Number-Theoretic Constructions of Efficient Pseudo-Random Functions

(Extended Abstract)

Moni Naor * Omer Reingold †

Abstract

We describe efficient constructions for various cryptographic primitives (both in private-key and in public-key cryptography). We show these constructions to be at least as secure as the decisional version of the Diffie-Hellman assumption or as the assumption that factoring is hard. Our major result is a new construction of pseudo-random functions such that computing their value at any given point involves two multiple products. This is much more efficient than previous proposals. Furthermore, these functions have the advantage of being in TC^0 (the class of functions computable by constant depth circuits consisting of a polynomial number of threshold gates) which has several interesting applications. The simple algebraic structure of the functions implies additional features. In particular, we show a zero-knowledge proof for statements of the form "$y = f_s(x)$" and "$y \neq f_s(x)$" given a commitment to a key s of a pseudo-random function f_s.

1 Introduction

This paper studies the efficient construction of several fundamental cryptographic primitives and reduces their security to the *decisional* version of the Diffie-Hellman assumption (**DDH-Assumption**). Our major construction is of pseudo-random (**p.r.**) functions. We also show a simple randomized-reduction between the worst-case and average-case of the DDH-Assumption that may increase our confidence in this assumption. In addition, we show a related construction of p.r. functions that is at least as secure as the *generalized* DH-Assumption (**GDH-Assumption**). Breaking the GDH-Assumption modulo a composite would imply an efficient algorithm for factorization (see [6, 49]). Therefore, we get an attractive construction of p.r. functions that is at least as secure as factoring.

P.r. functions were introduced by Goldreich, Goldwasser and Micali [27] and have innumerable applications (e.g., [3, 8, 19, 24, 25, 28, 38, 42, 43]). A distribution of functions is p.r. if: (1) It is easy to sample functions according to the distribution and to compute their value. (2) It is hard to tell apart a function sampled according to this distribution from a uniformly distributed function given access to the function as a black-box. Our new construction of p.r. functions is:

Efficient Computing the value of the function at a given point is comparable with two modular exponentiations and is more efficient by an $\Omega(n)$ factor than any previous proposal (that is proven to be as secure as some standard intractability assumption). This is essential for the efficiency of the many applications of p.r. functions.

Shallow Given appropriate preprocessing, the value of the functions at any given point can be computed in TC^0, compared with TC^1 previously (in [41]). Therefore this construction:

1. Achieves reduced latency for computing the functions in parallel and in hardware implementations.
2. Has applications to computational complexity (i.e., Natural Proofs [44]) and to computational learning-theory.

For more on these applications see Section 2.2.

Simple The simple algebraic structure of the functions implies additional desirable features. To demonstrate this, we show a simple zero-knowledge proof for the value of the function and other protocols. We suggest the task of designing additional protocols and improving the current ones as a line for further research.

*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 grant no. 356/94 from the Israel Science Foundation administered by the Israeli Academy of Sciences and by BSF grant no. 94-00032. E-mail: naor@wisdom.weizmann.ac.il.

†Dept. of Applied Mathematics and Computer Science, Weizmann Institute of Science, Rehovot 76100, Israel. Research supported by a Clore Scholars award. E-mail: reingold@wisdom.weizmann.ac.il.

The DDH-Assumption

In the following few paragraphs we briefly describe the DDH-Assumption, its different applications and the current knowledge on its security.

The DH-Assumption was introduced in the context of the Diffie and Hellman [20] key-exchange protocol (among quite a few of the fundamental ideas and concepts of public-key cryptography). Any method for exchanging even a single bit, using this protocol, relies on the *computational* version of the DH-Assumption (**CDH-Assumption**). By assuming its (stronger) *decisional* version one can exchange many bits. For concreteness we consider the DDH-Assumption in a subgroup of $\mathbb{Z}_P^*$ (the multiplicative group modulo P) of order Q, where P and Q are large primes and Q divides $P-1$. For such P and Q the DDH-Assumption is:

> *There is no efficient algorithm that, given* $\langle P, Q, g, g^a, g^b \rangle$, *distinguishes between* $g^{a \cdot b}$ *and* g^c *with non-negligible advantage. Where,* g *is a uniformly chosen element of order* Q *in* $\mathbb{Z}_P^*$, a, b *and* c *are uniformly chosen in* $\mathbb{Z}_Q$ *and all exponentiations are in* $\mathbb{Z}_P^*$.

Note that this assumption does not hold when g is a generator of $\mathbb{Z}_P^*$.

It turns out that the DDH-Assumption was assumed in quite a few previous works (both explicitly and implicitly). All these applications rely on the average-case assumption described above. In Section 3.2, we show that for any given P and Q the DDH-assumption can be reduced to its worst-case version[1]:

> *There is no efficient algorithm that, given* $\langle P, Q, g, g^a, g^b, g^c \rangle$, *decides with overwhelming success probability whether or not* $c = a \cdot b \pmod{Q}$ *for* every a, b *and* c *in* $\mathbb{Z}_Q$ *and* every *element,* g, *of order* Q *in* $\mathbb{Z}_P^*$ *(all exponentiations are in* $\mathbb{Z}_P^*$*).*

The randomized-reduction we describe may strengthen our confidence in the DDH-Assumption and in the security of its many applications. Additional evidence to the validity of the DDH-Assumption lies in the fact that it endured the extensive research of the related CDH-Assumption. To some extent, the DDH-Assumption is also supported by the results on the strength of the CDH-Assumption in several groups [12, 39, 40, 49] and by additional results [12, 17, 50]. For instance, Shoup [50] showed that the DDH-Problem is hard for any "generic" algorithm. However, a main conclusion of this paper is that the DDH-Assumption deserve more attention since it implies the security of many attractive cryptographic constructions.

[1] While this powerful reduction uses the usual methods of this area, for some reason it was overlooked.

An obvious application of the DDH-Assumption is to the Diffie-Hellman key-exchange protocol and to the related public-key cryptosystem [21]. In the ElGamal cryptosystem, given the public key g^a, the encryption of a message m is $\langle g^b, g^{a \cdot b} \cdot m \rangle$. In Section 3 (of the full version), we show how to adjust this cryptosystem in order to get a probabilistic encryption-scheme whose semantic security (see [30]) is equivalent to the DDH-Assumption[2]. The price of encrypting many bits using the ElGamal cryptosystem is a single (or two) exponentiation. This is comparable with the Blum-Goldwasser cryptosystem [9]. Other applications that previously appeared are [4, 13, 16, 22, 51].

To previous applications one can add a p.r. generator [10, 53] that practically doubles the input length and a p.r. synthesizer (see definition in [41]) whose output length is similar to its arguments length. Essentially, the generator is defined by $G_{P,g,g^a}(b) = \langle g^b, g^{a \cdot b} \rangle$ and the synthesizer by $S_{P,g}(a,b) = g^{a \cdot b}$. Both these constructions are overshadowed by our new construction of a very efficient family of p.r. functions. The p.r. function is defined by $n+1$ values, $\langle a_0, a_1 \ldots a_n \rangle$, chosen uniformly in $\mathbb{Z}_Q$. The value of the function on an n-bit input $x = x_1 x_2 \cdots x_n$ is essentially

$$f_{P,g,a_0,a_1 \ldots a_n}(x) \stackrel{\text{def}}{=} (g^{a_0})^{\prod_{x_i=1} a_i} \bmod P$$

For some applications, we still need to hash this value as described in Section 4.1. Note that, after appropriate preprocessing, the computation required consists of two multiple products. This can be done in TC^0 (see Section 4.3). The simple structure of the functions gives several attractive properties, as described in Section 6.

The GDH-Assumption

In Section 5, we suggest a related construction of p.r. functions that is based on the (computational) GDH-Assumption. This generalization of the DH-Assumption was previously considered in the context of a key-exchange protocol for a group of parties (see e.g., [49, 51]). The GDH-Assumption is implied by the DDH-Assumption (as shown in [51] and in this paper) but the assumptions are not known to be equivalent. In addition, the GDH-Assumption modulo a composite is not stronger than the assumption that factoring is hard (see [6, 49]). This implies an attractive construction of p.r. functions such that their security can be reduced to factoring:

Let N be distributed over Blum-integers ($N = P \cdot Q$ where P and Q are primes and $P = Q = 3 \bmod 4$) and assume that (under this distribution) it is hard to factor N. Let g be a uniformly distributed quadratic residue in Z_N^*, let $\vec{a} = \langle a_{1,0}, a_{1,1}, a_{2,0}, a_{2,1}, \ldots a_{n,0}, a_{n,1} \rangle$

[2] The semantic security of the original cryptosystem is equivalent to the DDH-Assumption only when the message space is restricted to the subgroup generated by g.

be a uniformly distributed sequence of $2n$ elements in $[N] \stackrel{\text{def}}{=} \{1, 2, \ldots, N\}$ and let r be a uniformly distributed bit-string of the same length as N. Then, the Binary-function, $f_{N,g,\bar{a},r}$, is p.r. Where, the value of $f_{N,g,\bar{a},r}$ on any n-bit input, $x = x_1 x_2 \cdots x_n$, is defined by:

$$f_{N,g,\bar{a},r}(x) \stackrel{\text{def}}{=} \left(g^{\prod_{i=1}^{n} a_{i,x_i}} \bmod N\right) \odot r$$

($\odot$ denotes the inner product mod 2).

Previous Work

In addition to introducing p.r. functions, Goldreich, Goldwasser and Micali [27] provided a construction of such functions (GGM-Construction) based on p.r. generators. Naor and Reingold [41] recently showed a *parallel* construction based on a new primitive called a p.r. synthesizer. Under concrete intractability assumptions like "factoring is hard" this construction gives pseudo-functions in TC^1. Our work is motivated by [41] both in the task of reducing the depth of the p.r. functions and in the construction itself (see Section 5). Parallel constructions of other cryptographic primitives were provided by Impagliazzo and Naor [32], based on the hardness of subset sum (and factoring), and by Blum et. al. [5], based on hard-to-learn problems.

The construction of this paper is not only more parallelizable than the concrete constructions based on [27, 41], but it is also much more efficient. Though this construction seems very different than the constructions of [27, 41], we were able to relate the proof of security of this construction to both [27] and [41] (see Sections 4.2 and 5).

It turns out that there are a number of researchers who observed that the average-case DDH-Assumption yields p.r. *generators* with good expansion. One such construction was proposed by Rackoff (unpublished). A different construction is suggested by Gertner and Malkin [23]. This construction is similar to the p.r. generator one gets by scaling down our p.r. functions.

Organization and Omissions

The full version of this paper is organized as follows: In Section 2.1 we describe the notation and conventions used in this paper. In Section 2.2 we describe some applications and constructions of p.r. functions and the motivation for our construction. In Section 3 we further consider the DDH-Assumption and show a simple randomized reduction between its worst-case and average-case. In Section 4 we describe a construction of p.r. functions based on the DDH-Assumption, prove its security and consider its complexity. In Section 5 we define the CDH-Assumption and show a related construction of p.r. functions based on this assumption. In Section 6 we consider some of the possible features of p.r. functions. In Section 7 we suggest directions for further research.

Given the limitation on the length of this extended-abstract, we omit in this version most of the proofs, most of Sections 2.2, 5 & 6 and a significant portion of Section 3.

2 Preliminaries

2.1 Notation and Conventions

- For any integer N the multiplicative group modulo N is denoted by $\mathbb{Z}_N^*$ and the additive group modulo N is denoted by $\mathbb{Z}_N$.
- For any integer k, denote by $[k]$ the set of integers $= \{1, 2, \ldots, k\}$.
- For any two bit-strings of the same length, x and y, the inner product mod 2 of x and y is denoted by $x \odot y$.
- All exponentiations in this paper are in $\mathbb{Z}_P^*$ (the definition of P will be clear by the context).

2.2 P.R. Functions

As mentioned in the introduction, our main result is a construction of a p.r. function that is *efficient, shallow and simple*. In the full version, we devote this section to motivating such a construction and to describing previous constructions and applications of p.r. functions. We also include the formal definition of p.r. functions (almost as it appears in [26]). In this version, we omit almost all the subsection (apart of the motivation for having a shallow construction). Good references on p.r. functions and pseudo-randomness in general are Goldreich [26] and Luby [37].

The constructions of this paper gives p.r. functions computable in TC^0 (given appropriate preprocessing). We briefly summarize part of the discussion that appears in [41] on the applications of parallel p.r. functions:

- It is likely that p.r. functions will be implemented in hardware (as is the case for DES). In such implementations, having shallow p.r. functions imply reduced latency in computing those functions which, for some applications (as the encryption of messages on a network), is essential.
- As was observed by Valiant [52], if a concept class contains p.r. functions then it cannot be learned: There exists a distribution of concepts, computable in this class, that is hard for *every* efficient learning algorithms, for *every* "non-trivial" distribution on the instances *even* when membership-queries are allowed. Notice that the unlearnability result implied by the existence of p.r. functions is very strong. Weaker unlearnability results for NC^1 and TC^0, based on other cryptographic assumptions, were obtained in [1, 34, 35]. It is also interesting to compare

with the result of Linial, Mansour and Nisan [36] who showed that AC^0 *can* be efficiently learned.

- Another application of p.r. functions in complexity was suggested by Razborov and Rudich [44]. They showed that if a circuit-class contains p.r. functions (that are secure against a subexponential-time adversary), then there are no, what they called, *Natural Proofs* (which include all known lower bound techniques) for separating this class from $P/poly$. We therefore get from our construction that *if the GDH-Assumption holds against a subexponential-time adversary (and in particular if factoring is hard) then there are no Natural Proofs for separating TC^0 from $P/poly$.*

  We note that one can extract a similar result (assuming the hardness of factoring) from the work of Kharitonov [34], which is based on the p.r. generator of Blum, Blum and Shub [7].

3 The Decisional DH-Assumption

As mentioned in the introduction, the DH-Assumption was introduced in the context of the Diffie and Hellman [20] key-exchange protocol. Informally, a key-exchange protocol is a way for two parties, $\mathcal{A}$ and $\mathcal{B}$, to agree on a common key, $K_{\mathcal{A},\mathcal{B}}$, while communicating over an insecure (but authenticated) channel. Such a protocol is secure if any efficient third party, $\mathcal{C}$, with access to the communication between $\mathcal{A}$ and $\mathcal{B}$ (but not to their private random strings) cannot tell apart $K_{\mathcal{A},\mathcal{B}}$ from a random value (i.e., $K_{\mathcal{A},\mathcal{B}}$ is p.r. to $\mathcal{C}$). This guarantees that it is computationally infeasible for an eavesdropper to gain "any" partial information on $K_{\mathcal{A},\mathcal{B}}$.

Given a large prime P and a generator g of $\mathbb{Z}_P^*$, the Diffie-Hellman key-exchange protocol goes as follows: $\mathcal{A}$ chooses an integer a uniformly at random in $[P-2]$ and sends g^a to $\mathcal{B}$.[3] In return $\mathcal{B}$ chooses an integer b uniformly at random in $[P-2]$ and sends g^b to $\mathcal{A}$. Both $\mathcal{A}$ and $\mathcal{B}$ can now compute $g^{a\cdot b}$ and their common key, $K_{\mathcal{A},\mathcal{B}}$, is defined by $g^{a\cdot b}$ in some public way. For this protocol to be secure we must have, at the minimum, that the CDH-Assumption holds:

Given $\langle g, g^a, g^b\rangle$, it is hard to compute $g^{a\cdot b}$.

The reason is that if this assumption does not hold then, $\mathcal{C}$ (as above) can also compute $K_{\mathcal{A},\mathcal{B}}$.

One method to produce the key, $K_{\mathcal{A},\mathcal{B}}$, is to apply the Goldreich-Levin [29] hard-core function[4] to $g^{a\cdot b}$ (an important improvement on the security of such an application was made by Shoup [50]). If the CDH-Assumption holds, then this method indeed gives a p.r. key. However, the proof in [29] only implies the pseudo-randomness of the key in case its length is at most logarithmic in the security parameter. A much more ambitious method is to take $g^{a\cdot b}$ itself as the key. For instance, in the ElGamal cryptosystem, given the public key g^a the encryption of a message m is $\langle g^b, g^{a\cdot b}\cdot m\rangle$. The security of the key-exchange protocol now relies on the DDH-Assumption:

Given $\langle g, g^a, g^b, z\rangle$, it is hard to decide whether or not $z = g^{a\cdot b}$.

However, when g is a generator of $\mathbb{Z}_P^*$, we have that g^a and g^b do give some information on $g^{a\cdot b}$. For example, if either g^a or g^b is a quadratic residue then so is $g^{a\cdot b}$. A standard solution for this problem is to take g to be a generator of the subgroup of $\mathbb{Z}_P^*$ of order Q, where Q is a large prime divisor of $P-1$. In fact, for most applications, taking g to be an element of order Q is an advantage. This is the case since all known subexponential algorithms for computing the discrete-log will be subexponential in the length of P even when applied to the subgroup of size Q generated by g (the best known algorithm for general groups has time square root of the size of the group). Therefore, we can use a rather small prime Q (say, 160 bits long) which results in a substantial improvement in efficiency.

For more on the validity of the DDH-Assumption and on its different applications, see the full version of this paper.

3.1 Formal Definition

To formalize the DDH-Assumption, we first need to specify an efficiently samplable distribution for P, Q and g (where g is an element of order Q in $\mathbb{Z}_P^*$).

Let n be the security parameter, for some function $\ell : \mathbb{N} \mapsto \mathbb{N}$ we want to choose an n-bit prime P with an $\ell(n)$-bit prime Q that divides $P-1$. A natural way to do this is to choose P and Q uniformly at random subject to those constraints. However, it is possible to consider different distributions. For example, it is not inconceivable that the assumption holds when for every n we have a *single* possible choice of P, Q and g. Another common example is letting P and Q satisfy $P = 2\cdot Q+1$ (although choosing a smaller Q may increase the efficiency of most applications). In order to keep our results general, we let P, Q and g be generated by *some* polynomial-time algorithm IG (where IG stands for instance generator).

Definition 3.1 (IG) *The Diffie-Hellman instance generator, IG, is a probabilistic polynomial-time algorithm such that on input 1^n the output of IG is distributed over triplets $\langle P, Q, g\rangle$ where, P is an n-bit prime, Q a (large) prime divisor of $P-1$ and g an element of order Q in $\mathbb{Z}_P^*$.*

[3] Recall that all exponentiations here and in the rest of the paper are in $\mathbb{Z}_P^*$ (the definition of P will be clear by the context).

[4] For example, to get a key of one bit, we can define $K_{\mathcal{A},\mathcal{B}}$ to be the inner product mod 2 of $g^{a\cdot b}$ and a random string r (chosen by one of the parties and sent to the other over the insecure channel).

For the various applications of the DDH-Assumption we need its average-case version. Namely, when a and b are uniformly chosen and c is either $a \cdot b$ or uniformly chosen. In Section 3.2, it is shown that a worst-case choice of a, b and c can be reduced to a uniform choice. Similarly, the assumption is not weakened if g (generated by IG) is taken to be a uniformly chosen element of order Q in $\mathbb{Z}_P^*$.

Assumption 3.1 (Decisional Diffie-Hellman) *For every probabilistic polynomial-time algorithm $\mathcal{A}$, every constant $\alpha > 0$ and all sufficiently large n's*

$$\begin{aligned} &\Big| \Pr[\mathcal{A}(P,Q,g,g^a,g^b,g^{a\cdot b}) = 1] \\ &- \Pr[\mathcal{A}(P,Q,g,g^a,g^b,g^c) = 1] \Big| \\ &< \frac{1}{n^\alpha} \end{aligned}$$

where the probabilities are taken over the random bits of $\mathcal{A}$, the choice of $\langle P,Q,g\rangle$ according to the distribution $IG(1^n)$ and the choice of a, b and c uniformly *at random in $\mathbb{Z}_Q$.*

3.2 A Randomized Reduction

In this subsection we use a simple randomized reduction to show that for every P, Q and g the DDH-Problem is either very hard on the average or very easy in the worst-case. Given the current knowledge of the DDH-Problem, such a result strengthens our belief in the DDH-Assumption.

Definition 3.2 *For any $\langle P,Q,g\rangle$ such that P is a prime, Q a prime divisor of $P-1$ and g an element of order Q in $\mathbb{Z}_P^*$ the function $DDH_{\langle P,Q,g\rangle}$ is defined by*

$$DDH_{P,Q,g}(g^a,g^b,g^c) = \begin{cases} 1 & \text{if } c = a\cdot b \bmod Q \\ 0 & \text{otherwise} \end{cases}$$

for any three elements a, b, c in $\mathbb{Z}_Q$.

Theorem 3.1 *Let $\mathcal{A}$ be any probabilistic algorithm with running time $t = t(n)$ and $\epsilon = \epsilon(n)$ any positive function such that $1/\epsilon$ is efficiently constructible. There exists a polynomial $p = p(n)$ and a probabilistic algorithm $\mathcal{A}'$ with running time $(t(n)\cdot p(n))/\epsilon(n)$ such that for* any *choice of $\langle P,Q,g\rangle$ as in Definition 3.2 if:*

$$\begin{aligned} &\Big| \Pr[\mathcal{A}(P,Q,g,g^a,g^b,g^{a\cdot b}) = 1] \\ &- \Pr[\mathcal{A}(P,Q,g,g^a,g^b,g^c) = 1] \Big| \\ &> \epsilon(n) \end{aligned}$$

where the probabilities are taken over the random bits of $\mathcal{A}$ and the choice of a, b and c uniformly *at random in $\mathbb{Z}_Q$, then, for* any *a, b and c in $\mathbb{Z}_Q$:*

$$\Pr[\mathcal{A}'(P,Q,g,g^a,g^b,g^c) \neq DDH_{P,Q,g}(g^a,g^b,g^c)]| < 2^{-n}$$

where the probability is only *over the random bits of $\mathcal{A}'$.*

In particular, if $\mathcal{A}$ is probabilistic polynomial-time and $\epsilon(n) \geq 1/poly(n)$ then $\mathcal{A}'$ is also probabilistic polynomial-time.

Blum and Micali [10] introduced the concept of random-self-reducibility (and randomized reductions). Informally, a problem is random-self-reducible if solving the problem on *any* instance x can be efficiently reduced to solving the problem on a random instance y. I.e., for any instance x, a random instance y can be efficiently sampled using a random string r such that given r and the solution of the problem on y it is easy to compute the solution of the problem on x. A problem that is random-self-reducible can either be efficiently solved for every instance with overwhelming success probability or it cannot be solved for a random instance with non-negligible success probability.

Our randomized reduction is closely related to other known reductions. Blum and Micali [10] showed that for any specific prime P and generator g, the discrete-log problem is random-self-reducible: given $\langle P,g,g^a\rangle$ for *any* a it is easy to generate a random instance $\langle P,g,g^{a+r} = g^a\cdot g^r\rangle$ (where r is uniform in $[P-1]$). Given the solution for the random instance (i.e., $a+r$) it is easy to compute the solution for the original instance (i.e., a). A similar property was shown for the CDH-Problem (e.g. Maurer [39]): given $\langle P,g,g^a,g^b\rangle$ for *any* a and b it is easy to generate a random instance $\langle P,g,g^{a+r},g^{b+s}\rangle$ (where r and s are uniform in $[P-1]$). Given the solution for the random instance (i.e., $z = g^{(a+r)\cdot(b+s)}$) it is easy to compute the solution for the original instance (i.e., $g^{a\cdot b} = z\cdot(g^a)^{-s}\cdot(g^b)^{-r}\cdot g^{-s\cdot r}$).

However, in order to prove Theorem 3.1, we need a somewhat different reduction. In particular, we need to use the fact that g is an element of prime order (Theorem 3.1 is not true when g is a generator of $\mathbb{Z}_P^*$).

Lemma 3.2 *There exists a probabilistic polynomial-time algorithm, $\mathcal{R}$ such that on* any *input*

$$\langle P,Q,g,g^a,g^b,g^c\rangle$$

where, P is a prime, Q a prime divisor of $P-1$, g an element of order Q in $\mathbb{Z}_P^*$ and a, b, c are three elements in $\mathbb{Z}_Q$ the output of $\mathcal{R}$ is:

$$\langle P,Q,g,g^{a'},g^{b'},g^{c'}\rangle$$

where if $c = a\cdot b \bmod Q$ then a' and b' are uniform in $\mathbb{Z}_Q$ and $c' = a'\cdot b' \bmod Q$ and if $c \neq a\cdot b \bmod Q$ then a', b' and c' are all *uniform in $\mathbb{Z}_Q$.*

Proof. $\mathcal{R}$ chooses s_1, s_2 and r uniformly in $\mathbb{Z}_Q$, computes

$$\begin{aligned} &g^{a'} = (g^a)^r\cdot g^{s_1},\ g^{b'} = g^b\cdot g^{s_2} \\ \&\quad &g^{c'} = (g^c)^r\cdot(g^a)^{r\cdot s_2}\cdot(g^b)^{s_1}\cdot g^{s_1\cdot s_2} \end{aligned}$$

and outputs

$$\langle P, Q, g, g^{a'}, g^{b'}, g^{c'} \rangle$$

Let $c = a \cdot b + e \bmod Q$ for e in $\mathbb{Z}_Q$ then $a' = r \cdot a + s_1 \bmod Q$, $b' = b + s_2 \bmod Q$ and $c' = a'b' + e \cdot r \bmod Q$. If $e = 0$ we get that a' and b' are uniformly distributed in $\mathbb{Z}_Q$ and $c' = a' \cdot b' \bmod Q$. If $e \neq 0$ we get that a', b' and c' are all uniformly distributed in $\mathbb{Z}_Q$ (this is the place we use the fact that Q is a prime which implies that $e \cdot r$ is a uniformly distributed element). Therefore, the output of $\mathcal{R}$ has the desired distribution. □

4 Construction of P.R. Functions

In this section we describe a construction of p.r. functions based on the DDH-Assumption, prove its security and consider its complexity. A related construction is described in Section 5.

4.1 Construction and Main Result

Construction 4.1 *We define the function ensemble $F = \{F_n\}_{n \in \mathbb{N}}$. For every n, a key of a function in F_n is a tuple, $\langle P, Q, g, \vec{a} \rangle$, where P is an n-bit prime, Q a prime divisor of $P - 1$, g an element of order Q in $\mathbb{Z}_P^*$ and $\vec{a} = \langle a_0, a_1, \ldots a_n \rangle$ a sequence of $n + 1$ elements of $\mathbb{Z}_Q$. For any n-bit input, $x = x_1 x_2 \cdots x_n$, the function $f_{P,Q,g,\vec{a}}$ is defined by:*

$$f_{P,Q,g,\vec{a}}(x) \stackrel{\text{def}}{=} (g^{a_0}) \prod_{x_i=1} {}^{a_i}$$

The distribution of functions in F_n is induced by the following distribution on their keys: $\vec{a}$ is uniform in its range and the distribution of $\langle P, Q, g \rangle$ is $IG(1^n)$.

F obtains the following pseudo-randomness property:

Theorem 4.1 *Let $F = \{F_n\}_{n \in \mathbb{N}}$ be as in Construction 4.1. If the DDH-Assumption (Assumption 3.1) holds, then for every probabilistic polynomial-time oracle machine $\mathcal{M}$, every constant $\alpha > 0$, and all sufficiently large n's*

$$\begin{aligned} &\Big| \Pr[\mathcal{M}^{f_{P,Q,g,\vec{a}}}(P, Q, g) = 1] \\ &\quad - \Pr[\mathcal{M}^{R_{P,Q,g}}(P, Q, g) = 1] \Big| \\ &< \frac{1}{n^\alpha} \end{aligned}$$

where in the first probability $f_{P,Q,g,\vec{a}}$ is distributed according to F_n and in the second probability, the distribution of $\langle P, Q, g \rangle$ is $IG(1^n)$ and $R_{P,Q,g}$ is uniformly chosen in the set of functions with domain $\{0,1\}^n$ and range $\langle g \rangle$ (the subgroup of $\mathbb{Z}_P^*$ generated by g).

Remark 4.1 *It is easy to verify from the proof of Theorem 4.1 that the following, more quantitative, version of the theorem also holds:*

Assume that there exists a probabilistic oracle machine with running time $t = t(n)$ that makes at most $m = m(n)$ queries and distinguishes $f_{P,Q,g,\vec{a}}$ from $R_{P,Q,g}$ (as in Theorem 4.1) with advantage $\epsilon = \epsilon(n)$. Then, there exists a probabilistic algorithm with running time $poly(n) \cdot t(n)$ that breaks the DDH-Assumption with advantage $\epsilon(n)/(n \cdot m(n))$.

Given Theorem 4.1, we have that F is "almost" an efficiently computable p.r. function ensemble. There is one difference: A function $f_{P,Q,g,\vec{a}}$ in F_n has domain $\{0,1\}^n$ and range $\langle g \rangle$. Therefore, different functions in F_n have different ranges which deviates from the standard definition of p.r. functions. However, for many applications of p.r. functions this deviation does not set a problem (e.g., the applications of p.r. functions to private-key authentication and identification and their applications to digital signatures [3]). In addition, it is rather easy to construct from F p.r. functions under the standard definition. In order to show this, we need the following lemma which is a simple corollary of the leftover hash lemma [31, 33]:

Lemma 4.2 *Let n, ℓ and e be three positive integers such that $3e < \ell < n$. Let $X \subseteq \{0,1\}^n$ be a set of at least 2^ℓ elements and x uniformly distributed in X. Let H be a family of pair-wise independent, $\{0,1\}^n \mapsto \{0,1\}^{\ell - 3e}$, hash functions. Then, for all but a 2^{-e} fraction of $h \in H$ the uniform distribution over $\{0,1\}^{\ell-3e}$ and $h(x)$ are of statistical distance of at most 2^{-e}.*

Lemma 4.2, suggests the following construction:

Construction 4.2 *Let $\ell = \ell(n)$ be an integer-valued function such that for any output, $\langle P, Q, g \rangle$, of $IG(1^n)$ we have that $Q > 2^{\ell(n)}$. Let $F = \{F_n\}_{n \in \mathbb{N}}$ be as in Construction 4.1 and $\forall n$, let H_n be a family of pair-wise independent, $\{0,1\}^n \mapsto \{0,1\}^{\ell(n)/2}$, hash functions. We define the function ensemble $\tilde{F} = \{\tilde{F}_n\}_{n \in \mathbb{N}}$. For every n, a key of a function in $\tilde{F}_n$ is a pair, $\langle k, h \rangle$, where k is a key of a function in F_n and $h \in H_n$. For any n-bit input, x, the function $\tilde{f}_{k,h}$ is defined by:*

$$\tilde{f}_{k,h}(x) \stackrel{\text{def}}{=} h(f_k(x))$$

The distribution of functions in $\tilde{F}_n$ is induced by the following distribution on their keys: h is uniform in H_n and the distribution of k is the same as the distribution of keys in F_n.

Note that choosing the range of the hash functions to be $\{0,1\}^{\ell(n)/2}$ is arbitrary. One can choose the range to be $\{0,1\}^{\ell(n)-e(n)}$ for any function $e(n)$ such that $2^{-e(n)/3}$ is negligible. Using Theorem 4.1 and Lemma 4.2 we can easily conclude:

Theorem 4.3 *If the DDH-Assumption holds, then $\tilde{F} = \{\tilde{F}_n\}_{n \in \mathbb{N}}$ (as in Construction 4.2) is an efficiently computable p.r. function ensemble.*

Remark 4.2 *From Theorem 4.1 and Lemma 4.2 it follows that $\tilde{F} = \{\tilde{F}_n\}_{n\in\mathbb{N}}$ remains indistinguishable from the uniform function-ensemble even when the distinguisher has access to $\langle P, Q, g\rangle$ and to h (as in the definition of functions in $\tilde{F}_n$).*

4.2 Proof of Security

The proof of Theorem 4.1 bares a close resemblance to the proof of security for the GGM-Construction of p.r. functions [27]. This may seem surprising since, in our construction, no tree is described and the order of the input bits does not really matter. However, in some sense, one may view our construction as a careful application (or a generalization) of the GGM-Construction. In the following few paragraphs we describe the resemblance and differences between the two constructions. However, since we cannot get Theorem 4.1 as a corollary of the GGM-Theorem, we include (in the full version) a complete proof. We note that there is also a close relation between the construction of this section and the construction of [41] (which indeed motivated our construction as described in Section 5).

Let G be a p.r. generator that doubles its input. Define G^0 and G^1 such that for any n-bit string x, both $G^0(x)$ and $G^1(x)$ are n-bit strings and $G(x) = \langle G^0(x), G^1(x)\rangle$. Under the GGM-Construction, the key of a p.r. function $f_s : \{0,1\}^n \mapsto \{0,1\}^n$ is a uniformly chosen n-bit string, s. For any n-bit input, $x = x_1x_2\cdots x_n$, the function f_s is defined by:

$$f_s(x) \stackrel{\text{def}}{=} G^{x_n}(\cdots(G^{x_2}(G^{x_1}(s)))\cdots)$$

The DDH-Assumption implies a simple p.r. generator that practically doubles its input: $G_{P,g,g^a}(b) \stackrel{\text{def}}{=} \langle g^b, g^{a\cdot b}\rangle$ (whose output is a p.r. pair of values in the subgroup generated by g) . It is tempting to use this generator for the GGM-Construction. However, a straightforward application of the GGM-Construction would give a rather inefficient function. We therefore suggest a slight change to the definition of the generator:

$$G_{P,g,g^a}(g^b) = \langle G^0_{P,g,g^a}(g^b), G^1_{P,g,g^a}(g^b)\rangle \stackrel{\text{def}}{=} \langle g^b, g^{a\cdot b}\rangle$$

At a first look this seems absurd: G_{P,g,g^a} is not efficiently computable unless the DH-Problem is easy. Therefore, if G_{P,g,g^a} is efficiently computable then it is not p.r. However, G_{P,g,g^a} has the following property that allows us to use a generalization of the GGM-Construction: $G_{P,g,g^a}(g^b)$ is efficiently computable if either a or b are known. A more general way to state this is:

1. G_{P,g,g^a} is efficiently computable (on any input), given the random bits that were used to sample it (in particular, given a).
2. For any G_{P,g,g^a}, it is easy to generate the distribution of its output, $G_{P,g,g^a}(g^b)$, on a uniformly chosen input, g^b (we use this fact in the proof of Lemma 4.4).

We now get the p.r. functions of Construction 4.1, using the GGM-Construction where at each level of the construction we use a different value, g^a, for the generator:

$$f_{P,g,a_0,a_1,\ldots,a_n}(x) \stackrel{\text{def}}{=} G^{x_n}_{P,g,g^{a_n}}(\cdots(G^{x_1}_{P,g,g^{a_1}}(g^{a_0}))\cdots)$$

The formal proof of Theorem 4.1 requires the following lemma:

Definition 4.1 *Let n and t be any pair of positive integers. For $0 \le i \le t$ define the i^{th} hybrid distributions $I_i^{n,t}$ to be:*

$$\langle P, Q, g, g^a, g^{b_1}, g^{a\cdot b_1}, \ldots, g^{b_i}, g^{a\cdot b_i}, g^{b_{i+1}}, g^{c_{i+1}}, \ldots, g^{b_t}, g^{c_t}\rangle$$

where $\langle P, Q, g\rangle$ is distributed according to $IG(1^n)$ and all the values in $\langle a, b_1, \ldots, b_t, c_{i+1}, \ldots, c_t\rangle$ are uniform in $\mathbb{Z}_Q$.

Lemma 4.4 *(Indistinguishability of a Polynomial Sample) If the DDH-Assumption holds, then for every probabilistic polynomial-time algorithm $\mathcal{D}$, every polynomial $t(\cdot)$, every constant $\alpha > 0$ and all sufficiently large n's*

$$\left|\Pr[\mathcal{D}(I^{n,t(n)}_{t(n)}) = 1] - \Pr[\mathcal{D}(I^{n,t(n)}_0) = 1]\right| < \frac{1}{n^\alpha}$$

We omit the proof of this lemma and of Theorem 4.1.

4.3 Efficiency of the P.R. Functions

Consider a function $f_{P,Q,g,\vec{a}} \in F_n$ (where $\vec{a} = \langle a_0, a_1, \ldots a_n\rangle$) as in Construction 4.1. Computing the value of this function at any given point, x, involves one multiple product, $y = a_0 \cdot \prod_{x_i=1} a_i$ (which can be performed modulo Q), and one modular exponentiation, g^y. This gives a p.r. function which is much more efficient than previous constructions. Furthermore, one can use preprocessing in order to get improved efficiency. The most obvious preprocessing is computing the values g^{2^i} (for every positive integer i up to the length of Q). Now computing the value of the function requires two multiple products modulo a prime[5]. Additional preprocessing can reduce the work by a factor of $O(\log n)$ (see Brickell et. al. [15]). Actually, in order to get the value of the p.r. function as in Construction 4.2, we also need one application of a pair-wise independent hash function but this operation is much cheaper than a multiple product or a modular exponentiation.

As described in the Introduction and in Section 2.2, we are also interested in finding the parallel complexity of the p.r. functions. In order to do so, let us first recall

[5] In the case that Q is much smaller than P we have that the first multiple product is much cheaper than the second

the result of Beame, Cook and Hoover [2], who showed that devision and related operations *including multiple product* are computable in NC^1. Based on this result Reif and Tate [45, 46] showed that these operations are also computable in TC^0. The exact depth required for these operations was considered in [47, 48] were it was shown that multiple sum is in TC^0_2, multiplication and division in TC^0_3 and multiple product in TC^0_4.

By the results above, we get that after preprocessing (i.e., computing the values g^{2^i}), it is possible to evaluate the function $f_{P,Q,g,\vec{a}}$ in TC^0. A more detailed analysis reveals that some further optimizations in the depth are possible: Using additional preprocessing, this function is in fact in TC^0_5. We defer this detailed analysis to the final version of this paper and only briefly comment on a few facts regarding $f_{P,Q,g,\vec{a}}$ that allow us to reduce the depth (compared with a naive application of [47, 48]). First, note that in both multiple products we can assume any preprocessing of the values in the multiplication (since these values are taken from the sequence $\langle a_0, a_1, \ldots a_n \rangle$ or from the set $\{g^{2^i}\}$). Second, we don't need the actual value of the first multiple product, $y = \prod_{x_i=1} a_i$: Computing values r_i (obtained by the CRT-representation) for which $y = \sum m_i \cdot r_i$ (where the values m_i are known in advance and can be preprocessed) is just as good. Finally, the value P is also known in advance. Therefore, the depth of the final modular reduction can be reduced by precomputing the values $2^i \bmod P$. In conclusion:

Theorem 4.5 *Let $F = \{F_n\}_{n\in\mathbb{N}}$ be as in Construction 4.1. Then, there exists a polynomial, $p(\cdot)$, such that for every $n \in \mathbb{N}$ and every function $f_k \in F_n$ there exists a depth 5 threshold circuit of size bounded by $p(n)$ that computes f_k.*

Remark 4.3 *The same analysis hold for efficiency and depth of the p.r. functions of Section 5.*

5 A Related Construction

In the full version, we show an additional construction of p.r. functions – Construction 5.1, that is very similar to Construction 4.2. The security of Construction 5.1 is reduced to the GDH-Assumption which is a generalization of the *computational* DH-Assumption. We include this construction for two main reasons:

1. The GDH-Assumption is implied by the DDH-Assumption but they are not known to be equivalent. Therefore, Construction 5.1 may still be valid even if the DDH-Assumption does not hold. In addition, the GDH-Assumption modulo a composite is not stronger than the assumption that factoring is hard. This gives an attractive construction of p.r. functions that is at least as secure as factoring.
2. Construction 5.1 is based on a somewhat different methodology than Construction 4.2. It may be easier to apply this methodology in order to construct p.r. functions based on additional assumptions (in fact, Construction 4.2 was obtained as a modification of Construction 5.1).

The motivation for Construction 5.1 lies in the concept of p.r. synthesizers and the construction of p.r. functions using p.r. synthesizers as building blocks [41]. Informally, a p.r. synthesizer, S, is:

> *An efficiently computable function of two arguments such that given polynomially-many, uniformly-chosen, inputs for each argument, $\{x_i\}_{i=1}^m$ and $\{y_i\}_{i=1}^m$, the output of S on all the combinations, $(S(x_i, y_j))_{i,j=1}^m$, cannot be efficiently distinguished from uniform.*

A natural generalization is a k-dimensional p.r. synthesizer. Informally, a k-dimensional p.r. synthesizer, S, may be defined to be:

> *An efficiently computable function of k arguments such that given polynomially-many, uniformly-chosen, inputs for each argument, $\left\{\left\{x_i^j\right\}_{i=1}^m\right\}_{j=1}^k$, the output of S on all combinations, $M = \left(S(x_{i_1}^1, x_{i_2}^2, \ldots, x_{i_k}^k)\right)_{i_1,i_2,\ldots,i_k=1}^m$, cannot be efficiently distinguished from uniform by an algorithm that can access M at points of its choice.*

The construction of [41] can be viewed as first recursively applying a 2-dimensional synthesizer to get an n-dimensional synthesizer, S, and then defining the p.r. function, f, by:

$$\begin{aligned} & f_{\langle a_{1,0},a_{1,1},a_{2,0},a_{2,1},\ldots a_{n,0},a_{n,1}\rangle}(\sigma_1\sigma_2\ldots\sigma_n) \\ \stackrel{\text{def}}{=} \; & S(a_{1,\sigma_1}, a_{2,\sigma_2}, \ldots, a_{n,\sigma_n}) \end{aligned}$$

However, using this construction, the depth of the n-dimensional synthesizer (and the p.r. functions) is larger by a logarithmic factor than the depth of the 2-dimensional synthesizer. Therefore, a natural problem is to come up with a direct construction of an n-dimensional synthesizer.

In the full version, it is shown that, under the GDH-Assumption, the function, $S_{P,Q,g,r}$, defined by $S_{P,Q,g,r}(a_1, a_2, \ldots, a_n) \stackrel{\text{def}}{=} \left(g^{\prod_{i=1}^n a_i}\right) \odot r$, is an n-dimensional synthesizer. Construction 5.1 is then obtained as described above.

6 Additional Properties

The p.r. functions of Construction 4.2 and of Section 5 have a simple algebraic structure. We consider this to

be an important advantage over all previous constructions, mainly since several attractive features seem more likely to exist for a simple construction of p.r. functions. An interesting example arouse by the work of Bellare and Goldwasser [3]. They suggested a way to design a digital-signature scheme which would be very attractive given efficient p.r. functions and an efficient *non-interactive zero-knowledge proof* for claims of the form $y = f_s(m)$ (when a commitment to a key, s, of a p.r. function, f_s, is available as part of the public-key).

In the full version, we suggest preliminary designs for several protocols: (1) A rather simple zero-knowledge proof for claims of the form $y = f_s(m)$ and $y \neq f_s(m)$. (2) A way to distribute a p.r. function among a set of parties such that only an authorized subset can compute the value of the function at any given point. (3) A protocol for "blind computation" of the value of the function: Assume that a party, $\mathcal{A}$, knows a key, s, of a p.r. function. Then $\mathcal{A}$ and a second party, $\mathcal{B}$, can perform a protocol during which $\mathcal{B}$ learns exactly one value $f_s(x)$ of its choice whereas $\mathcal{A}$ does not learn a thing (and, in particular, does not learn x).

These designs are probably not efficient enough in practice but still they serve as a demonstration to the potential of our construction. We hope this work will stimulate further research both in improving these designs and in suggesting designs for other protocols (as the non-interactive zero-knowledge proof mentioned above and a function-sharing scheme for p.r. functions).

7 Further Research

This paper shows two, very efficient, constructions of p.r. functions. The first construction is based on the *decisional* DH-Assumption and the second construction is based on a generalization of the *computational* DH-Assumption. Therefore, a natural line for further research is the study of the validity of these assumptions and the relations between these assumptions and the standard *computational* DH-Assumption. Since our constructions can be based on the corresponding assumptions for other groups (e.g., in elliptic-curve groups), it is interesting to study the validity of these assumptions as well.

In Section 5, the concept of a k-dimensional p.r. synthesizer and the immediate construction of p.r. functions from n-dimensional synthesizers are described. The GDH-Assumption gives a simple construction of an n-dimensional synthesizer which indeed translates to a construction of p.r. functions. An interesting problem is to construct efficient n-dimensional synthesizers using other intractability assumptions.

Another interesting line for further research is improving the protocols described in Section 6 and designing additional protocols (as a non-interactive zero-knowledge proof for the value of a p.r. function and a function-sharing scheme).

Acknowledgments

We thank Ran Canetti, Kobbi Nissim and Amnon Ta-Shma for many helpful discussions.

References

[1] A. Angluin and M. Kharitonov, When won't membership queries help?, *J. Comput. System Sci.*, vol. 50, 1995, pp. 336-355.

[2] P. W. Beame, S. A. Cook and H. J. Hoover, Log depth circuits for division and related problems, *SIAM J. Comput.*, vol. 15, 1986, pp. 994-1003.

[3] M. Bellare and S. Goldwasser, New paradigms for digital signatures and message authentication based on non-interactive zero knowledge proofs *Proc. Advances in Cryptology - CRYPTO '89*, LNCS, Springer, 1990, pp. 194-211.

[4] M. Bellare and S. Micali, Non-interactive oblivious transfer and applications, *Proc. Advances in Cryptology - CRYPTO '89*, LNCS, Springer, 1990, pp. 547-557.

[5] A. Blum, M. Furst, M. Kearns and R.J. Lipton, Cryptographic primitives based on hard learning problems, in: D.R. Stinson, ed., *Advances in Cryptology - CRYPTO '93*, LNCS, vol. 773, Springer, 1994, pp. 278-291.

[6] E. Biham, D. Boneh and O. Reingold, Generalized Diffie-Hellman modulo a composite is not weaker than factoring, manuscript.

[7] L. Blum, M. Blum and M. Shub, A simple secure unpredictable pseudo-random number generator, *SIAM J. Comput.*, vol. 15, 1986, pp. 364-383.

[8] M. Blum, W. Evans, P. Gemmell, S. Kannan, M. Naor, Checking the correctness of memories, *Algorithmica*, 1994, pp. 225–244.

[9] M. Blum and S. Goldwasser, An efficient probabilistic public-key encryption scheme which hides all partial information, *Advances in Cryptology - CRYPTO '84*, LNCS, vol. 196, Springer, 1984, pp. 289–302.

[10] M. Blum and S. Micali, How to generate cryptographically strong sequence of pseudo-random bits, *SIAM J. Comput.*, vol. 13, 1984, pp. 850-864.

[11] D. Boneh and R. Lipton, Algorithms for Black-Box fields and their application to cryptography, *Advances in Cryptology - CRYPTO '96*, LNCS, vol. 1109, Springer, 1996, pp. 283–297.

[12] D. Boneh and R. Venkatesan, Hardness of computing most significant bits in secret keys in Diffie-Hellman and related schemes, *Advances in Cryptology - CRYPTO '96*, LNCS, vol. 1109, Springer, 1996, pp. 129–142.

[13] S. Brands, An efficient off-line electronic cash system based on the representation problem, CWI Technical Report, CS-R9323, 1993.

[14] G. Brassard, **Modern cryptology**, LNCS, vol. 325, Springer, 1988.

[15] E. F. Brickell, D. M. Gordon, K. S. McCurley and D. B. Wilson, Fast exponentiation with precomputation, *Proc. Advances in Cryptology - EUROCRYPT '92*, LNCS, Springer, 1992, pp. 200–207.

[16] R. Canetti, Towards de-mystification of random oracles: hash functions that hide all partial information, *Proc. Advances in Cryptology - CRYPTO '97*, LNCS, Springer, 1997.

[17] R. Canetti, J. Friedlander and I. Shparlinsky, On certain exponential sums and the distribution of Diffie-Hellman triples, preprint, 1997.

[18] D. Chaum and H. van Antwerpen, Undeniable signatures, *Proc. Advances in Cryptology - CRYPTO '89*, LNCS, Springer, 1990, pp. 212-216.

[19] B. Chor, A. Fiat and M. Naor, Tracing traitors, *Advances in Cryptology - CRYPTO' 94*, LNCS, 1994, pp. 257–270.

[20] W. Diffie and M. Hellman, New directions in cryptography, *IEEE Trans. Inform. Theory*, vol. 22(6), 1976, pp. 644-654.

[21] T. ElGamal, A public-key cryptosystem and a signature scheme based on discrete logarithms, *Advances in Cryptology - CRYPTO '84*, LNCS, vol. 196, 1985, pp. 10-18.

[22] M. Franklin and S. Haber, Joint encryption and message-efficient secure computation, *J. of Cryptology*, vol 9(4), 1996, pp. 217-232.

[23] Y. Gertner and T. Malkin, A PSRG based on the decision Diffie-Hellman assumption, preprint, 1997.

[24] O. Goldreich, Two remarks concerning the Goldwasser-Micali-Rivest signature scheme, *Advances in Cryptology - CRYPTO' 86*, LNCS, Springer, vol. 263, 1987, pp. 104-110.

[25] O. Goldreich, Towards a theory of software protection, *Proc. 19th Ann. ACM Symp. on Theory of Computing*, 1987, pp. 182-194.

[26] O. Goldreich, **Foundations of Cryptography (Fragments of a Book)**, 1995. Electronic publication: `http://www.eccc.uni-trier.de/eccc/` (Electronic Colloquium on Computational Complexity).

[27] O. Goldreich, S. Goldwasser and S. Micali, How to construct random functions, *J. of the ACM.*, vol. 33, 1986, pp. 792-807.

[28] O. Goldreich, S. Goldwasser and S. Micali, On the cryptographic applications of random functions, *Advances in Cryptology - CRYPTO '84*, LNCS, vol. 196, Springer, 1985, pp. 276-288.

[29] O. Goldreich and L. Levin, A hard-core predicate for all one-way functions, in: *Proc. 21st Ann. ACM Symp. on Theory of Computing*, 1989, pp. 25-32.

[30] S. Goldwasser and S. Micali, Probabilistic encryption, *J. Comput. System Sci.*, vol. 28(2), 1984, pp. 270-299.

[31] J. Hastad, R. Impagliazzo, L. A. Levin and M. Luby, Construction of a pseudo-random generator from any one-way function, To appear in *SIAM J. Comput.* Preliminary versions by Impagliazzo et. al. in *21st STOC*, 1989 and Hastad in *22nd STOC*, 1990.

[32] R. Impagliazzo and M. Naor, Efficient Cryptographic schemes provably secure as subset sum, *Journal of Cryptology* vol 9, 1996, pp. 199-216.

[33] R. Impagliazzo, and D. Zuckerman, Recycling random bits, *Proc. 30th IEEE Symposium on Foundations of Computer Science*, 1989, pp. 248-253.

[34] M. Kharitonov, Cryptographic hardness of distribution-specific learning, *Proc. 25th ACM Symp. on Theory of Computing*, 1993, pp. 372-381.

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

[36] N. Linial, Y. Mansour and N. Nisan, Constant depth circuits, Fourier transform, and learnability, *J. of the ACM.*, vol 40(3), 1993, pp. 607-620.

[37] M. Luby, **Pseudo-randomness and applications**, Princeton University Press, 1996.

[38] M. Luby and C. Rackoff, How to construct pseudorandom permutations and pseudorandom functions, *SIAM J. Comput.*, vol. 17, 1988, pp. 373-386.

[39] U. Maurer, Towards the equivalence of breaking the Diffie-Hellman protocol and computing discrete logarithms, *Advances in Cryptology - CRYPTO 94*, LNCS, vol. 740, Springer, 1994, pp. 271–281.

[40] K. McCurley, A key distribution system equivalent to factoring, *J. of Cryptology*, vol 1, 1988, pp. 95-105.

[41] M. Naor and O. Reingold, Synthesizers and their application to the parallel construction of pseudo-random functions, *Proc. 36th IEEE Symp. on Foundations of Computer Science*, 1995, pp. 170-181.

[42] M. Naor and O. Reingold, On the construction of pseudo-random permutations: Luby-Rackoff revisited, *Proc. 29th Ann. ACM Symp. on Theory of Computing*, 1997. Full version in: *Theory of Cryptography Library: Record 96-11* at: `http://theory.lcs.mit.edu/~tcryptol/homepage.html`

[43] R. Ostrovsky, An efficient software protection scheme, *Proc. 22nd Ann. ACM Symp. on Theory of Computing*, 1990, pp. 514-523.

[44] A. Razborov, and S. Rudich, Natural proofs, *Proc. 26th Ann. ACM Symp. on Theory of Computing*, 1994, pp. 204-213.

[45] J. Reif, On threshold circuits and polynomial computation, *Proc. of the 2nd Conference on Structure in Complexity Theory* 1987, pp. 118–123.

[46] J. Reif and S. Tate, On threshold circuits and polynomial computation, *SIAM J. Comput.*, vol. 5, 1992, pp. 896–908.

[47] K.-Y. Siu, J. Bruck, T. Kailath and T. Hofmeister, Depth efficient neural network for division and related problems, *IEEE Trans. Inform. Theory*, vol. 39, 1993, pp. 946-956.

[48] K.-Y. Siu and V. P. Roychowdhury, On optimal depth threshold circuits for multiplication and related problems, *SIAM J. Disc. Math.*, vol. 7(2), 1994, pp. 284-292.

[49] Z. Shmuely, Composite Diffie-Hellman public-key generating systems are hard to break, Technical Report No. 356, Computer Science Department, Technion, Israel, 1985.

[50] V. Shoup, Lower bounds for discrete logarithms and related problems, to appear in: *Proc. Advances in Cryptology - EUROCRYPT '97*, 1997.

[51] M. Steiner, G. Tsudik and M. Waidner, Diffie-Hellman key distribution extended to group communication, *Proceedings 3rd ACM Conference on Computer and Communications Security*, 1996, pp. 31-37.

[52] L. G. Valiant, A theory of the learnable, *Comm. ACM*, vol. 27, 1984, pp. 1134-1142.

[53] A. C. Yao, Theory and applications of trapdoor functions, *Proc. 23rd IEEE Symp. on Foundations of Computer Science*, 1982, pp. 80-91.