Evaluation may be easier than generation

(Extended Abstract)

Moni Naor *

Abstract

Kearns et al. [18] defined two notions for learning a distribution D. The first is with *generator*, where the learner presents a generator that outputs a distribution identical or close to D. The other is with an *evaluator*, where the learner presents a procedure that on input x evaluates correctly (or approximates) the probability that x is generated by D. They showed an example where efficient learning by a generator is possible, but learning by an evaluator is computationally infeasible.

Though it may seem that generation is, in general, easier than evaluation, in this paper we show that the converse may be true: we provide a class of distributions where efficient learning with an evaluator is possible, but coming up with a generator that approximates the given distribution is infeasible.

We also show that some distributions may be learned (with either a generator or an evaluator) to within any $\epsilon > 0$, but the learned hypothesis must be of size proportional to $1/\epsilon$. This is in contrast to the distribution-free PAC model where the size of the hypothesis can always be proportional to $\log 1/\epsilon$.

*Incumbent of the Morris and Rose Goldman Career Development Chair, Dept. of Applied Mathematics and Computer Science, Weizmann Institute of Science, Rehovot 76100, Israel. Research supported by grants from the Israel Science Foundation administered by the Israeli Academy of Sciences and by a US-Israel Binational Science Foundation. E-mail: naor@wisdom.weizmann.ac.il.

1 Introduction

The problem of learning a distribution D from several independent samples from D occupies a large part of Statistics and Pattern Recognition. However, only recently did Kearns et al. [18] put it in a computational setting. They distinguished between two notions of learning - evaluation and generation. In *evaluation* the goal of the learner is to be able to evaluate (or approximate) $D[y]$ - the probability that D generates y, given y. In *generation* the goal of the learner is to construct a distribution D' (i.e. a machine or a circuit whose output distribution is D' on uniformly random inputs) such that D' is as close as possible to D.

In the setting of [18] there is a class of distributions $\mathcal{D}_n$ over $\{0,1\}^n$. Members of $\mathcal{D}_n$ are generated by polynomial sized circuits. Such a class may be learned efficiently with a generator (with confidence δ and approximation ϵ): assuming that $D \in \mathcal{D}_n$ but is unknown, then given enough samples there is an (efficient) learner that can construct (with probability at least $1-\delta$) a distribution D' that is close to D (within ϵ under some notion of distance). The class may be efficiently learnable with an evaluator if, given enough samples, there is a learner that can successfully (with probability at least $1-\delta$) construct for $D \in \mathcal{D}_n$ an evaluator that is good for D (within ϵ under some notion of distance).

We therefore can speak of classes of distributions that are efficiently learnable with a generator and those that are efficiently learnable with an evaluator. Note that efficient learnability with evaluation is important for such tasks such as maximum likelihood computation. Kearns et al. [18] showed that generation may be easier than evaluation: there are classes that can be learned efficiently with a generator, but cannot be learned efficiently with an evaluator (unless $\#P$ has small circuits).

It might seem that in general generation is an easier task than evaluation. However, The goal of this paper is to show the converse: there are classes of distributions that are efficiently learnable with an evaluator, but which are not efficiently learnable with a generator.

Our construction of the class of distributions follows a line in computational learning theory of using Cryptography to derive hardness results [1, 5, 6, 17, 19, 28]. We construct an *irreproducible distribution* which samples uniformly at random from a sparse set $S_n \subset \{0,1\}^n$ with the following property: given points $x_1, x_2, \ldots x_m$ one cannot find an additional point $x_{m+1} \in S_n$ such that $x_{m+1} \notin \{x_1, x_2, \ldots x_m\}$, but given a point x one can determine efficiently whether $x \in S_n$ [1]. For such a distribution evaluation is easy: given x, check whether $x \in S_n$; if it is, assign probability $1/|S_n|$ (assume that $|S_n|$ is known) and otherwise assign 0. Learning with a generator is hard by assumption, as long as $|S_n|$ is not too small.

To see an intuition as to why one may suspect that we can come up with such distributions, consider the following example of a class of distributions: each distribution $D \in \mathcal{D}$ is defined by a pair of trapdoor permutations[2] (f_1, f_2) on $\{0,1\}^n$: for $r \in \{0,1\}^n$ produce $(f_1^{-1}(r), f_2^{-1}(r))$. Note that given the trapdoor information D can be generated efficiently. However, it seems reasonable to assume that without this information it is hard to generate D, even if many examples have been given, since "the only way to generate D is by inverting f_1 and f_2 at random points" which is hard. On the other hand, D has an efficient evaluator: given a pair (y_1, y_2) check whether $f_1(y_1) = f_2(y_2)$; if they are equal assign probability of 2^{-n}; otherwise assign probability 0.

Unfortunately, we do not know how to show for any family of trapdoor permutations that it is indeed hard to generate D (without access to the trapdoor information). Therefore we will explore a more complicated construction.

The next section describes the intuition of our construction and is intended to be self contained. Section 3 contains the definitions of the new notion we propose: irreproducible distributions and countable non-interactive zero-knowledge proofs and shows how they imply the existence of hard to generate but easy to evaluate distributions. It also describes the cryptographic tools we need to build such distributions. Section 4 shows how to convert a construction of [7] into a countable one, yielding the desired result. Both Sections 3 and 4 assume knowledge of the papers [7, 18].

We also address another problem from [18]: whether there are classes of distributions that are learnable but not "compressible", in the sense that essentially all the samples given are stored. Kearns et al [18] conjectured that this is indeed the case and we provide a simple construction that has this property. This is described in Section 5.

2 Intuition

A natural tool to apply for constructing irreproducible distributions are *existentially unforgeable signature schemes*, defined by Goldwasser, Micali and Rivest [16]. A signature scheme is *existentially unforgeable* if, given any polynomial (in the security parameter) number of pairs

$$(m_1, S(m_1)), (m_2, S(m_2)), \ldots (m_k, S(m_k))$$

where $S(m)$ denotes the signature on the message m, it is computationally infeasible to generate a pair $(m_{k+1}, S(m_{k+1}))$ for any message $m_{k+1} \notin \{m_1, \ldots m_k\}$. Suppose that each distribution $D \in \mathcal{D}_n$ corresponds to an instance of the signature scheme (i.e. to a pair of (public-key, private-key)). The distribution is produced by signing a random message of length n (concatenated with the public-key). The correspondence between the hardness of generating the distribution and the hardness of creating additional signed messages seems reasonable, and the efficient verification property that signature schemes posses seems to imply the ease of evaluation. Unfortunately, things are more complicated than they seem and for most existentially unforgeable signature schemes we do not know how to implement this intuition.

One problem is that many of these signature schemes have many possible signatures for each message. Another problem is that they are history dependent, i.e. the signature on message m_i is a function of the message $m_1, m_2, \ldots$. These problems can be avoided, as shown by Goldreich [14], by applying a pseudo-random function as defined and constructed in [15]. However, there may still be a difficulty in the evaluation procedure. It is true that given a message one can verify that it is legitimate, i.e. was created by someone who has access to the private key; however a problem that does not exist in signature schemes suddenly arises: what if the signature was created by

[1] This is slightly stronger than what we actually need and do. In our case we will distinguish the x_i's by their prefix and the assumption is that you cannot generate an x_{m+1} with a different prefix.

[2] A trapdoor permutation is a function that can be computed efficiently in one direction, but given y finding x such that $f(x) = y$ is computationally hard; however, there exist a secret key (the trapdoor information) so that with it inversion can be performed efficiently. The best known example of trapdoor permutation is the RSA function [27].

someone who has access to the private key but who did not follow the protocol and created what seems like legitimate message, but would *never* be generated by a signer following the protocol. There can be many such "pseudo-signatures". The evaluator assigns such messages a non-zero probability, whereas it should be zero. Getting around this problem is the main technical content of this paper and we outline how this can be done now.

Almost all the known existentially unforgeable signature schemes are "tree-based" [4, 9, 10, 16, 25] and all those scheme suffer from the problem of inconsistency. The signer is supposed to assign random (or pseudo-random) values to each node in a large tree. Each message then corresponds to a path from the root to a leaf and the values along the path are exposed. However, there is no guarantee that the signer is consistent, i.e. that the same node receives the same value each time it appears in a path. Again, this is a problem not encountered when proving the security of a signature scheme, since such a behavior is not beneficial to any side in the cryptographic setting.

We did find a way to bypass the above problems in one signature scheme, the Bellare-Goldwasser scheme [3]. It can be described roughly as follows: The public key contains a commitment to a pseudo-random function $F_k : \{0,1\}^n \mapsto \{0,1\}^n$ (i.e. to k, the key of the pseudo-random function). To sign a message m, the signer computes $z_m = F_k(m)$; the signature is z_m together with a proof that $z_m = F_k(m)$. This last part is done via a *non-interactive zero-knowledge proof.* A non-interactive proof system for a language L allows one party $\mathcal{P}$ to prove membership in L to another party $\mathcal{V}$ for any $x \in L$. $\mathcal{P}$ and $\mathcal{V}$ share a random string U of length polynomial in the security parameter n. To prove membership of a string x in $L_n = L \cap \{0,1\}^n$, $\mathcal{P}$ sends a message p as a proof of membership. $\mathcal{V}$ decides whether to accept or to reject the proof. To apply them in the context of signature scheme, the random string is part of the public-key, and the language L in which membership is proven is that of the triples $L = \{(c, m, z_m) | (c \text{ is a commitment to } k \ \& \ z_m = F_k(m)\}$.

The usage of non-interactive zero-knowledge proofs introduces many of the problems described above:

1. There may be many proofs to each statement $x \in L$
2. Even if the number of proofs can be computed or approximated efficiently, there may be many strings that look like a proof, yet would never be generated by a legitimate prover.

Non-interactive zero knowledge (NIZK) proof systems were introduced by Blum et al. in [7] who based their schemes on the hardness of distinguishing squares from non-squares modulo a composite (the quadratic residuosity assumption). There are more general and more efficient constructions [11, 20, 21], yet we do not know how to convert those proofs for our purpose (though it is conceivable that it can be done) and make them what we call *countable.* We are able to use only the scheme described in [7]. (we do employ however the method of [11] for converting single-proof system to many-proofs systems).

Non-interactive zero-knowledge has been used to construct public key cryptosystems secure against chosen ciphertext attacks in [26] which was used by Angluin and Kharitonov [1] in order to show hardness of learning under membership queries.

What we need to do in order to convert a non-interactive zero knowledge proof system for our purposes is to be able to count the number of proofs that the verifier declares as legitimate. We call such system *countable* and discuss how this can be done with the scheme of [7] in Section 4. This is done by using the proof system itself to show that the prover did not deviate from the correct protocol. The reason that this new proof does not introduce further possibilities of cheating is that we will be applying parts already used previously.

To summarize, the construction of a class of distributions that is easy to evaluate but hard to generate is as follows: each member D of the class is defined by a pseudo-random function F_k and random string U (to be used for a countable NIZK proof system). Such a distribution D outputs quintuples (m, z_m, p, U, c) where m is a uniformly random string, $z_m = F_k(m)$, p is a proof with common string U that $z_m = F_k(m)$ given c, the commitment to k (i.e. that $(m, z_m, c) \in L$). Note that c and U are the same for every string D outputs.

3 Tools, Definitions and Constructions

3.1 Definitions

We first propose a new concept and then define some of the tools we need for our construction. We embrace the notions of Kearns et al. [18] of efficient learning of distributions with generators and with evaluators. They considered the Kullback-Leibler divergence as the distance between distributions D_1 and D_2, say the original one and the one that the learner produces[3] (either with a generator or an evaluator). Our results are applicable to other notions of distance such as the L_1 distance.

For a string $x \in \{0,1\}^n$ and for $m \leq n$ let the m-prefix(x) be the prefix of length m of x. For a distribution D let S_D be its support.

Definition 3.1 *Let $\mathcal{D}_n$ be a class of distributions over $\{0,1\}^n$ and let $m(n) \leq n$. We say that $\mathcal{D}_n$ is* irreproducible *if*

1. *For any $D \in \mathcal{D}_n$ there exists a polynomial time generator G_D that on random input its output is distributed according to D.*
2. *There exist an efficient* key generation algorithm *that can creates G_D for a $D \in_R \mathcal{D}_n$.*
3. *For any polynomial $\ell(n)$ and any polynomial sized family of circuits $\mathcal{C} = \{C_n | n \geq 1\}$ such that on input $x_1, x_2, \ldots x_{\ell(n)} \in_R D$ the circuit C_n attempts to create $x_{\ell(n)+1} \in S_D$ with*

$$m(n)\text{--prefix}(x_{\ell(n)+1}) \notin \{m(n)\text{--prefix}(x_1), \\ m(n)\text{--prefix}(x_2), \ldots m(n)\text{--prefix}(x_{\ell(n)})\}$$

   has probability of success smaller than $1/poly(n)$.

4. *If x_1 and x_2 are drawn independently from D, then*

$$Prob[m(n)\text{--prefix}(x_1) = m(n)\text{--prefix}(x_2)]$$

   is smaller than $1/poly(n)$.

From the definition the following is quite evident (it can be shown following proof of Theorem 17 of [18]):

Theorem 3.1 *An irreproducible class of distributions cannot be learned with a generator.*

Theorem 3.2 *Let $\mathcal{D}_n$ be an irreproducible class of distributions such that for any $D \in \mathcal{D}_n$*

1. *The distribution induced on the $m(n)$-prefix on $x \in D$ is uniform on $\{0,1\}^{m(n)}$.*
2. *There exists an efficient procedure $A(x)$ such that given several samples from D can approximate, on input x, $Prob_D[x | m(n)\text{--prefix}(x)]$ to within relative error $\epsilon(n)$ where $\epsilon(n)$ is smaller than $1/poly(n)$.*
3. *There exists an efficient procedure $B(x)$ such that given several samples from D can determine whether $x \in S_D$.*

Then $\mathcal{D}_n$ can be learned efficiently by an evaluator.

Proof. First execute procedures A and B on the number of samples they require. Then apply $B(x)$ to test whether $x \in S_D$ and then run the approximator to get $a = A(x)$. The evaluation on the probability for generating x is $\frac{a}{2^{m(n)}}$. This means that for every $x \in S_D$ $\frac{a}{2^{m(n)}}$ is within relative error of $\epsilon(n)$ from $Prob_D[x]$. Therefore the Kullback-Leibler divergence of the estimation and the true distribution is bounded by $\epsilon(n)$. □

From these two theorems it is clear that if we manage to construct a class of irreproducible distributions satisfying the conditions of Theorem 3.2 then we have achieved our goal.

3.2 Tools

We now describe some of the existing cryptographic tools which we use. We do not describe most of them formally, since we do not need to alter those definitions. The only modification we need to make is in the definition of NIZK.

A *pseudo-random function*, as defined by Goldreich, Goldwasser and Micali [15], is a function that is indistinguishable from a truly random function to a (probabilistic polynomial-time bounded) observer who can access the function as a black-box i.e. can provide inputs of his choice and inspects the value of the function on these inputs. We need a family of functions

$$\mathcal{F} = \{F_k : \{0,1\}^{m(n)} \mapsto \{0,1\}^{m(n)} |\ k \in \{0,1\}^{k(n)}\}$$

where $k(n)$ and $m(n)$ are polynomials in n. Pseudo-random functions can easily yield irreproducible distributions: a distribution D is determined by k and on input $m \in_R \{0,1\}^{m(n)}$ it produces $(m, F_k(m))$. However, this distribution does not satisfy the conditions of Theorem 3.2. (In fact, it was used in Theorem 17 of [18] to demonstrate the existence of distributions that are hard to learn by generators *and* by evaluators).

A *bit commitment* protocol is a basic component of many cryptographic protocols. The committer, P, *commits* to a secret bit b to be revealed later to the receiver, Q. The commitment is such that once it is completed, but before the value is revealed, b is (computationally) unknown to Q; however, when the value b is revealed, the receiver Q will be able to determine that the correct value has been revealed – i.e. there

[3] The Kullback-Leibler distance between two distributions is defined to be $KL(D_1 || D_2) = \sum_{x \in \{0,1\}^n} D_1(x) \log \frac{D_1(x)}{D_2(x)}$. For more information see Cover and Thomas [8].

has been no switch. A good analogy is that P writes the bit and puts it in a locked box to which only she has the key. She gives the box to Q (this is the commit stage) and, when the time is ripe, she gives the key to Q. The receiver Q can be certain that the contents were not tampered with, since the box was in its possession. For our purposes we require that in the bit commitment protocol, the commit stage consists of a single message (sent by the committer of course) and that the reveal stage is also a single message (also sent by the committer). Furthermore, we need that in the reveal stage there will be exactly one string that the committer can send and it will be accepted by the receiver (this does not refer to the value being committed, which in all bit commitment protocols is unique, but to the actual communication in the reveal stage). The protocol of [23] can have these properties, provided that it is based on a one-way *permutation* (and not on any one-way function).

The following is the definition of non-interactive proof systems of [7], which was modified in order to incorporate the tractability of the prover $\mathcal{P}$ and to which we add the new *countability* requirements.

Definition 3.2 *A triple $(\mathcal{P}, \mathcal{V}, \mathcal{U})$, where $\mathcal{P}$ is a probabilistic machine, $\mathcal{V}$ is a polynomial time machine, and $\mathcal{U}$ is a polynomial time sampleable probability distribution is a* non-interactive zero-knowledge proof system *for the language $L \in NP$ with witness set $WL(x)$ for every $x \in L$ if:*

1. `Completeness` *(if $x \in L$ then $\mathcal{P}$ generates a proof that $\mathcal{V}$ accepts): For all $x \in L_n$, for all $z \in WL(x)$, with overwhelming probability for $U \in_R \mathcal{U}(n)$ and $p \in_R \mathcal{P}(x, z, U)$, $p \in ACCEPT(U, x)$. The probability is over the choice of the shared string U and the internal coin flips of $\mathcal{P}$.*

2. `Soundness` *(if $y \notin L$ then no prover can generate a proof that $\mathcal{V}$ accepts): For all $y \notin L_n$ with overwhelming probability over $U \in_R \mathcal{U}(n)$ for all $p \in \{0,1\}^*$, $p \in REJECT(U, y)$. The probability is over the choices of the shared string U.*

3. `Zero-knowledge` *(there is a probabilistic polynomial time machine Sim which is a simulator for the system): For all probabilistic polynomial time machines $\mathcal{C}$, if $\mathcal{C}$ generates $x \in L$ and $z \in WL(x)$ then,*

$$\begin{aligned} | \quad & Prob[\mathcal{C}(w) = 1 | w \in_R Sim(x)] - \\ & Prob[\mathcal{C}(w) = 1 | w \in_R \mathcal{CONV}(x, z)] | < \frac{1}{p(n)} \end{aligned}$$

for all polynomials p and sufficiently large n.

The new property we need is `countability`*: given an $x \in L$ and a proof $p \in ACCEPT(U, x)$, there is an efficient procedure for estimating*

$$Pr[p \text{ is generated by } \mathcal{P}(x, z, U)].$$

A scheme which satisfies the 4th condition as well is called *countable*. In section 4 we describe how to convert the scheme of [7] into a countable one, where one can estimate $|ACCEPT(U, x)|$ and $p \in_R \mathcal{P}(x, z, U)$ is close to uniform in $ACCEPT(U, x)$.

3.3 The construction

We now define the class of distributions $\mathcal{D}_n$ which is easy to evaluate but hard to generate given the above tools. Each member D of the class D_n is determined by

1. A pseudo-random function
$$F_k : \{0,1\}^{m(n)} \mapsto \{0,1\}^{m(n)}$$
specified by the key k.

2. A commitment c to k according to the protocol above.

3. A string $U \in \mathcal{U}_{good}$ which is to be used for a countable NIZK proof system and where
$$\begin{aligned} \mathcal{U}_{good} = \{U \quad | \quad & \nexists x \notin L \ \& \ p \text{ s. t.} \\ & p \in ACCEPT(x, U)\}. \end{aligned}$$
Note that by assumption on the NIZK proof system almost all strings are good and we can easily sample from $\mathcal{U}_{good}$.

Such a distribution D defined by (k, c, U) outputs quintuples (m, z_m, p, U, c) where $m \in \{0,1\}^{m(n)}$ is chosen uniformly at random, $z_m = F_k(m)$, and p is a proof with common string U that $z_m = F_k(m)$ given c, the commitment to k (i.e. that $(m, z_m, c) \in L$). Note that c and U are the same for every string D outputs. We choose $m(n)$ so that the length of the quintuple (m, z_m, p, U, c) is n.

The evaluator for D needs only one sample from D in order to get U and c. From this point on, given a quintuple (m, z_m, p, U', c'), to evaluate its probability check whether $U' = U$ and $c' = c$ and whether p is indeed a proof that $(m, z_m, c) \in L$ (i.e. that $p \in ACCEPT(U, (m, z_m, c)))$. Estimate $a = Pr[p$ is generated by $\mathcal{P}((m, z_m, c), \text{reveal}(k), U)]$ where reveal(k) is the string used to reveal that k is the committed string. Output $a/2^{m(n)}$.

Theorem 3.3 *$\mathcal{D}_n$ cannot be learned by a generator, but can be learned by an evaluator.*

Proof. (Sketch) We show that the conditions of Definition 3.1 and Theorem 3.2 are satisfied and hence we get the desired result from Theorems 3.1 and 3.2. Properties (1) and (2) of Definition 3.1 are satisfied since all the tools used have efficient key generators: the pseudo-random functions, the bit-commitment schemes and $\mathcal{U}_{good}$. Property (3) follows from the security of the pseudo-random function F_k and the non-interactive zero-knowledge proof system. It can be shown, as in [3], that if this property is not satisfied, then either the pseudo-random F_k can be distinguished from a truly random function or that there is no simulator for the NIZK proof system. Property (4) is satisfied simply by choosing $m(n)$ to be large enough.

Condition (1) of Theorem 3.2 is satisfied since m is chosen at random from $\{0,1\}^{m(n)}$. The existence of procedure A and B follows, since given one sample from D the learner may obtain U and c. The soundness of the NIZK proof system yields B. As for A, make $A(x)$ to be the inverse of the estimate of $Pr[p \text{ is generated by } \mathcal{P}(x,z,U)]$. □

4 A countable non-interactive zero-knowledge proof system

Our goal in this section is to describe a non-interactive zero-knowledge proof system with the countability property, i.e. that it is possible to derive a good estimate on the number of proofs a given statement has. It is heavily dependent on [7] and we assume familiarity with that paper. The description is, however, rather informal.

We are interested in proving membership in the language

$$L = \{(c, m, z_m) | (c = \text{ commitment to } k \ \& \ z_m = F_k(m)\}$$

defined above. We can convert the proof system for 3-SAT into a proof for L (from the NP-Completeness of 3-SAT). There are several thing to note before we start: L is a circuit satisfiability problem that either has one satisfying assignment or none. This follows from the property we required from the bit commitment protocol: the receiver should accept only one string as the committer message at the reveal stage. Given that the circuit satisfiability problem has either a unique assignment or none, the reduction to 3-SAT preserves this property. (Note that without this property we should have applied tools from [32, 22] which would have introduced some non-negligible error.)

The NIZK proof system for 3-SAT of [7] contains the following parts:

1. Select a Blum integer $N = PQ$ where P and Q are primes of length at most n bits and y a quadratic non-residue such that $y \in J_N^+$ (has Jacobi symbol 1), We require that P and Q will have at least, say $2/3n$ bits.

2. Prove using one part of the public string U that N is indeed a Blum integer and y a quadratic non-residue.

3. Prove the satisfiability of the formula using this N and the second part of U.

We claim that once N has been fixed, counting the number of legitimate proofs that choose N is easy, at least within a good approximation. For the part of proving that N is a Blum integer the prover has to provide a square-root for a sequence of numbers (determining which numbers should get a square root is easy - it depends on the Jacobi symbol). There are 4 possibilities for the square-root, so enumerating this part is easy. As for the number of possible y's, it is $\phi(N)/4$. The verifier does not know $\phi(N)$ (it would yield the factorization of N), however it can be approximated to within ϵ. Furthermore, since $N = P \cdot Q$ we know that $\phi(N)/N \approx (1 - 1/\sqrt{N})$.

As for the second part, the prover has to select

1. A random square modulo N for each variable. This can be estimated quite accurately, as described above, by taking $N/4$. Furthermore we can modify the protocol slightly and make the prover chose a number that is not necessarily relatively prime to N, in which case we can calculate the probability exactly. This does not hurt the zero-knowledge property of the protocol since it is a rare event.

2. 7 triples $(\alpha_i, \beta_i, \gamma_i)$ for each clause (of the 3-SAT formula) where each one is selected by choosing a random square and multiplying by the non-square y as needed. This implies choosing 21 squares and a permutation on six elements. This is, again, simple to count.

3. A square-root for a sequence of numbers a square root (4 possibilities).

Given N and a proof $p \in ACCEPT(U, x)$ all the above estimates are multiplied. Since each on of the estimates is very good (to within relative error of $1/2^{n/3}$), the relative error of their product is also small. Therefore we see that given N the number of possible proofs of a member of L can be approximated efficiently to within relative error of $\epsilon = 1/2^{n^\delta}$ for some $\delta > 0$. .

The remaining problem is calculating the probability that N is selected. Here the problem is that N should be a product of two primes of equal length n. Estimating the number of N's of this form can be done either analytically or "empirically", generating random *pre-factored* numbers (according to Bach's algorithm [2]) and testing which fraction has the proper form.

The catch, however, is that the proof that N is a Blum integer only shows that it has two distinct prime divisors (and is not a perfect square), it does not necessarily imply that these primes are of length n. This has no effect of the soundness of the proof, so it was not a problem considered in [7]. However, here we should estimate the number of valid proofs and it becomes significant. Note that allowing N to have a very large and a very small factor might make it factorable, in which case the zero-knowledge property of the proof system is destroyed. Furthermore, suppose we change the distribution of selecting N from selecting P and Q as n bit primes and forming $N = P \cdot Q$ to generating a sequence of random pre-factored numbers until one which is a product of two primes is encountered. This allows a non-negligible probability of having a very small factor, which does not guarantee the zero-knowledge property. We thus should find a way of proving that N was formed properly.

We now come to the punch-line and major modification of the scheme of [7] which we apply. We do not know of any direct way of demonstrating that all the divisors of a number have at most n bits, and therefore we will employ the general 3-SAT protocol of [7]. This seems to reintroduce the problem of estimating this number of proofs, since the prover has to select an N' (which, again should be a product of two primes etc.) for that purpose and we are back to square one. However, we solve this by making $N' = N$ and adding a third part to the common random string U' with which the fact that N is of the right form is proved using the 3-SAT protocol. This might seem similar to the grave sin of cryptography: encrypting a key using the same key. However, we can show that the resulting scheme is still secure. This follows from the way the zero-knowledge simulator of [7] was constructed. It works just as well even if N is fixed and known *before* the theorem whose proof will be simulated using N is selected. This implies that theorem may depend on N and therefore the zero-knowledge property is not altered[4].

Therefore it is possible to estimate the probability that a given proof p is generated by the prover $\mathcal{P}$ to within relative error $\epsilon = 1/2^{n^\delta}$ and we can conclude with the following theorem:

Theorem 4.1 *The modified proof system for L is a countable non-interactive zero-knowledge.*

5 Learning distributions may require a lot of storage

In this section we address another problem raised in [18]: whether there are classes of distributions that are learnable but not "compressible", in the sense that essentially all the samples given must be stored. Kearns et al. [18] suggested a family of distributions for which they conjectured that this is indeed the case, i.e. that in order to learn a member of this family to within ϵ, one needs a generator of size proportional to polynomial in $1/\epsilon$. We do not know how to prove this conjecture, but we provide an alternative construction for which we can prove the desired property.

We show that assuming pseudo-random generators exist, then there exists a family of distributions $\mathcal{D}$ such that any $D \in \mathcal{D}$ is learnable with a generator to within error ϵ, for any $\epsilon > 0$, but the learned generator must be of size at least $\Omega(1/\epsilon)$. This is in contrast to the distribution-free PAC model, where the results of Schapire and Freund [30, 12] on precision boosting imply that the size of the hypothesis can always be be much smaller than the number of samples, and be only polynomial in $\log 1/\epsilon$.

The family $\mathcal{D}_n$ we construct may be based on any pseudo-random generator. A member of $\mathcal{D}_n$ is defined by a seed s to a pseudo-random sequence denoted by X_s. (It is better to think of X_s as a pseudo-random sequence with random-access, i.e. a pseudo-random function; the two assumptions are equivalent [15].) Once s is fixed, the distribution D_s outputs $(i, X_s[i])$ where $1 \le i \le 2^{n-1}$ is chosen so that $Prob[i > k] = 1/k$ by making $Prob[i = k] = \frac{1}{k(k-1)}$ for $k \ge 2$. (Since we chop the sequence at 2^{n-1}, let $Prob[i = 2^{n-1}] = \frac{1}{2^{n-1}}$. It can be easily verified that

[4] The reason the simulator can publish N before getting the element whose membership should be proven is that the simulated U consists only of quadratic residues modulo N (and numbers whose Jacobi symbol is -1) and does not depend at all on the element.

any $D_s \in \mathcal{D}_n$ can be generated by a polynomial size circuit.

In order to learn with a generator such a distribution to within ϵ (in the L_1 norm sense, you can get a similar result for the Kullback-Leibler distance), sample the distribution until you have seen all $i \leq 1/\epsilon$ (this should not take more than $\epsilon^{-2} \log 1/\epsilon$ samples). Record the ith value of the sequence in $X[i]$. Following the learning stage, simulate the distribution in the natural way, by choosing i according to the above distribution. If $i \leq 1/\epsilon$, output $(i, X[i])$; if it is larger, output (i, b) for random $b \in \{0, 1\}$. The distance between the generated distribution and the true one is bounded by ϵ since this is the bound of the probability that $i > 1/\epsilon$.

The following is the key observation: given a black box whose output distribution D' supposedly approximates $D_s \in \mathcal{D}$ to within ϵ, we can extract the first $\sqrt{1/\epsilon}$ bits of X_s without errors and the first $1/\epsilon$ bits of X_s with at most $\alpha\epsilon$ errors for some $\alpha < 1/3$. This is done by sampling D' many times, say $\epsilon^{-2}k$ for some $k > 1$. For each $i \leq 1/\epsilon$ record the number of time $(i, 0)$ and the $(i, 1)$ were output (the distribution D' is not necessarily consistent on i). Take $X[i]$ to be the majority of the $(i, *)$'s. If $|D'[(i, 0)] - D'[(i, 1)]| > 1/3$, then with high probability[5] over the samples from D' $X[i]$ is the larger of the two, i.e. $D'[(i, X[i])] > D'[(i, 1 - X[i])]$.

If $X[i] \neq X_s[i]$ or $|D'[(i, 0)] - D'[(i, 1)]| \leq 1/3$ for some $i \leq \sqrt{1/\epsilon}$, then the distance between D' and D_s is larger than ϵ. Regarding $i \leq 1/\epsilon$, suppose that for more than α/ϵ of them either $|D'[(i, 0)] - D'[(i, 1)]| \leq 1/3$ or D' and D "disagree on i. (In both cases $|D'[(i, 0)] - D[(i, 0)]| + |D'[(i, 0)] - D[(i, 0)]| \geq 2/3$.) The closest D' can be to D is when the disagreement occurs at the α/ϵ largest i's. But the distance in this case is $2/3 \cdot \epsilon(1/\alpha - 1) \geq \epsilon$.

To see why this observation implies that that there are no succinct representations for the learner of $\mathcal{D}$, consider the family $\mathcal{D}_\epsilon$ of distributions whose members D_{rs} are defined by a truly random sequence r of length $1/\epsilon$, continued with the pseudo-random sequence determined by s for larger i's. The family of distributions $\mathcal{D}_\epsilon$ is indistinguishable from the original one $\mathcal{D}_n$. Suppose we apply the learning algorithm for $\mathcal{D}_n$ on a $D_{rs} \in \mathcal{D}_\epsilon$ and get a circuit whose output is D'. By the above argument we can fully reconstruct the first $\sqrt{1/\epsilon}$ bits of r from the circuit for D', except with some probability $1 - \delta'$ where δ' is a bound on probability of distinguishing a pseudo-random sequence from a truly random one and may be sub-polynomial (if it is not the case, then we have a distinguisher for the pseudo-random generator.) Therefore, by a counting argument the size of the circuit must be at least $(1 - \delta') \cdot \sqrt{1/\epsilon}$. Furthermore, we can reconstruct correctly at least $2/3$ of the bits of r, and again by a counting argument (based on Coding Theory) it must be the case the size of the circuit is $\Omega(1/\epsilon)$.

A similar argument works for learning with an evaluator. We therefore conclude

Theorem 5.1 *Assuming X_s is a pseudo-random generator, then the family of distributions $\mathcal{D}$ is such that any $D \in \mathcal{D}$ is learnable with a generator (or evaluator) to within error ϵ, for any $\epsilon > 0$, but the the length of the description of the learned generator (or evaluator) must be of size $\Omega(1/\epsilon)$ with high probability for most $D \in \mathcal{D}$.*

6 Open questions and further work

Non-interactive zero-knowledge proof systems are quite inefficient (though still of polynomial size), so a reasonable question is whether they are really necessary for constructing irreproducible distributions satisfying the conditions of Theorem 3.2. This question is not so relevant to the problem considered in this paper, of demonstrating that evaluation may sometimes be easier than generation, since efficiency is not a major consideration here. However, we can show the following: given an irreproducible distribution that has an efficient procedure for membership in the support (procedure $B(x)$ in the condition of Theorem 3.2) and a trapdoor permutation, we can construct an existentially unforgeable signature schemes. The interesting property of this scheme is that it is not tree based, which has as least the potential of being efficient.

A different problem in learning distributions where cryptographic tools may be applicable is that of learning finite automata. In [18] the problem of learning finite automata with an evaluator is related to the noisy parity problem, but it is not clear whether hardness based on standard cryptographic assumptions can be shown or what is the status of learning such distributions with a generator. The recent construction of synthesizers in NC^1 [24] seems related, but we have not been able to use it.

[5] this probability is exponentially in k close to 1, so we assume that for all $1 \leq i \leq 1/\epsilon$ this event did not happen and X records faithfully all those i's such that $|D'[(i, 0)] - D'[(i, 1)]| > 1/3$.

Acknowledgments

I would like to thank Ronitt Rubinfeld for introducing me to this area and Oded Goldreich and Dan Roth for several useful discussions and suggestions.

References

[1] D. Angluin and M. Kharitonov, *When won't membership queries help*, Proc. 23rd ACM Annual Symposium on the Theory of Computing, 1991, pp. 444–454.

[2] E. Bach, *How to generate factored random numbers*, SIAM J. Computing 17, 1988, 179–193.

[3] Bellare, M. and S. Goldwasser, *New paradigms for digital signatures and message authentication based on non-interactive zero knowledge proofs*, Advances in Cryptology - CRYPTO 89, Lecture Notes in Computer Science, vol. 435, Springer-Verlag, 1990, pp. 194–211.

[4] M. Bellare and S. Micali, *How to Sign Given Any Trapdoor Function*, Proc. 20th ACM Annual Symposium on the Theory of Computing, 1988, pp. 32–42.

[5] A. Blum, *Separating distribution-free and mistake-bound learning models over the Boolean domain*, Proc. 31st IEEE Symposium on Foundations of Computer Science, 1990, pp. 211–218.

[6] A. Blum, M. Furst, M. Kearns, and R. Lipton. *Cryptographic Primitives Based on Hard Learning Problems*. Advances in Cryptology — CRYPTO 93, Lecture Notes in Computer Science 773, , Springer-Verlag, 1994, pp. 278–291.

[7] M. Blum, A. De Santis, S. Micali and , G. Persiano, *Non-Interactive Zero-Knowledge*, SIAM J. Computing, 1991, pp. 1084–1118.

[8] T. M. Cover and J. A. Thomas, **Elements of Information Theory**, Wiley, 1991.

[9] C. Dwork and M. Naor, *An Efficient Existentially Unforgeable Signature Scheme and its Applications*, Advances in Cryptology – CRYPTO '94, Springer Verlag, 1994, pp. 234–246.

[10] S. Even, O. Goldreich, and S. Micali, *On-line/Off-line Digital Signatures*, Advances in Cryptology – CRYPTO '89, Lecture Notes in Computer Science, vol. 435, Springer-Verlag, 1990, pp. 263–275.

[11] U. Feige, D. Lapidot and A. Shamir, *Multiple Non-Interactive Zero-Knowledge Proofs Based on a Single Random String*, Proc. of 31st Symposium on Foundations of Computer Science, 1990, pp. 308–317.

[12] Y. Freund, *An improved boosting algorithm and its implication on learning complexity*, Proc. 5th ACM Workshop on Computational Learning Theory, 1992, pp. 391–398.

[13] Goldreich, O., **Foundations of Cryptography** (Fragments of a Book) 1995. Available in the Electronic Colloquium on Computational Complexity: http://www.eccc.uni-trier.de/eccc/info/ECCC-Books/eccc-books.html.

[14] O. Goldreich, *Two remarks concerning the Goldwasser-Micali-Rivest signature scheme*, Advances in Cryptology - CRYPTO' 86, Lecture Notes in Computer Science, vol. 263, Springer-Verlag, 1987, pp. 104–110.

[15] O. Goldreich, S. Goldwasser and S. Micali, *How to construct random functions*, J. of the ACM. 33 (1986) 792–807.

[16] S. Goldwasser, S. Micali and R. Rivest *A secure digital signature scheme*, SIAM J. on Computing 17, 1988, pp. 281–308.

[17] M. Kearns and L. Valiant, *Cryptographic limitations on learning Boolean formulae and finite automata, J. of the ACM.* 41(1) (Jan, 1994) 67–95.

[18] M. Kearns, Yishai Mansour, Dana Ron, Ronitt Rubinfeld, Robert E. Schapire and Linda Sellie, *On the learnability of discrete distributions*, Proc. 26th ACM Annual Symposium on the Theory of Computing, 1994, pp. 273–282.

[19] M. Kharitonov, *Cryptographic hardness of distribution-specific learning*, Proc. 25th ACM Symposium on Theory of Computing, (1993) 372–381.

[20] J. Kilian, *On the complexity of bounded-interaction and non-interactive zero-knowledge proofs*, Proc. of the 35th IEEE Symposium on the Foundation of Computer Science, 1994.

[21] J. Kilian and E. Petrank, *An efficient non-interactive zero-knowledge proof system for NP with general assumptions*, Electronic Colloquium on Computational Complexity TR95-038. Available: http://www.eccc.uni-trier.de/eccc/info/ECCC.

[22] K. Mulmuley, U. V. Vazirani and V. V. Vazirani *Matching is as easy as matrix inversion*, Combinatorica **7**, 1987, 105–113.

[23] M. Naor, *Bit commitment using pseudo-randomness,* Journal of Cryptology, vol 4, 1991, pp. 151–158.

[24] M. Naor and O. Reingold, *Synthesizers and their applications* Proc. 36th IEEE Symposium on Foundations of Computer Science, 1995, pp. 170–181.

[25] M. Naor and M. Yung, *Universal one-Way hash functions and their cryptographic applications*, Proc. 21st ACM Annual Symposium on the Theory of Computing, 1989, pp. 33–43.

[26] M. Naor and M. Yung, *Public key cryptosystems provably secure against chosen ciphertext attacks*, Proc. 22nd ACM Annual Symposium on the Theory of Computing, 1990, pp. 427–437.

[27] Rivest, R.L., A. Shamir, and L.M. Adleman, A method for obtaining digital signature and public key cryptosystems, *Comm. ACM* 21 (1978) 120–126.

[28] Rivest, R.L., *Cryptography and Machine Learning*, Proc. ASIACRYPT'91, Springer Verlag, 1993, pp. 427–439.

[29] J. Rompel, *One-way Function are Necessary and Sufficient for Signatures*, Proc. 22nd ACM Annual Symposium on the Theory of Computing, 1990, pp. 387–394.

[30] R. E. Schapire, *The strength of weak learnability,* Machine Learning 5, 1990, pp. 197–227.

[31] A.Sinclair and M. Jerrum, *Approximate Counting, Uniform Generation, and Rapidly Mixing Markov Chains*, Information and Computation, 82:93–133, 1989.

[32] L. G. Valiant and V. V. Vazirani, *NP is as easy as detecting unique solutions*, Theoretical Computer Science **47**, 1986, pp. 85–93.