When Won't Membership Queries Help?

(Extended Abstract)

*Dana Angluin** Yale University

Michael Kharitonov † Stanford University

Abstract

We investigate cryptographic limitations on the power of membership queries to help with concept learning.

In particular, we use the recent construction of a public-key encryption system secure against chosen cyphertext attack by Naor and Yung [19] (and refinements of it) together with the techniques of Kearns and Valiant [16] to show that assuming the intractability of (1) quadratic residues modulo a composite, (2) inverting RSA encryption, or (3) factoring Blum integers, there is no polynomial time prediction algorithm with membership queries for boolean formulas, constant depth threshold circuits, 3μ-boolean formulas, finite unions or intersections of DFAs, 2-way DFAs, NFAs, or CFGs.

Also, we show that if there exist one-way functions that cannot be inverted by polynomial-sized circuits, then Naor and Yung's [18] and Rompel's [21] construction of a signature scheme can be used to show that CNF or DNF formulas are either bounded polynomial time predictable without membership queries, or are not polynomial time predictable even with membership queries; so, in effect, membership queries won't help with predicting CNF or DNF formulas.

*Supported by NSF Grants IRI-8718975 and CCR-9014943. Address: Computer Science Department, Yale University, P. O. Box 2158, New Haven, CT 06520. E-mail: angluin@cs.yale.edu.

†Supported by a Fannie and John Hertz Fellowship and in part by NSF PYI Grant CCR-8858097 with matching funds provided by AT&T and Digital Equipment Corporation, and a grant from the 3M Corporation. Address: Computer Science Department, Stanford University, Stanford, CA 94305. E-mail: misha@cs.stanford.edu.

1 Introduction

We consider the problem of learning a concept from examples. In particular, we consider the task of predicting the classification of a new example, given access to a large collection of correctly classified examples. In the distribution-free paradigm, the examples are all chosen independently according to a fixed but unknown probability distribution, and the goal is to predict the new example correctly with high probability. This model of learning is "passive" in the sense that the learner has no control over the selection of examples.

One can also consider a more "active" version of this setting, in which the learner is allowed to ask about particular examples, that is, the learner makes *membership queries*. We assume the membership queries occur before the new example to predict is given to the learner. This capability appears to increase the power of polynomial time bounded prediction algorithms.

For example, there is a polynomial time algorithm to predict deterministic finite acceptors (DFAs) if membership queries are available [1], but Kearns and Valiant have shown that predicting DFAs without membership queries is as hard as computing certain apparently hard cryptographic predicates [16]. The same situation holds of μ-formulas [3, 15, 16]. There is a polynomial time prediction algorithm for propositional Horn sentences with membership queries [2], but without membership queries they are no easier to predict than general CNF formulas [15], a problem whose status remains open.

Despite these positive results for prediction with membership queries, several important concept classes have so far resisted attack using membership queries, for example, CNF and DNF formulas, general boolean formulas, nondeterministic finite acceptors (NFAs) and context-free grammars (CFGs). The general question we address is: when can we expect membership queries to help in a prediction task?

Valiant [23] used pseudorandom functions [10] to show that general boolean circuits are unpredictable

even with membership queries under the assumption that one-way functions exist. Our results further illuminate the relationship between computational learning theory and cryptography.

First, we show that if there is a public key encryption system secure against chosen cyphertext attack whose decryption function can be polynomially represented as concepts in a given class, then that class of concepts cannot be predicted with membership queries in polynomial time. We use the public key encryption scheme of Naor and Yung [19] and refinements of it to show that the cryptographic assumptions of Kearns and Valiant also imply that there is no polynomial time prediction algorithm with membership queries for boolean formulas, constant-depth threshold circuits, 3μ-boolean formulas, finite unions or intersections of DFAs, 2-way DFAs, NFAs, or CFGs.

Second, we show that if there exists a signature scheme secure against existential forgery under nonuniform adaptive chosen message attack then membership queries will not help with the prediction of any class of concepts that is closed under intersection and in which the set of accepting computations of the signature verifier can be expressed. In particular, the class of CNF formulas satisfies these conditions, and we give cryptographic evidence that CNF and DNF formulas may be as hard to predict with membership queries as they are without.

2 Preliminaries

2.1 Representations of concepts

Let X denote $\{0,1\}^*$; binary strings will represent both examples and concept names. If x is a string, $|x|$ denotes its length. For any natural number n, $X^{[n]} = \{x \in X : |x| \leq n\}$.

A *representation of concepts* $\mathcal{C}$ is any subset of $X \times X$. We interpret an element $\langle u, x\rangle$ of $X \times X$ as consisting of a *concept name* u and an *example* x. The example x is a member of the concept u if and only if $\langle u, x\rangle \in \mathcal{C}$.

Define the *concept represented by* u as

$$\kappa_{\mathcal{C}}(u) = \{x : \langle u, x\rangle \in \mathcal{C}\}.$$

The *set of concepts represented by* $\mathcal{C}$ is

$$\{\kappa_{\mathcal{C}}(u) : u \in X\}.$$

If a set A of binary strings is represented by $\mathcal{C}$, then we define $size_{\mathcal{C}}(A)$ to be the length of the shortest string u such that $\kappa_{\mathcal{C}}(u) = A$.

To represent DFAs we fix a straightforward binary encoding of DFAs with input alphabet $\{0,1\}$ and define $\mathcal{C}_{DFA}$ as the set of pairs $\langle u, x\rangle$ such that u encodes a DFA M and M accepts x. The set of concepts represented is the regular sets over $\{0,1\}$. For each such set A, there is a polynomially bounded relation between $size_{\mathcal{C}_{DFA}}(A)$ and the number of states in the smallest deterministic finite acceptor for A.

2-way DFAs, NFAs and CFGs are represented (analogously to DFAs) by $\mathcal{C}_{2DFA}$, $\mathcal{C}_{NFA}$ and $\mathcal{C}_{CFG}$. Finite unions of DFAs are represented by $\mathcal{C}_{\cup DFA}$ as follows. In a pair $\langle u, x\rangle$, the string u is interpreted as specifying a finite set $M_1, \ldots M_r$ of DFAs, and the string x is in the concept represented by u if and only if at least one M_i accepts x. This representation can be exponentially more succinct than $\mathcal{C}_{DFA}$ in representing regular sets. Finite intersections of DFAs are represented by $\mathcal{C}_{\cap DFA}$.

We fix a straightforward binary representation of general boolean formulas over the variables $X_1, X_2, \ldots$ and the basis AND, OR, and NOT. Then $\langle u, x\rangle$ is an element of $\mathcal{C}_{BF}$ if and only if u represents a positive integer n and a boolean formula ϕ over the variables $X_1, \ldots, X_n$ such that $|x| = n$ and the assignment $X_i = x_i$ for $i = 1, \ldots, n$ satisfies the formula ϕ.

The class $\mathcal{C}_{CNF}$ consists of all those elements $\langle u, x\rangle$ of $\mathcal{C}_{BF}$ such that the formula represented by u is in conjunctive normal form (CNF); $\mathcal{C}_{DNF}$ represents DNF formulas; $\mathcal{C}_{HCNF}$ represents propositional Horn sentences. For $k \geq 1$, $\mathcal{C}_{k\mu}$ consists of all those elements $\langle u, x\rangle$ of $\mathcal{C}_{BF}$ such that u represents a formula with at most k occurrences of each variable. $\mathcal{C}_{1\mu}$ represents the read-once formulas (equivalently, μ-formulas.)

The class $\mathcal{C}_{TC^0,d}$ of threshold circuits of depth d is specified analogously to boolean formulas: $\langle u, x\rangle$ is an element of $\mathcal{C}_{TC^0,d}$ if and only if u represents a positive integer n and a boolean threshold circuit C of depth at most d over the inputs $X_1, \ldots, X_n$, and $|x| = n$ and the assignment $X_i = x_i$ for $i = 1, \ldots, n$ causes the output of C to be 1.

Since each representation of concepts is a set of pairs of binary strings, we can investigate its computational complexity, that is, the computational complexity of the *evaluation problem* for the class of concepts. Note that $\mathcal{C}_{NFA}$ is in NSPACE($\log n$), $\mathcal{C}_{CFG}$ is in log-CFL, and the other representations of concepts defined above are in DSPACE($\log n$).

2.2 Prediction with membership queries

We generalize the definitions of Pitt and Warmuth of prediction algorithms [20] to allow membership queries. A *prediction with membership queries algorithm*, or *pwm-algorithm*, is a possibly randomized algorithm A that takes as input a bound s on the size of the target concept, a bound n on the length of examples, and an accuracy bound ϵ. It may make three different kinds of oracle calls, the responses to which are determined by the unknown target concept c and the unknown distribution D on X, as follows.

1. A membership query takes a string x as input and returns 1 if $x \in c$ and 0 otherwise.

2. A request for a random classified example takes no input and returns a pair $\langle x, b \rangle$ where x is a string chosen independently according to D and $b = 1$ if $x \in c$ and $b = 0$ otherwise, and

3. A request for an element to predict takes no input and returns a string x chosen independently according to D.

A may make any number of membership queries or requests for random classified examples. However, A must eventually make one and only one request for an element to predict, and then eventually halt with an output of 1 or 0 without making any further oracle calls. The output is interpreted as A's guess of how the target concept classifies the element returned by the request for an element to predict. *A runs in polynomial time* if its running time (counting one step per oracle call) is bounded by a polynomial in s, n, $1/\epsilon$.

We say that A *successfully predicts* a representation of concepts $\mathcal{C}$ if and only if for all positive integers s and n, for all positive rationals ϵ, for all concept names $u \in X^{[s]}$, for all probability distributions D on $X^{[n]}$, when A is run with inputs s, n, and ϵ, and oracles determined by $c = \kappa_{\mathcal{C}}(u)$ and D, the probability is at most ϵ that the output of A is not equal to the correct classification of x by $\kappa_{\mathcal{C}}(u)$, where x is the string returned by the (unique) request for an element to predict.

A representation of concepts $\mathcal{C}$ is *polynomially predictable with membership queries* if and only if there is a *pwm*-algorithm A that runs in polynomial time and successfully predicts $\mathcal{C}$. If a representation of concepts is learnable in polynomial time with membership and equivalence queries then it is polynomially predictable with membership queries, thus, $\mathcal{C}_{DFA}$, $\mathcal{C}_{1\mu}$, and $\mathcal{C}_{HCNF}$ are polynomially predictable with membership queries [1, 2, 3].

By the results of Schapire showing that weak learning implies strong learning even with access to membership queries [22], we may without loss of generality replace the arbitrary ϵ in the definition of the success of a *pwm*-algorithm with the fixed value $\epsilon = 1/4$.

3 A reducibility for prediction with membership queries

We define a notion of reducibility analogous to the prediction-preserving reducibility of Pitt and Warmuth [20], but allowing membership queries. We start with two representations of concepts, $\mathcal{C}$ and $\mathcal{C}'$, and define a reduction that will allow us to convert a polynomial time *pwm*-algorithm A' for $\mathcal{C}'$ into a polynomial time *pwm*-algorithm A for $\mathcal{C}$.

There are three mappings involved in this reduction: a mapping g of concept names in $\mathcal{C}$ to concept names in $\mathcal{C}'$, a mapping f of elements x for A to predict to elements x' for A' to predict, and a mapping h of elements x' queried by A' to elements x to be queried by A. In the last case, we also allow h to answer membership queries directly. These will be specified formally in the definition below; first we give some intuition.

The prediction-preserving reductions of Pitt and Warmuth consist of maps g and f; it is instructive to see what the map h is. The algorithm A running on concept $c \in \mathcal{C}$ will be a simulation of A' running on the image c' of c under the map g. When A' requests a classified example, A requests a classified example and receives some $\langle x, b \rangle$. A then supplies $\langle f(x), b \rangle$ to A' to continue the simulation. This is a correct classification of the example $f(x)$ with respect to c' because $x \in c$ if and only if $f(x) \in c'$. Similarly, if A' requests an element to predict, A requests an element to predict, and receives some x. A then supplies $f(x)$ to A' as the element to predict and returns the prediction of A' as its own. If the prediction of A' for $f(x)$ is correct for c', then the same prediction for x is correct for c.

Now consider the situation when A' makes a membership query with a string y. If y is in the image of f, that is, $y = f(x)$ for some string x, then things are rosy – all A needs to do is make a membership query for x and return the answer to A', since in this case $y \in c'$ if and only if $x \in c$. However, typically the image of f is not all strings, and A' is free to ask membership queries for strings y not in the image of f – in order for the simulation to continue correctly, such queries must be answered correctly.

What we require in this case is that the correct answer be computable in polynomial time from y independent of the concept being predicted. All we have actually used in the specific reductions in this paper is a constant 1 or constant 0 answer for all such y. Note that this puts an implicit requirement on the expressive power of the image concept $c' \in \mathcal{C}'$. In particular, c' must be able to exclude (or include) all the strings y that are not in the image of f, so that the answer of 0 (or 1) for all such strings is correct. It is this requirement that distinguishes, for example, between intersections of DFAs and DFAs – size polynomial in n is sufficient to accept $\{w^{n^k} : |w| = n\}$ for intersections of DFAs but not for DFAs, and the element map f for Pitt and Warmuth's reduction of DSPACE($\log n$) to DFAs has an image of this form. The formal definitions follow.

Definition 1 *Let $\mathcal{C}$ and $\mathcal{C}'$ be representations of concepts. Let $\top$ and $\bot$ be elements not in X. Then $\mathcal{C}$ is* pwm-reducible *to $\mathcal{C}'$, denoted $\mathcal{C} \leq_{pwm} \mathcal{C}'$, if and only if there exist three mappings g, f, and h with the following properties.*

1. *There is a nondecreasing polynomial $q(s, n)$ such*

that for all natural numbers s and n and for all $u \in X^{[s]}$, $g(s,n,u)$ is a string u' of length at most $q(s,n)$.

2. *For all natural numbers s and n, for every $u \in X^{[s]}$, and for every $x \in X^{[n]}$, $f(s,n,x)$ is a string x' and $x \in \kappa_{\mathcal{C}}(u)$ if and only if $x' \in \kappa_{\mathcal{C}'}(g(s,n,u))$. Moreover, f is computable in time bounded by a polynomial in s, n, and $|x|$.*

3. *For all natural numbers s and n, for every $u \in X^{[s]}$, and every $x' \in X$, $h(s,n,x')$ is either $\top$, $\bot$, or a string x, and if $h(s,n,x') = \top$ then $x' \in \kappa_{\mathcal{C}'}(g(s,n,u))$, if $h(s,n,x') = \bot$ then $x' \notin \kappa_{\mathcal{C}'}(g(s,n,u))$, and otherwise $x' \in \kappa_{\mathcal{C}'}(g(s,n,u))$ if and only if $x \in \kappa_{\mathcal{C}}(u)$. Moreover, h is computable in time bounded by a polynomial in s, n, and $|x'|$.*

Lemma 2 *The reducibility $\leq_{pwm}$ is transitive. Let $\mathcal{C}$ and $\mathcal{C}'$ be representations of concepts. If $\mathcal{C} \leq_{pwm} \mathcal{C}'$ and $\mathcal{C}'$ is polynomially predictable with membership queries, then $\mathcal{C}$ is also polynomially predictable with membership queries.*

A representation of concepts $\mathcal{C}$ is *$\leq_{pwm}$-complete* in a complexity class $\mathcal{S}$ if and only if $\mathcal{C} \in \mathcal{S}$ and and for every representation of concepts $\mathcal{C}'$ in $\mathcal{S}$, $\mathcal{C}' \leq_{pwm} \mathcal{C}$. Then we have the following results:

Theorem 3
- $\mathcal{C}_{BF} \leq_{pwm} \mathcal{C}_{3\mu}$,
- $\mathcal{C}_{\cup DFA}$, $\mathcal{C}_{\cap DFA}$, *and* $\mathcal{C}_{2DFA}$ *are $\leq_{pwm}$-complete for* $DSPACE(\log n)$,
- $\mathcal{C}_{NFA}$ *is $\leq_{pwm}$-complete for* $NSPACE(\log n)$, *and*
- $\mathcal{C}_{CFG}$ *is $\leq_{pwm}$-complete for* log-CFL.

Since $\mathcal{C}_{BF}$ is in DSPACE($\log n$), these results imply that if $\mathcal{C}_{BF}$ is not polynomially predictable with membership queries, then the classes $\mathcal{C}_{3\mu}$, $\mathcal{C}_{\cup DFA}$, $\mathcal{C}_{\cap DFA}$, $\mathcal{C}_{2DFA}$, $\mathcal{C}_{NFA}$, and $\mathcal{C}_{CFG}$ are also not polynomially predictable with membership queries.

(In Section 7 we give a sketch of the reduction $\mathcal{C}_{BF} \leq_{pwm} \mathcal{C}_{3\mu}$; the others are omitted.)

4 Membership queries and chosen cyphertext attack

In a public-key encryption system, a user A publishes a public key for a (probabilistic) encryption function E and keeps secret the key for the decryption function D. For simplicity we assume that messages are single bits.

In a chosen-cyphertext attack, the attacker has access to an oracle for the function D and can query it repeatedly as part of the attempt to compromise the security of the system. After gathering information, the attacker is presented with an encryption $E(b)$ of a randomly chosen bit $b \in \{0,1\}$. The attacker's goal is to guess the value of b, without further access to the oracle for D. The attack is said to be successful if the probability of the attacker's correctly guessing b is larger than 1/2 by a "polynomially useful" amount.

Suppose instead we think of the decryption function D as representing a concept, namely, all those strings that decrypt to 1. Then the attacker's goal may be thought of as predicting the value of this concept for a string chosen by encrypting a random bit b. The attacker can generate "solved problems" from this distribution because the probabilistic encryption function E is public. In this setting, the ability to query an oracle for D is simply the availability of membership queries for the concept D. Hence the security of a public-key encryption system against chosen cyphertext attack implies the impossibility of a polynomial time algorithm for predicting the associated representation of concepts with membership queries.

Then the problem becomes: how weak a representation system is sufficient to express (sufficiently concisely) the concepts corresponding to decryption functions? Here we can apply the techniques of Kearns and Valiant [16] to reduce the complexity of a decryption function in a way that does not compromise its cryptographic security. The formal definitions follow.

4.1 Chosen cyphertext security

We use essentially the definition given by Naor and Yung for public-key encryption [19], specialized to single-bit messages.

Definition 4 *A* public-key cryptosystem *consists of three probabilistic Turing machines: a* key generator G, *an* encryption mechanism E, *and a* decryption mechanism D. *Each of these machines halts in expected time bounded by a polynomial in the lengths of its inputs. Also, the length of the output of each machine is always bounded by a polynomial in the length of its input. The machines have the following inputs and outputs:*

1. *G takes as input a positive integer n represented in unary and outputs a pair of strings (PK, SK).*
2. *E takes as input a pair (b, PK), where b is a bit and PK is a string, and outputs a string x.*
3. *D takes as input a triple of strings (PK, SK, x) and outputs a bit b.*

Here n is the security parameter, *PK is an* encryption key, *SK is a* decryption key, *b is a (single bit)* message, *and x is a* cyphertext. *We require that decryption of properly encrypted bits be correct. That is, for all pairs (PK, SK) output by G and for all $b \in \{0,1\}$, if*

E outputs x on inputs b and PK, then D outputs b on inputs (PK, SK, x).

For a chosen cyphertext attack (abbreviated CC-attack) we also use essentially Naor and Yung's definition. A *CC-attacker* is a probabilistic program A with two inputs: a positive integer n and a string PK. A must run in expected time bounded by a polynomial in n and $|PK|$. The output of A is a single bit.

In addition to the usual complement of instructions, the program has access to oracles for (1) requesting a decryption and (2) requesting a challenge. Each of these intructions takes unit time. Before it halts, the program must execute exactly one request for a challenge. After the program executes a request for a challenge it must not execute any further oracle calls.

The results of the oracle calls are determined by the inputs n and PK and an oracle set c (the set of all binary strings that decrypt to 1.) When A is run with inputs n and PK and oracle set c, the oracle calls work as follows:

1. The input to a request for a decryption is a binary string x. The result is 1 if $x \in c$ and 0 otherwise.

2. A request for a challenge has no input. The result is a binary string x produced by first flipping a coin to determine a bit b and then running $E(b, PK)$ to obtain x.

If the program A halts, its output is a single bit b'. The CC-attacker A *succeeds* if $b' = b$, where b was the bit encrypted in the (unique) request for a challenge made by the program.

For a CC-attacker A and a positive integer n we define a boolean-valued random variable $T_A(n)$ as follows. Run G on input n to generate (PK, SK). Let c be the set of strings x such that $D(PK, SK, x) = 1$. Now run A on inputs n and PK with oracle set c, and define $T_A(n) = 1$ if A succeeds. The probability that $T_A(n) = 1$, denoted $Pr\{T_A(n) = 1\}$, depends on the random choice of b and the random choices made by G, E, and A.

Definition 5 *A public-key encryption system is said to be* secure against CC-attack *provided that for every CC-attacker A and every polynomial $p(n)$,*

$$|Pr\{T_A(n) = 1\}| < \frac{1}{2} + \frac{1}{p(n)},$$

for all sufficiently large n.

4.2 Representing decryption

Let (G, E, D) be a public-key encryption system. For any pair (PK, SK) produced by G, let

$$c(PK, SK) = \{x : D(PK, SK, x) = 1\}.$$

Let $\mathcal{C}$ be a representation of concepts. Then $\mathcal{C}$ *polynomially represents decryption in* (G, E, D) if and only if there exists a polynomial $q(n)$ such that for all positive integers n and all pairs (PK, SK) that could be output by $G(n)$, we have

$$c(PK, SK) = \kappa_{\mathcal{C}}(u)$$

for some string u such that $|u| \leq q(n)$. That is, the set $c(PK, SK)$ is a represented concept in the system $\mathcal{C}$ and has size bounded by the polynomial q in the security parameter n.

Now we can state the main theorem relating security against chosen cyphertext attack and unpredictability with membership queries.

Theorem 6 *Let (G, E, D) be any public key encryption system. Suppose $\mathcal{C}$ is a representation of concepts that polynomially represents decryption in (G, E, D). Then if (G, E, D) is secure against CC-attack, $\mathcal{C}$ is not polynomially predictable with membership queries.*

The proof of this theorem goes by assuming that $\mathcal{C}$ is polynomially predictable with membership queries and then converting a successful prediction algorithm into a successful CC-attacker. The proof is given in Section 7.

4.3 Specific systems secure against CC-attack

Naor and Yung constructed the first probabilistic encryption system secure against CC-attack, assuming the intractability of testing quadratic residuosity modulo a composite (the QRA) [19]. Recent improvements of this system due to Naor and Yung [personal communication] and Feige, Lapidot, and Shamir [8] allow the construction of similar systems assuming the existence of trapdoor permutations, and in particular under any of the three intractability assumptions used by Kearns and Valiant. Using these systems, together with the methods of Kearns and Valiant that show how decryption functions can be polynomially represented by boolean formulas or constant-depth threshold functions, we have the following.

Corollary 7 *If we assume the intractability of any of the following three problems: testing quadratic residues modulo a composite, inverting RSA encryption, or factoring Blum integers, then the following representations of concepts are not polynomially predictable with membership queries: $\mathcal{C}_{BF}$, $\mathcal{C}_{TC^0,d}$ (for any sufficiently large d), $\mathcal{C}_{3\mu}$, $\mathcal{C}_{\cup DFA}$, $\mathcal{C}_{\cap DFA}$, $\mathcal{C}_{2DFA}$, $\mathcal{C}_{NFA}$, and $\mathcal{C}_{CFG}$.*

5 Membership queries and signature schemes

We show that the existence of secure signature schemes of a certain type implies that a representation of concepts that can efficiently represent intersection and the verification of signatures can, in effect, render membership queries useless. CNF formulas are one such representation; for concreteness we sketch the idea in terms of CNF formulas.

For any CNF formula ϕ with input x we may define a polynomially larger CNF formula ϕ' on input xyz which is true if and only if ϕ is true on x, y consists of a valid signature for x with respect to some public key PK and z is a step-by-step encoding of the accepting computation of the signature verifier on PK, x and y. If CNF formulas are polynomially predictable with membership queries, the class of all such formulas ϕ' can be predicted with membership queries. In order to get an answer other than 0 from a membership query, the learning algorithm must produce a correct signature for prefix x. Thus, if the signature scheme is existentially unforgeable, the membership queries will concern only x's already seen – that is, the algorithm must succeed in predicting ϕ without membership queries. The formal development of this idea follows.

5.1 Signature schemes

Definition 8 *A public-key signature scheme consists of three Turing machines: a key-generator G, a signing program SP, and a signature verifier V, with the following properties:*

1. *G is a probabilistic machine that takes as input a positive integer n and outputs a pair of strings (PK, SK). The length of its output and its expected running time are bounded by a polynomial in n.*
2. *SP is a probabilistic machine that takes as input a pair of strings (PK, SK), and a string x, and outputs a string y. The length of its output and its expected running time are bounded by a polynomial in the lengths of its inputs.*
3. *V takes as input three strings PK, x, and y and outputs a single bit. V is a deterministic machine that runs in time bounded by a polynomial the lengths of its inputs.*

Here n is an upper bound on the length of messages to be signed, (PK, SK) is a pair of public and secret keys for the system, x is a message to be signed, and y is a signature. We require that V accept all signatures produced by SP, that is, for all n, for all (PK, SK) output by $G(n)$, and for all strings $x \in X^{[n]}$, if y is output by $SP((PK, SK), x)$ then $V(PK, x, y)$ must output 1.

The security of a public-key signature scheme against existential forgery under adaptive chosen message attack uses a polynomial time attacker with black box access to the signing program $x \rightarrow SP((PK, SK), x)$. The attacker is permitted to query the black box arbitrarily, and succeeds if it can output a correct signature for a message x not queried. We require security against a non-uniform attacker, defined below.

A *non-uniform polynomial time adaptive chosen message attacker*, (or *nonuniform forger*) F is a probabilistic advice Turing machine M_F with fixed advice tapes $t_1, t_2, \ldots$. M_F takes as as input a positive integer n and a string PK, and has access to the advice tape t_n. It must run in time polynomial in n and $|PK|$. It may make calls to an oracle that takes as input a string x of length at most n and produces as output a string y. The output of M_F is a pair of strings x' and y'.

For each n we define the binary-valued random variable $T_F(n)$ as follows. Run G on input n to produce (PK, SK). Run M_F on inputs n and PK with advice tape t_n. When M_F makes an oracle call with a string x, run $SP((PK, SK), x)$ to generate a signature y and return y as the value of the oracle call. Eventually M_F halts with outputs (x', y'). The value of $T_F(n)$ is 1 if and only if $V(PK, x', y') = 1$ and the string x' was not the input to any oracle call during the run.

Definition 9 *The signature scheme (G, SP, V) is* secure against existential forgery under nonuniform polynomial time adaptive chosen message attack *(or* n-secure*) if and only if for every nonuniform forger F and for every polynomial $p(n)$ and for all sufficiently large n,*

$$Pr\{T_F(n) = 1\} < 1/p(n),$$

where the probability is taken over the random choices of G, SP, and M_F.

The signature schemes we have defined above are memoryless, do not have an explicit signature bound, and must exhibit security against nonuniform forgers. Using the results of Goldwasser, Micali, and Rivest [11], Goldreich [9], Bellare and Micali [4], Naor and Yung [18], and Rompel [21] on the construction of secure signature schemes, we have the following.

Corollary 10 *If there exists a one-way function that cannot be inverted by polynomial-sized circuits, then an n-secure signature scheme exists.*

5.2 Polynomial representation of signatures

We define what it means for $\mathcal{C}'$ to represent "signed" versions of the concepts in $\mathcal{C}$. *$\mathcal{C}'$ polynomially represents signatures for $\mathcal{C}$ with respect to the signature scheme*

(G, SP, V) if and only if there exist two maps, a *concept map* g and an *example map* f and two nondecreasing polynomials $q_1(s,n)$ and $q_2(s,n)$ with the following properties:

1. The inputs to $g(PK, s, n, u)$ are a string PK, positive integers s and n, and a string u; the output is a string. For all positive integers s and n and all strings PK such that (PK, SK) can be output by $G(n)$, and all strings $u \in X^{[s]}$, $|g(PK, s, n, u)| \leq q_1(s,n)$.

2. The inputs to $f(PK, s, n, x, y)$ are a string PK, positive integers s and n represented in unary, and strings x, and y; the output is a string w. The function f is computable in time bounded by a polynomial in the lengths of its inputs. For all positive integers s and n, all strings PK such that (PK, SK) can be output by $G(n)$, all strings $x \in X^{[n]}$, all strings y that could be output by $SP((PK, SK), x)$, we have $|w| \leq q_2(s,n)$.

3. Moreover, if for a fixed PK, s, and n we define

$$f'(x,y) = f(PK, s, n, x, y),$$

then f' is injective and there is a polynomial time algorithm that given PK, s, n, and w determines the unique pair of strings (x,y) such that $f'(x,y) = w$ (if any.)

4. For every n and for every (PK, SK) output by $G(n)$, for every string $x \in X^{[n]}$, for every positive integer s and every string $u \in X^{[s]}$ and every string w

$$w \in \kappa_{C'}(g(PK, s, n, u))$$

if and only if for some strings x and y, we have $w = f(PK, s, n, x, y)$ and $x \in \kappa_C(u)$ and $V(PK, x, y) = 1$.

We say that $\mathcal{C}$ *polynomially represents signatures* if and only if for any signature scheme (G, SP, V), $\mathcal{C}$ polynomially represents signatures for $\mathcal{C}$ with respect to (G, SP, V).

Lemma 11 *The following representations of concepts polynomially represent signatures:* $\mathcal{C}_{3CNF}$, $\mathcal{C}_{CNF}$, $\mathcal{C}_{TC^0,d}$ *for* $d \geq 2$, $\mathcal{C}_{BF}$, $\mathcal{C}_{\cap DFA}$, $\mathcal{C}_{2DFA}$.

5.3 Bounded polynomial prediction

We need a (possibly) weaker notion of polynomial predictability without membership queries, in which we bound by a polynomial in n the size of the unknown concept and the complexity of the distribution D. A circuit with $n+1$ outputs may be taken to represent a distribution over $X^{[n]}$ in a straightforward way. A distribution D has *complexity* at most t if and only if it is representable by a circuit of at most t gates in this way.

A representation of concepts $\mathcal{C}$ is *bounded polynomially predictable* if and only if there exists a polynomial time pwm-algorithm A that does not use membership queries such that for all polynomials $p(n)$, for all but finitely many positive integers n, for all strings u of length at most $p(n)$ and for all distributions D on $X^{[n]}$ of complexity at most $p(n)$, A with inputs $p(n)$ and n, and oracles determined by concept $\kappa_C(u)$ and distribution D, the probability is at most $1/4$ that the output of A is not equal to the correct classification of x by $\kappa_C(u)$, where x is the string returned by the (unique) request for an element to predict.

Theorem 12 *Let $\mathcal{C}$ and $\mathcal{C}'$ be representations of concepts in PTIME. If (G, SP, V) is an n-secure public-key signature scheme, $\mathcal{C}'$ polynomially represents signatures for $\mathcal{C}$ with respect to (G, SP, V), and $\mathcal{C}'$ is polynomially predictable with membership queries, then $\mathcal{C}$ is bounded polynomially predictable (without membership queries.)*

The proof of this theorem is sketched in Section 7. Combining this theorem with Lemma 11 on representing signatures and Corollary 10 on assumptions sufficient to guarantee the existence of n-secure signature schemes, we have the following.

Corollary 13 *If there exists a one-way function that is not invertible by polynomial-sized circuits, then each of the following representations of concepts is either bounded polynomially predictable (without membership queries) or is not polynomially predictable with membership queries:* $\mathcal{C}_{CNF}$, $\mathcal{C}_{DNF}$, $\mathcal{C}_{TC^0,d}$ *for* $d \geq 2$, $\mathcal{C}_{BF}$, $\mathcal{C}_{\cap DFA}$, $\mathcal{C}_{\cup DFA}$, $\mathcal{C}_{2DFA}$, $\mathcal{C}_{NFA}$, $\mathcal{C}_{CFG}$.

In view of the results in the preceding section, the main thrust of the corollary is to give evidence that membership queries will not help with CNF or DNF formulas; they will either be bounded polynomial predictable without them, or won't be predictable with them.

6 Additional remarks

The learner's ability to ask membership queries raises two new issues: samplability of the underlying distributions of examples, and learnability of concepts over arbitrary domains.

Sections 4 and 5 describe two different methods for showing non-learnability with membership queries – one using chosen cyphertext secure encryption, the other using existentially unforgeable signatures. Except for CNF and DNF formulas and threshold circuits

of "small" constant depth, Corollaries 7 and 13, together with results of Kearns and Valiant [16], prove non-learnability of the same representations of concept classes under virtually the same cryptographic assumptions.

There is, however, a major difference between the underlying distributions of examples that are used; in the case of chosen cyphertext secure encryption, the underlying distributions are polynomial time samplable (by the learner), while in the case of signatures not all distributions are polynomial time samplable. Therefore, it is still open whether membership queries may help to learn CNF and DNF formulas or boolean threshold circuits of small constant depth over polynomial time samplable distributions.

The question of learnability of concepts over domains that are subsets of the "universal" domain $\{0,1\}^*$ does not arise in the "passive" model because such domains can always be extended to be all of $\{0,1\}^*$ by assigning probability 0 to the missing elements. On the other hand, when membership queries are allowed, requiring a learning algorithm to be domain-independent as well as distribution-independent may weaken the power of the algorithm. In effect, such a requirement prohibits the learner from asking queries on examples whose probability under the underlying distribution is 0, since the domain can always be restricted to consist of only examples with positive probability. By the reductions of Theorem 3 it follows that the representation of concepts $\mathcal{C}_{DFA}$, which is learnable with membership queries over domain $\{0,1\}^*$ (see [1]), becomes not learnable (under cryptographic assumptions) when the domain is restricted to a subset of $\{0,1\}^*$.

We may consider allowing a learning algorithm to ask membership queries *after* receiving an example to predict, disallowing only the query on this example. By using a new encryption scheme of Dolev, Dwork, and Naor [7], as well as secure signatures, we show that the representations of concepts shown in sections 4 and 5 to be non-learnable remain so in the new model.

Finally, we may use the techniques of Section 5 to show that Amsterdam's "experiments" may be strictly more powerful than membership queries; this development will appear in the full paper.

7 Proofs and sketches

7.1 Sample reduction: $\mathcal{C}_{BF} \leq_{pwm} \mathcal{C}_{3\mu}$

Lenny Pitt collaborated in the discovery of this reduction. For ease of understanding, we present this reduction by example. Suppose for concreteness $n = 4$ and $s = 6$.

The instance map f takes a binary string of length n and replaces each bit by s copies of that bit, e.g.,

$$f(6,4,1101) = 111111111111000000111111.$$

The concept map g takes a boolean formula ϕ over n variables of size at most s to a 3μ-boolean formula over ns variables as follows. First, it replaces each distinct occurrence of the variable X_i by distinct variables from the i^{th} group of s consecutive variables: $X_{(i-1)s+1}, \ldots, X_{is}$. This produces a read once formula ϕ'. Then it conjoins an additional formula to check that the input is in the image of f, that is, to check that the inputs in each group all have the same value.

For example, if

$$\phi = (\neg X_1 \vee X_3 \vee X_4) \wedge (X_1 \vee X_2 \vee \neg X_3)$$

then

$$\phi' = (\neg X_1 \vee X_{13} \vee X_{19}) \wedge (X_2 \vee X_7 \vee \neg X_{14}).$$

A formula to check that the inputs in the first group all have the same value is

$$\begin{aligned}\Delta_1 \;=\; & (X_1 \rightarrow X_2) \wedge (X_2 \rightarrow X_3) \wedge (X_3 \rightarrow X_4) \wedge \\ & (X_4 \rightarrow X_5) \wedge (X_5 \rightarrow X_6) \wedge (X_6 \rightarrow X_1).\end{aligned}$$

(As usual, $(A \rightarrow B)$ is $(\neg A \vee B)$.) Similarly, formula Δ_i checks that the inputs in the i^{th} group all have the same value. Hence, the 2μ-formula

$$\psi = \Delta_1 \wedge \Delta_2 \wedge \Delta_3 \wedge \Delta_4.$$

checks that the input is in the image of the instance map f. The final 3μ-formula (over 24 variables) output by the concept map is $\phi' \wedge \psi$.

The query map h is easily specified – if the input y is a string of length ns and $y = f(s,n,x)$ for some string x of length n, then the result is x, otherwise, the result is $\perp$. For example:

$$h(6,4,111111000000111111000000) = 1010,$$

while

$$h(6,4,111101000000111111000000) = \perp.$$

We omit the straightforward proof that this reduction satisfies the definition of $\leq_{pwm}$.

Note that if the original formula ϕ is in CNF then the final formula $\phi' \wedge \psi$ is also in CNF. Thus, in particular, we get the result (independently discovered by Tom Hancock [12]) that $\mathcal{C}_{CNF} \leq_{pwm} \mathcal{C}_{3\mu CNF}$. Thus, 3μCNF formulas are as hard to predict with membership queries as general CNF formulas. (Dually for DNF.) Hancock has shown that 2μDNF formulas are polynomially predictable with membership queries [12].

With a little more work we can get some insight into the oft-remarked phenomenon that finite disjunction or

conjunction seems to make learning much harder in some cases. If $\mathcal{C}$ is a class of concepts, define $\wedge(\mathcal{C})$ to be the class of concepts (under some straightforward encoding) consisting of intersections of pairs of concepts from $\mathcal{C}$.

It is not difficult to modify the reduction above so that the concept map g produces a formula that is the conjunction of three read-once formulas. The idea is to separate the read-twice checking formula Δ_1 into the conjunction of its first, third, and fifth clauses (a read-once formula) and its second, fourth, and sixth clauses (another read-once formula.)

Now consider formulas that are the conjunction of k read once formulas. For $k = 1$, such formulas are polynomially predictable with membership queries, and for $k = 3$ Corollary 7 and the modified reduction show that they are not polynomially predictable with membership queries (under suitable cryptographic assumptions.) The case of $k = 2$ is open, but whichever way it goes, there must be some class $\mathcal{C}$ which is polynomially predictable with membership queries while $\wedge(\mathcal{C})$ is not (under suitable cryptographic assumptions.)

7.2 Chosen cyphertext and membership queries

Proof of Theorem 6. Suppose that (G, E, D) is a public key encryption system that is secure against CC-attack. Let $p(n)$ be a polynomial such that for every (PK, SK) output by $G(n)$ and every x output by $E(b, PK)$ for $b \in \{0, 1\}$, $|x| \leq p(n)$.

Also suppose that $\mathcal{C}$ is a representation of concepts that polynomially represents decryption in (G, E, D). Let $q(n)$ be a polynomial such that for every n and every (PK, SK) output by $G(n)$, the size of the concept

$$c(PK, SK) = \{x \in X : D((PK, SK), x) = 1\}$$

is at most $q(n)$.

To show that $\mathcal{C}$ is not polynomially predictable with membership queries, we assume to the contrary that a pwm-algorithm A' witnesses the polynomial predictability of $\mathcal{C}$ with membership queries, and use A' to construct a successful CC-attack on (G, E, D).

The CC-attacker A on inputs n and PK simulates the pwm-algorithm A' with inputs $q(n)$ (bounding the size of the concept) and $p(n)$ (bounding the length of examples) and error parameter $1/4$. The oracle queries of A' are answered as follows.

1. When A' makes a membership query with string x, A requests the decryption of x and returns the value to A' as the result of the membership query.

2. When A' requests a random classified example, A flips a coin to determine a bit b and then runs $E(b, PK)$ to generate a string x. The pair (x, b) is then returned to A'.

3. When A' requests a element to predict, A requests a challenge. The resulting string x is returned to A as the element to predict.

When A' makes a prediction b, the CC-attacker A outputs b and halts.

For any n, consider the test to determine the value of $T_A(n)$. $G(n)$ is run to determine (PK, SK) and then the CC-attacker A is run with inputs n and PK. Consider the distribution D on strings x induced by flipping a coin to determine b and then running $E((PK, SK), b)$ to determine x. This assigns probability zero to strings longer than $p(n)$.

In the simulation of A', the membership queries are answered according to the concept $c(PK, SK)$, the random classified examples are generated according to D and classified according to $c(PK, SK)$, and the request for an element to predict is generated according to D. Since the input $q(n)$ bounds the size of $c(PK, SK)$ and the input $p(n)$ bounds the length of examples produced from D, the correctness of A' means that with probability at least $3/4$ it correctly predicts the element x returned by its request for an element to predict. This means that with probability at least $3/4$ A produces a correct decryption of the string x returned by its request for a challenge. That is, $Pr\{T_A(n) = 1\} \geq 3/4$, which contradicts our assumption that (G, E, D) is secure against CC-attack. Thus $\mathcal{C}$ is not polynomially predictable with membership queries. Q.E.D.

7.3 Signatures and membership queries

Proof sketch for Theorem 12. Let the concept map g, the example map f, and the pair of nondecreasing polynomials q_1 and q_2 witness the fact that $\mathcal{C}'$ polynomially represents signatures for $\mathcal{C}$ with respect to (G, SP, V). Let the *pwm*-algorithm A' witness the fact that $\mathcal{C}'$ is polynomially predictable with membership queries.

The prediction algorithm A to predict $\mathcal{C}$ without membership queries is defined as follows.

The algorithm A: On inputs s (bound on the concept size) and n (bound on the example length), let $s' = q_1(s, n)$ and $n' = q_2(s, n)$. A runs the key generator G on input n' to generate (PK, SK).

Now A simulates A' on inputs n' and s' with error parameter $1/8$. The oracle calls that A' makes are handled as follows.

1. When A' requests a random classified example, A requests a random classified example. Suppose (x, b) is the response to A; this pair is saved in memory. A runs $SP((PK, SK), x)$ to generate a signature y. The classified example $(f(PK, s, n, x, y), b)$ is returned to A'.

2. When A' requests an element to predict, A requests an element to predict. Suppose the element returned is x. Then A generates y as in the preceding case, and returns the string $f(PK, s, n, x, y)$ as the element for A' to predict.

3. When A' makes a membership query with string w, then A determines the unique strings x and y such that $f(PK, s, n, x, y) = w$ (if any) and checks to see that $V(PK, x, y) = 1$. If so, and if a pair (x, b) has been stored in memory, then the classification b is returned as the result of the membership query to A'. Otherwise the value returned to A' is 0.

When the algorithm A' outputs a prediction and halts, A outputs the same prediction and halts. □

Note that A makes no membership queries. Also note that it succeeds in answering the membership queries of A' in the two cases (1) the string w is not in the right form or does not contain a valid signature of x (in which case the query is correctly answered with 0) or (2) the string w is in the right form, y is a valid signature of x, and x has already occurred as a classified example (in which the correct stored value of b is returned.)

Thus, the only case in which A may fail to answer a membership query of A' correctly is when the string w is in the right form, y is a valid signature of x, but the example x has not occurred as a classified example. When this happens, A has not previously requested a signature for x, that is, y is a correct signature for a *new* x; in effect, the pair x, y is a successful forgery.

Also, if all the membership queries of A' were correctly answered, then by the correctness of A', the predictions of A' for the signed version of a concept c (and therefore, the predictions of A for the original concept c) should be correct with probability at least 7/8. If in fact there are infinitely many concepts and distributions on which A has prediction error at least 1/4, then these may be exploited to get a nonuniform forger (incorporating knowledge of the "hard" distributions and concepts) whose success probability is at least 1/8. We now proceed more formally.

We claim that A witnesses the bounded polynomial predictability of $\mathcal{C}$. The proof is by contradiction; we assume that A fails and use it to construct a successful nonuniform forger for signature scheme (G, SP, V).

If A fails to be a witness to the bounded polynomial predictability of $\mathcal{C}$, then there exist a polynomial $p(n)$ and infinitely many values $n_1, n_2, \ldots$ such that for each n_i, there exist a concept name u_i of length at most $p(n_i)$ and a distribution D_i on $X^{[n_i]}$ that can be represented by a circuit C_i of size at most $p(n_i)$, such that when A is run with inputs $s = p(n_i)$ and $n = n_i$, concept $\kappa_{\mathcal{C}}(u_i)$, and distribution D_i, the probability that A makes an error of prediction exceeds 1/4.

Consider a nonuniform forger F defined as follows. For each i, the advice tape t_{n_i} contains the string u_i and a straightforward encoding of the circuit C_i, representing the "hard" concept and distribution for n_i. The other advice tapes contain empty strings.

The machine M_F: On inputs n and PK, if the advice tape contains the empty string, then M_F halts with a pair of empty strings as output. Otherwise, $n = n_i$ for some i and the advice tape contains a string u_i and a circuit C_i representing distribution D_i. Let $s = p(n)$ and let $s' = q_1(s, n)$ and $n' = q_2(s, n)$. M_F simulates A' on inputs s', n' and 1/8. The oracle calls of A' are handled as follows by M_F:

1. When A' requests a random classified example, M_F first generates an element x according to the distribution D_i (using the circuit C_i) and then makes a call on the signing oracle with input x to get string y. M_F uses the string u_i to calculate whether $x \in \kappa_{\mathcal{C}}(u_i)$ and sets the bit $b = 1$ if so. The pair (x, b) is saved in memory, and the pair $(f(PK, s, n, x, y), b)$ is returned as the random classified example requested by A'.

2. When A' requests an element to predict, M_F generates strings x and y as in the preceding case and returns $f(PK, s, n, x, y)$ to A' as the string to predict.

3. When A' makes a membership query with a string w, M_F computes the unique strings (if any) x and y such that $w = f(PK, s, n, x, y)$. It also checks that $V(PK, x, y) = 1$. If not, it returns the value 0 to A' as the result of the membership query. Otherwise, if there is a stored pair (x, b) in memory, M_F returns the value b to A' as the result of the membership query. In the final case, $V(PK, x, y) = 1$ and x has not been an input to the oracle, so M_F outputs the successful forgery (x, y) and halts.

If A' outputs a prediction and halts, then M_F has failed to produce a successful forgery, so it outputs a pair of empty strings and halts. □

Our goal is to show that this nonuniform forger F witnesses that the signature scheme (G, SP, V) is not n-secure. We in fact show that $Pr\{T_F(n) = 1\} \geq 1/8$. (The proof of this is omitted.) Thus the scheme (G, SP, V) is not n-secure, contradicting one of our assumptions. This shows that A witnesses the bounded polynomial predictability of $\mathcal{C}$ (without membership queries.) Q.E.D.

8 Acknowledgements

The authors thank Avrim Blum, Manuel Blum, David Haussler, Michael Kearns, Moni Naor, Lenny Pitt, Ron Rivest, and Moti Yung for helpful discussions of this material. The second author thanks the International

Computer Science Institute for its hospitality. This research was funded by the National Science Foundation, under grant numbers IRI-8718975, CCR-9014943, and CCR-8858097, by a Fannie and John Hertz Fellowship, and by grants from AT&T, Digital Equipment and 3M Corporations.

References

[1] D. Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75:87–106, 1987.

[2] D. Angluin, M. Frazier, and L. Pitt. Learning conjunctions of horn clauses. In *Proceedings of the 31st Annual IEEE Symposium on Foundations of Computer Science*, pages 186–192. IEEE, 1990.

[3] D. Angluin, L. Hellerstein, and M. Karpinski. Learning read-once formulas with queries. Technical report, UC Berkeley, Report No. 89/528, 1989. (Also, ICSI Technical Report TR-89-05099. Submitted to *JACM*.).

[4] M. Bellare and S. Micali. How to sign given any trapdoor function. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 32–42. ACM, 1988.

[5] M. Blum, A. DeSantis, S. Micali, and G. Persiano. Non-interactive zero knowledge. Technical report, MIT/LCS/TM-430, 1990.

[6] M. Blum, P. Feldman, and S. Micali. Non-interactive zero-knowledge and its applications. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 103–112. ACM, 1988.

[7] D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. These Proceedings.

[8] U. Feige, D. Lapidot, and A. Shamir. Multiple non-interactive zero knowledge proofs based on a single random string. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*, pages 308–317, 1990.

[9] O. Goldreich. Two remarks concerning the GMR signature scheme. In *Proceedings of CRYPTO 86*, pages 104–110, 1986.

[10] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *J. ACM*, 33:792–807, 1986.

[11] S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17:281–308, 1988.

[12] T. Hancock. Learning 2μdnf formulas and $k\mu$ decision trees. Extended Abstract, Harvard University, 1990.

[13] D. Haussler, M. Kearns, N. Littlestone, and M. Warmuth. Equivalence of models for polynomial learnability. In *Proc. of the 1988 Workshop on Computational Learning Theory*, pages 42–55. Morgan Kaufmann Publishers, 1988.

[14] D. Haussler, N. Littlestone, and M. Warmuth. Predicting $\{0,1\}$-functions on randomly drawn points. In *Proc. 29th Symposium on Foundations of Computer Science*, pages 100–109. IEEE, 1988.

[15] M. Kearns, M. Li, L. Pitt, and L. Valiant. On the learnability of boolean formulae. In *Proc. 19th ACM Symposium on Theory of Computing*, pages 285–295. ACM, 1987.

[16] M. Kearns and L. Valiant. Cryptographic limitations on learning boolean formulae and finite automata. In *Proc. 21st ACM Symposium on Theory of Computing*, pages 433–444. ACM, 1989.

[17] N. Littlestone. Learning quickly when irrelevant attributes abound: a new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.

[18] M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pages 33–43. ACM, 1989.

[19] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proceedings of the 22d Annual ACM Symposium on Theory of Computing*, pages 427–437. ACM, 1990.

[20] L. Pitt and M. Warmuth. Reductions among prediction problems: On the difficulty of predicting automata. In *Proceedings of the Third Annual Structure in Complexity Theory Conference*, pages 60–69. IEEE Computer Society Press, 1988.

[21] J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the 22d Annual ACM Symposium on Theory of Computing*, pages 387–394. ACM, 1990.

[22] R. E. Schapire. The strength of weak learnability. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pages 28–33. IEEE, 1989.

[23] L. G. Valiant. A theory of the learnable. *C. ACM*, 27:1134–1142, 1984.