x)] = \frac{1}{2}(\Pr_x[h|_0(x) = 0] + \Pr_x[h|_1(x) = g_k(x[2:n])]) \ge \frac{3}{4} + \frac{1}{2p_\epsilon}$$

$$\implies \Pr_x[h|_1(x) = g_k(x[2:n])] \geq \frac{1}{2} + \frac{1}{p_\epsilon(n)}.$$

Outputting $h|_1$, $\mathcal{B}$ manages to weakly MQ learn the concept $\mathcal{G}_\lambda$. That this concept is weakly learnable violates the security of the PRF G in the usual way. □

ECCC ISSN 1433-8092

http://eccc.hpi-web.de