Number-Theoretic Constructions of Efficient Pseudo-Random Functions*

Moni Naor † Omer Reingold ‡

Abstract

We describe efficient constructions for various cryptographic primitives in private-key as well as public-key cryptography. Our major results are two new constructions of pseudo-random functions. We prove the pseudo-randomness of one construction under the assumption that *factoring* (Blum integers) is hard while the other construction is pseudo-random if the *decisional version of the Diffie-Hellman* assumption holds. Computing the value of our functions at any given point involves two subset 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). This fact has several interesting applications. The simple algebraic structure of the functions implies additional features such as 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.

*A preliminary version of this paper appeared in the Proceedings of FOCS'97 [58].

†Dept. of Computer Science and Applied Math, 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. E-mail: naor@wisdom.weizmann.ac.il.

‡AT&T Labs - Research. 180 Park Avenue, Bldg. 103, Florham Park, NJ, 07932, USA. Most of this research was performed while still at the Weizmann Institute of Science, Rehovot, Israel. Research supported by a Clore Scholars award. E-mail: omer@research.att.com.

1 Introduction

This paper studies the efficient construction of several fundamental cryptographic primitives. Our major result are two related constructions of pseudo-random functions based on number-theoretic assumptions. The first construction gives pseudo-random functions iff the *decisional* version of the Diffie-Hellman assumption (**DDH-Assumption**) holds. The second construction is at least as secure as the assumption that factoring the so called Blum-integers is hard.[1] Having efficient pseudo-random functions based on factoring is very desirable since this is one of the most established concrete intractability assumption used in cryptography. The construction based on the DDH-Assumption is also attractive since these pseudo-random functions are even more efficient (in that they have a larger output size) and since the construction is *linear preserving* (see Remark 4.1). We consider the study of the DDH-Assumption (which was recently used in quite a few interesting applications) to be one of the contributions of this paper.

Properties of Our Pseudo-Random Functions

Pseudo-random functions were introduced by Goldreich, Goldwasser and Micali [35] and have innumerable applications (e.g., [3, 9, 22, 32, 40, 36, 50, 59]). A distribution of functions is pseudo random 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. The properties of our new pseudo-random functions are:

Efficiency: 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 pseudo-random functions.

Depth: Given appropriate preprocessing of the key, the value of the functions at any given point can be computed in TC^0, compared with TC^1 previously (in [60]). 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 [64]) and to computational learning-theory.

Simplicity: The simple algebraic structure of the functions implies additional desirable features. To demonstrate this, we showed in [58] 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.

More on the motivation of such a construction and on pseudo-random functions in general can be found in Section 2.2.

[1] In fact we prove the security of the second construction based on a *generalized* version of the computational DH-Assumption (**GDH-Assumption**). However, breaking the GDH-Assumption modulo a composite would imply an efficient algorithm for factorization (see [6, 69]).

The DDH-Assumption

As mentioned above, we base our constructions on two number-theoretic assumptions: Factoring and the DDH-Assumption. While the assumption that factoring is hard is a well-established cryptographic assumption that needs little introduction the DDH-Assumption is relatively new. In the following few paragraphs we briefly describe the DDH-Assumption, its different applications and the current knowledge on its security. In addition we briefly describe the contribution of this paper to the study of this assumption. A more detailed description appears in Section 3.1.

The DH-Assumption was introduced in the context of the Diffie and Hellman [28] 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 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 Z_P^*, and a, b and c are uniformly chosen from Z_Q (naturally all exponentiations are in Z_P^*).*

Note that this assumption does not hold when g is a generator of 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 3.3 we show that for any given P and Q the DDH-assumption can be reduced to its worst-case version:

> *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 $c = a \cdot b$ for* every *a, b and c in Z_Q and* every *element, g, of order Q in Z_P^*.*

The randomized reduction we describe is based on the random-self-reducibility of the DDH-Problem that was previously used by Stadler [73]. This reduction 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 [13, 51, 52, 69] and by additional results [13, 18, 70]. For instance, Shoup [70] showed that the DDH-Problem is hard for any "generic" algorithm. However, a main conclusion of this paper is that the DDH-Assumption deserves more attention since it implies the security of many attractive cryptographic constructions.

The most obvious application of the DDH-Assumption is to the Diffie-Hellman key-exchange protocol and to the related public-key cryptosystem [29]. 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 we show how to adjust this cryptosystem in order to obtain a probabilistic encryption-scheme whose semantic security (see [39]) 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 [10]. Other applications that previously appeared are [4, 14, 17, 30, 74, 73] and recently [24] (we describe these applications in Section 3.1).

To previous applications one can add a pseudo-random generator [11, 76] that practically doubles the input length and a pseudo-random synthesizer (see definition in [60]) 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 pseudo-random functions. The pseudo-random function is defined by $n+1$ values, $\langle a_0, a_1 \ldots a_n\rangle$, chosen uniformly in Z_Q. The value of the function on an n-bit input $x = x_1 x_2 \cdots x_n$ is essentially

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

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 subset products. This can be done in TC^0 (see Section 4.3). The simple structure of the functions gives several attractive properties as was shown in [58] (see further details in Section 2.2).

The GDH-Assumption and Factoring

In Section 5 we suggest a related construction of pseudo-random 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., [69, 74]). The GDH-Assumption is implied by the DDH-Assumption (as shown in [74] and in this paper) but the assumptions are not known to be equivalent. In addition, the GDH-Assumption modulo a Blum-integer is not stronger than the assumption that factoring Blum-integers is hard (see [6, 69]). This implies an attractive construction of pseudo-random functions that are at least as secure as 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 $\vec{a} = \langle a_{1,0}, a_{1,1}, a_{2,0}, a_{2,1}, \ldots a_{n,0}, a_{n,1}\rangle$ 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, $f_{N,g,\vec{a},r}$, is pseudo-random. Here the value of $f_{N,g,\vec{a},r}$ on any n-bit input, $x = x_1 x_2 \cdots x_n$, is defined by:

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

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

This construction was recently improved by Naor, Reingold and Rosen [61]. The work in [61] provides a method of expanding the one bit output of $f_{N,g,\vec{a},r}$ to polynomially many

[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.

bits while paying only a small overhead in the complexity of the evaluation (i.e. one modular multiplication for each additional output bit). In particular, this implies a length-preserving pseudo-random function that is at least as secure as Factoring whose evaluation requires only a constant number of modular multiplications per output bit.

Previous Work

In addition to introducing pseudo-random functions, Goldreich, Goldwasser and Micali [35] provided a construction of such functions (GGM-Construction) based on pseudo-random generators. Naor and Reingold [60] recently showed a *parallel* construction based on a new primitive called a pseudo-random synthesizer. Under concrete intractability assumptions like "factoring is hard" this construction gives pseudo-random functions in TC^1. Our work is motivated by [60] both in the task of reducing the depth of the pseudo-random functions and in the construction itself (see Section 5.2). Parallel constructions of other cryptographic primitives were provided by Impagliazzo and Naor [42] based on the hardness of subset sum and factoring, and by Blum et. al. [7] based on hard-to-learn problems.

The construction of this paper is not only more parallelizable than the concrete constructions based on [35, 60], but it is also much more efficient. Though this construction seems very different than the constructions of [35, 60], we were able to relate the proof of security of this construction to both [35] and [60] (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 pseudo-random *generators* with good expansion. One such construction was proposed by Rackoff (unpublished). A different construction is suggested by Gertner and Malkin [31]. This construction is similar to the pseudo-random generator one gets by scaling down our pseudo-random functions.

Organization

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 pseudo-random 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 pseudo-random functions based on the DDH-Assumption, prove its security and consider its complexity. In Section 5 we define the GDH-Assumption and show a related construction of pseudo-random functions based on this assumption. In Section 6 we consider some of the possible features of our pseudo-random functions and suggest directions for further research.

2 Preliminaries

2.1 Notation and Conventions

- For any integer N the multiplicative group modulo N is denoted by Z_N^* and the additive group modulo N is denoted by Z_N.

- For any integer k, denote by $[k]$ the set of integers - $\{1, 2, \ldots, k\}$. For any two integers $k < m$, denote by $[k..m]$ the set of integers - $\{k, k+1, \ldots, m\}$.
- For any two bit-strings of the same length, x and y, the inner product mod 2 of x and y is denoted by $x \odot y$.

2.2 Pseudo-Random Functions

As mentioned in the introduction, our main result is a construction of a pseudo-random function that is *efficient, has shallow depth and is simple*. We devote this section to motivate such a construction and to describe previous constructions and applications of pseudo-random functions. Good references on pseudo-random functions and pseudo-randomness in general are Goldreich [33, 34] and Luby [49].

The concept of a pseudo-random function-ensemble was introduced by Goldreich, Goldwasser and Micali [35]. Loosely, this is an efficient function-ensemble that cannot be efficiently distinguished from the uniform function-ensemble by an adversary that has access to the functions as a black-box (see Definition 2.1). In addition, Goldreich, Goldwasser and Micali provided a construction of pseudo-random functions (GGM-Construction). This construction uses pseudo-random generators [11, 76] as building blocks, which in turn can be obtained from any one-way function (as shown by Hastad, Impagliazzo, Levin and Luby [41]).

A pseudo-random function is a powerful cryptographic primitive that can replace functions truly chosen uniformly at random in many applications. Probably, the most notable applications of pseudo-random functions are in private-key cryptography. They provide parties who share a common key straightforward protocols for sending secret messages to each other, for identifying themselves and for authenticating messages [15, 36, 49]. As shown by Luby and Rackoff [50], it is possible to efficiently construct pseudo-random *permutations* (which, in particular, can be used as block-ciphers) from pseudo-random functions (see also [59] for an "optimal" construction). However, pseudo-random functions have many other applications including applications in public-key cryptography. For example, Bellare and Goldwasser [3] showed how to use pseudo-random functions and a non-interactive zero-knowledge proof of their values to construct digital-signatures. Another interesting example was given by Goldreich [32] who showed how to eliminate the state in the Goldwasser-Micali-Rivest signature scheme (the technique of [32] is very general). For some of the additional applications of pseudo-random functions see [9, 22, 32, 40].

For quite a while, the GGM-Construction was the only known construction of pseudo-random functions (that was proven to be as secure as some general or concrete intractability assumption). Motivated by the inherent sequentiality of the GGM-Construction, Naor and Reingold [60] recently showed a *parallel* construction based on a new primitive called a pseudo-random synthesizer. In addition, they showed how to construct pseudo-random synthesizers in parallel from general cryptographic-primitives (such as trapdoor permutations) and based on several concrete intractability assumption. Their concrete constructions give NC^2 (or actually TC^1) pseudo-random functions. In fact, our work is motivated by [60], as described in Section 5.2.

The construction of this paper gives pseudo-random functions computable in TC^0 (given appropriate preprocessing). We briefly summarize part of the discussion that appears in

[60] on the applications of parallel pseudo-random functions:

- It is likely that pseudo-random functions will be implemented in hardware (as is the case for DES). In such implementations, having shallow-depth pseudo-random functions implies reduced latency in computing those functions which, for some applications (such as the encryption of messages on a network), is essential.

- As was observed by Valiant [75], if a concept class contains pseudo-random 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 pseudo-random functions is very strong. Weaker unlearnability results for NC^1 and TC^0, based on other cryptographic assumptions, were obtained in [1, 45, 44]. It is also interesting to compare with the result of Linial, Mansour and Nisan [48] who showed that AC^0 *can* be learned in time slightly super-polynomial under the uniform distribution on the examples.

- Another application of pseudo-random functions in complexity was suggested by Razborov and Rudich [64]. They showed that if a circuit class contains pseudo-random functions (that are secure against a subexponential-time adversary), then there are no (what they called) *Natural Proofs* (which include all previously 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 sufficiently 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 [45], which is based on the pseudo-random generator of Blum, Blum and Shub [8].

Except of being more parallelizable, our construction has two additional advantages over previous ones:

1. It is *efficient*: computing the value of the function at any given point is comparable with two exponentiations. This is the first construction that seems efficient enough to be implemented and indeed these functions were implemented by Langberg in [47]. Given the many applications of pseudo-random functions it is clear that having efficient pseudo-random functions is an important goal.

   A somewhat surprising fact is that although our construction is much more efficient than previous ones it is still closely related to the GGM-Construction and to the construction of [60]. The connection with previous constructions is described in Sections 4.2 and 5.2.

2. It has a *simple* algebraic structure. To see our main motivation here, consider the Bellare-Goldwasser signature scheme. The public key in this scheme contains a commitment for a key s of a pseudo-random function. The signature for a message m is composed of a value y and a non-interactive zero-knowledge proof that $y = f_s(m)$.

In order for this scheme to be attractive, we must have a simple non-interactive zero-knowledge proof for claims of the form $y = f_s(m)$. In this and other scenarios we might wish to have additional properties for the functions such as a simple function-sharing scheme in the sense of [25]. It seems that for such properties to be possible we need a simple construction of pseudo-random functions.

In [58] we considered some desirable features of pseudo-random functions. We also presented preliminary results in obtaining these features for our construction of pseudo-random functions: (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 pseudo-random 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 "oblivious evaluation" of the value of the function: Assume that a party, $\mathcal{A}$, knows a key s of a pseudo-random 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). We consider the task of improving these protocols and designing additional ones to be an interesting line for further research.

2.2.1 Definition of Pseudo-Random Functions

For the sake of concreteness we include the definition of pseudo-random functions almost as it appears in [33, 34]:

Definition 2.1 (efficiently computable pseudo-random function ensemble) *Let $\{A_n, B_n\}_{n\in N}$ be a sequence of domains and let $F = \{F_n\}_{n\in N}$ be a function ensemble such that the random variable F_n assumes values in the set of $A_n \rightarrow B_n$ functions. Then F is called* an efficiently computable pseudo-random function ensemble *if the following conditions hold:*

1. (pseudo-randomness) *for every probabilistic polynomial-time oracle machine $\mathcal{M}$, every constant $c > 0$, and all but a finite number of n's*

$$\left|\Pr[\mathcal{M}^{F_n}(1^n) = 1] - \Pr[\mathcal{M}^{R_n}(1^n) = 1]\right| < \frac{1}{n^c},$$

   where $R = \{R_n\}_{n\in N}$ is the corresponding uniform function ensemble (i.e., $\forall n$, R_n is uniformly distributed over the set of $A_n \rightarrow B_n$ functions).

2. (efficient computation) *There exist probabilistic polynomial time algorithms, $\mathcal{I}$ and $\mathcal{V}$, and a mapping from strings to functions, ϕ, such that $\phi(\mathcal{I}(1^n))$ and F_n are identically distributed and $\mathcal{V}(i, x) = (\phi(i))(x)$.*

Remark 2.1 *In this definition, as well as the other definitions of this paper, "efficient adversary" is interpreted as "probabilistic polynomial-time algorithm" and "negligible" is interpreted as "smaller than $1/poly$". In fact, all the proofs in this paper easily imply more quantitative results. For a discussion on security preserving reductions see [49].*

3 The Decisional Diffie-Hellman Assumption

As mentioned above, we base our first construction of pseudo-random functions (described in Section 4) on the DDH-Assumption (the decisional version of the DH-Assumption). This assumption is relatively new, or more accurately, was *explicitly* considered only recently. We therefore devote this section to a discussion of the DDH-Assumption: we describe and define the assumption, consider some of its different applications and the current knowledge on its security. Furthermore, we show in Section 3.3 a randomized reduction of the worst-case DDH-Assumption to its average case. In Section 5 we describe a related construction of pseudo-random functions based on a more conservative assumption: the assumption that factoring Blum-integers is hard (in fact, this construction is based on the GDH-Assumption that in turn can be reduced to Factoring).

3.1 Background

The DH-Assumption was introduced in the context of the Diffie and Hellman [28] 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 pseudo-random 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 Z_P^* (both publicly known), 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 $\mathcal{B}$. 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 publicly known manner. 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 [37] hard-core function[3] to $g^{a \cdot b}$ (an important improvement on the security of such an application was made by Shoup [70]). If the CDH-Assumption holds, then this method indeed gives a pseudo-random key. However, the proof in [37] 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 $z = g^{a \cdot b}$.

[3]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).

However, when g is a generator of 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 Z_P^* of order Q, where Q is a large prime divisor of $P-1$. In fact, for most applications, using g of order Q is an advantage since Q may be much smaller than P (say, 160 bits long) which results in a substantial improvement in efficiency. The reason that Q may be so small is that all known *subexponential* algorithms for computing the discrete log are *subexponential* in the length of P (as long as $P-1$ is not too smooth) even when applied to the subgroup of size Q generated by g (see, [53, 62] for surveys on algorithms for the discrete log; the best known algorithm for general groups has time square root of the size of the largest prime divisor of the group).

How Much Confidence Can we Have in the DDH-Assumption?

It is clear that the computational DH-Problem is at most as hard as computing the discrete log (given $\langle g, g^a \rangle$ find a). Recent works by Maurer and Wolf [51] and Boneh and Lipton [12] show that in several settings these two problems are in fact equivalent. For example, Maurer and Wolf showed that given some information which only depends on P and an efficient algorithm for computing the DH-Problem in Z_P^*, one can efficiently compute the discrete log in Z_P^* (so in some non-uniform sense these problems are equivalent). Shoup [70] showed that there are no efficient "generic" algorithms for computing the discrete log or the DH-Problem, where loosely speaking, a generic algorithm is one that does not exploit any special properties of the encoding of group elements. A bit more formally, a generic algorithm is one that works for a "black-box" group (where each element has a random encoding and given the encodings of a and b the algorithm can query for the encodings of $a+b$ and $-a$).

Perhaps the best evidence for the validity of the CDH-Assumption is the fact that it endured extensive research over the last two decades. This research does not seem to undermine the (stronger) decisional version of the DH-Assumption as well. In addition, the DDH-Assumption did appear both explicitly and implicitly in several previous works. However, it seems that, given the many applications of the DDH-Assumption, a more extensive study of its security is in place.

To some extent, the DDH-Assumption is supported by the work of Shoup [70] and the work of Boneh and Venkatesan [13]. Shoup showed that the DDH-Problem is hard for any generic algorithm (where a generic algorithm is as defined above). Boneh and Venkatesan showed that computing the k ($\approx \sqrt{\log P}$) most significant bits of $g^{a \cdot b}$ (given $\langle g, g^a, g^b \rangle$) is as hard as computing $g^{a \cdot b}$ itself. A recent result with applications to the DDH-Assumption was shown by Canetti, Friedlander and Shparlinski [18].

In Section 3.3 we prove an attractive feature of the DDH-Assumption: There is a quite simple randomized reduction between its worst-case and its average-case for fixed P and Q. More specifically:

> *For any primes P and Q (such that Q divides $P-1$), the following statements are equivalent:*
>
> - *Given $\langle P, Q, g, g^a, g^b \rangle$, it is easy to distinguish with non-negligible advantage*

between $g^{a \cdot b}$ and g^c, where g is a uniformly chosen element of order Q in Z_P^*, and a, b and c are uniformly chosen from Z_Q.

- *Given $\langle P, Q, g, g^a, g^b, g^c \rangle$, it is easy to decide with overwhelming success probability whether or not $c = a \cdot b$, where a, b and c are* any *three elements in Z_Q and g is* any *element of order Q in Z_P^*.*

This reduction is based on the random-self-reducibility of the DDH-Problem that was previously used by Stadler [73]. The reduction may strengthen our confidence in the DDH-Assumption and in the security of its applications.

For most applications of the DDH-Assumption (including ours) there is no reason to insist on working in a subgroup of Z_P^* (where P is a prime). Therefore, a natural question is how valid is this assumption for other groups. Specific groups that were considered in the context of the CDH-Assumption are: (1) Z_N^* where N is a composite. McCurley and Shmuely [52, 69] showed that for many of those groups breaking the CDH-Assumption is at least as hard as factoring N. (2) Elliptic-curve groups, for which (in some cases) no subexponential algorithms for the discrete log are currently known. We stress that the randomized reduction mentioned above relies on the primality of the order of g.

The Decisional DH-Assumption is Very Attractive

It turns out that the DDH-Assumption was assumed in several previous works (both explicitly and implicitly). In the following, we briefly refer to some of those works and describe some additional applications.

The most obvious application of the DDH-Assumption is to the Diffie-Hellman key-exchange protocol and to the related public-key cryptosystem, namely the ElGamal cryptosystem - given the public key g^a the encryption of a message m is $\langle g^b, g^{a \cdot b} \cdot m \rangle$. Assume that the message space is restricted to the subgroup generated by g. In this case, it is easy to see that the semantic security (see [39]) of the cryptosystem is equivalent to the DDH-Assumption. In the general case (without the restriction on the message space), we can use the following related cryptosystem: given the public key $\langle g^a, h \rangle$ the encryption of a message m is $\langle g^b, h(g^{a \cdot b}) \oplus m \rangle$, where h is a pair-wise independent hash function from n-bit strings to strings of approximately the length of Q (see Lemma 4.2 for more details on the role of h). Therefore, given the DDH-Assumption, we get a probabilistic encryption of many bits for the price of a single (or two) exponentiation. This is comparable with the Blum-Goldwasser cryptosystem [10].

Other applications that previously appeared are:

- Bellare and Micali [4] showed an efficient non-interactive oblivious transfer of many bits that relies on the DDH-Assumption.

- Brands [14] pointed out that several suggestions for undeniable signatures (as the one in [19] where this concept was introduced) implicitly rely on the DDH-Assumption. If this assumption does not hold then such schemes are in fact *ordinary* digital signatures.

- Canetti [17] gave a simple construction based on the DDH-Assumption for a new cryptographic primitive called "Oracle Hashing" (later renamed "perfectly one-way probabilistic hash functions"). Loosely, these are hash functions that "hide all partial information" on their input.

- Franklin and Haber [30] showed a construction of a joint encryption scheme based on the DDH-Assumption modulo a *composite*. Using this scheme they showed how to obtain an efficient protocol for secure circuit computation.

- Stadler [73] presents verifiable secret sharing based on the DDH-Assumption.

- Steiner, Tsudik and Waidner [74] showed how to extend the Diffie-Hellman protocol to a key-exchange protocol for a group of parties. They reduced the security of the extended protocol to the DDH-Assumption (by showing that the DDH-Assumption implies the *decisional* GDH-Assumption).

A very attractive application of the DDH-Assumption was recently proposed by Cramer and Shoup [24]. They have presented a new public-key cryptosystem that is secure against adaptive chosen ciphertext attacks. Both encryption and decryption in this cryptosystem only require a few exponentiations (in addition to universal one-way hashing).

To all these applications we can add:

- A pseudo-random generator that practically doubles the input length. Essentially, the generator is defined by $G_{P,Q,g,g^a}(b) = \langle g^b, g^{a \cdot b} \rangle$.[4] As mentioned in the introduction, several unpublished constructions of pseudo-random generators based on the DDH-Assumption were previously suggested.

- A pseudo-random synthesizer (see definition in [60]) whose output length is similar to its arguments length, essentially defined by $S_{P,Q,g}(a, b) = g^{a \cdot b}$.

Both these constructions are overshadowed by the construction of pseudo-random functions introduced in Section 4.1.

3.2 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 Z_P^*).

Let n be the security parameter. For some function $\ell : N \to N$ 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* probabilistic polynomial-time algorithm IG (where IG stands for instance generator). On input 1^n the output of IG is distributed over triplets

[4] In fact, the output of G_{P,Q,g,g^a} is a pseudo-random pair of values in the subgroup generated by g. In order to obtain a pseudo-random value in $\{0,1\}^\ell$, for ℓ of approximately twice the length of Q, one needs to hash the output of the generator (see Lemma 4.2). A similar observation holds for the constructions of pseudo-random synthesizers and pseudo-random functions.

$\langle P, Q, g \rangle$, where P is an n-bit prime, Q a (large) prime divisor of $P-1$ and g an element of order Q in Z_P^*. Any instantiation of IG will imply a different DDH-Assumption and a different construction of pseudo-random functions. Our proof of pseudo-randomness of the functions based on the DDH-Assumption is independent of the particular instantiation of IG.

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 3.3 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 strengthened if g (generated by IG) is taken to be a uniformly chosen element of order Q in Z_P^*.

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

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

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 Z_Q.*

3.3 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. The main part of the reduction (Lemma 3.2) was previously used by Stadler [73].

Definition 3.1 *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 Z_P^* the function $DDH_{P,Q,g}$ is defined by*

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

for any three elements a, b, c in 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 exist a polynomial $p = p(n)$ and a probabilistic algorithm $\mathcal{A}'$ with running time $(t(n) \cdot p(n))/(\epsilon(n))^2$ such that, for* any *choice of $\langle P, Q, g \rangle$ as in Definition 3.1, if:*

$$\left| \Pr[\mathcal{A}(P, Q, g, g^a, g^b, g^{a \cdot b}) = 1] - \Pr[\mathcal{A}(P, Q, g, g^a, g^b, g^c) = 1] \right| > \epsilon(n),$$

where the probabilities are taken over the random bits of $\mathcal{A}$ and the choice of a, b and c uniformly *at random in Z_Q, then for* any *a, b and c in 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 [11] 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 (or on a polynomial number of random instances). That is, 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 [11] 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. [51]): 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 can only hold when g is a generator of Z_P^* if the DDH-Problem is always easy (in which case the theorem holds trivially).

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 Z_P^* and a, b, c are three elements in Z_Q the output of $\mathcal{R}$ is:

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

where if $c = a \cdot b$, then a' and b' are uniform in Z_Q and $c' = a' \cdot b'$ and if $c \neq a \cdot b$, then a', b' and c' are all *uniform in Z_Q.*

Proof: $\mathcal{R}$ chooses s_1, s_2 and r uniformly in Z_Q, computes

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

and outputs

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

Let $c = a \cdot b + e$ for e in Z_Q then:

$$a' = r \cdot a + s_1,\ b' = b + s_2,\ c' = a'b' + e \cdot r.$$

If $e = 0$ we have that a' and b' are uniformly distributed in Z_Q and $c' = a' \cdot b'$. If $e \neq 0$ we have that a', b' and c' are all uniformly distributed in Z_Q (this is the place we use the fact that Q is a prime which implies that $e \cdot r$ is uniformly distributed in Z_Q). Therefore, the output of $\mathcal{R}$ has the desired distribution. $\square$

Proof: (of Theorem 3.1) Let $\mathcal{A}$ be any probabilistic algorithm with running time $t = t(n)$, let $\epsilon = \epsilon(n)$ be any positive function such that $1/\epsilon$ is efficiently constructible and let $\langle P, Q, g\rangle$ be as in Definition 3.1. Assume that:

$$\left|\Pr[\mathcal{A}(P, Q, g, g^a, g^b, g^{a \cdot b}) = 1] - \Pr[\mathcal{A}(P, Q, g, g^a, g^b, g^c) = 1]\right| > \epsilon(n),$$

where the probabilities are taken over the random bits of $\mathcal{A}$ and the choice of a, b and c *uniformly* at random in Z_Q.

Let $\mathcal{R}$ be the probabilistic polynomial-time algorithm that is guaranteed to exist by Lemma 3.2. By the definition of $\mathcal{R}$ and our assumption, we have that for *any* a, b and $c \neq a \cdot b$ in Z_Q:

$$\left|\Pr[\mathcal{A}(\mathcal{R}(P, Q, g, g^a, g^b, g^{a \cdot b})) = 1] - \Pr[\mathcal{A}(\mathcal{R}(P, Q, g, g^a, g^b, g^c)) = 1]\right| > \epsilon(n).$$

Now the probabilities are *only* taken over the random bits of $\mathcal{A}$ and $\mathcal{R}$. Therefore, by standard methods of amplification (see e.g., [34] and references therein) we can define a probabilistic algorithm $\mathcal{A}'$ such that for *any* a, b and $c \neq a \cdot b$ in Z_Q:

$$\Pr[\mathcal{A}'(P, Q, g, g^a, g^b, g^{a \cdot b}) = 1] - \Pr[\mathcal{A}'(P, Q, g, g^a, g^b, g^c) = 1] > 1 - 2^{-n}.$$

On any input $\langle P, Q, g, g^a, g^b, g^c\rangle$, the output of $\mathcal{A}'$ is essentially a threshold function of $O(n/(\epsilon(n))^2)$ independent values - $\mathcal{A}(\mathcal{R}(P, Q, g, g^a, g^b, g^c))$. It is clear that $\mathcal{A}'$ satisfies the conditions required in Theorem 3.1. $\square$

4 Construction of Pseudo-Random Functions

In this section we describe a construction of pseudo-random functions based on the DDH-Assumption, prove its security and consider its complexity. A related construction (based on a weaker assumption) is described in Section 5.

4.1 Construction and Main Result

Construction 4.1 *We define the function ensemble $F = \{F_n\}_{n \in N}$. 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 Z_P^* and $\vec{a} = \langle a_0, a_1, \ldots a_n\rangle$ a sequence of $n + 1$ elements of 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 $IG(1^n)$.

It is clear that F is efficiently computable (since IG is efficient). The pseudo-randomness property of F is the following:

Theorem 4.1 *Let $F = \{F_n\}_{n \in N}$ 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 but a finite number of n's*

$$\left| \Pr[\mathcal{M}^{f_{P,Q,g,\vec{a}}}(P,Q,g) = 1] - \Pr[\mathcal{M}^{R_{P,Q,g}}(P,Q,g) = 1] \right| < \frac{1}{n^\alpha},$$

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 Z_P^* generated by g).

Moreover, if there exists a probabilistic oracle machine with running time $t = t(n)$ that distinguishes $f_{P,Q,g,\vec{a}}$ from $R_{P,Q,g}$ (as above) 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 $\epsilon(n)/n$.

Remark 4.1 *The "moreover" part of Theorem 4.1 implies that the security of the functions does not significantly decrease when the* number of queries *the distinguisher makes increases. More formally, we have that this reduction is in fact* linear-preserving *(see [49]). This is a strong and quite unique property (and in particular it is very different from the proofs of security for the functions in [35, 60]).*

Given Theorem 4.1, we have that F is "almost" an efficiently computable pseudo-random 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 pseudo-random functions (Definition 2.1). However, for many applications of pseudo-random functions this deviation does not present a problem (e.g., the applications of pseudo-random functions to private-key authentication and identification and their applications to digital signatures [3]). In addition, it is rather easy to construct from F pseudo-random functions under Definition 2.1. In order to show this, we need the following lemma which is a simple corollary of the leftover hash lemma [41, 43]:

Lemma 4.2 *Let n, ℓ and e be three positive integers such that $3e + 1 < \ell < n$. Let $X \subseteq \{0,1\}^n$ be a set of at least $2^{\ell-1}$ elements and x uniformly distributed in X. Let H be a family of pair-wise independent, $\{0,1\}^n \to \{0,1\}^{\ell-1-3e}$, hash functions. Then for all but a 2^{-e} fraction of $h \in H$ the uniform distribution over $\{0,1\}^{\ell-1-3e}$ 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 is $\ell(n)$-bit long. Let $F = \{F_n\}_{n \in N}$ be as in Construction 4.1 and $\forall n$, let H_n be a family of pair-wise independent, $\{0,1\}^n \to \{0,1\}^{\lfloor \ell(n)/2 \rfloor}$, hash functions. We define the function ensemble $\tilde{F} = \{\tilde{F}_n\}_{n \in N}$. 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 $\{0,1\}^{\lfloor \ell(n)/2 \rfloor}$ is arbitrary. One can choose the range to be $\{0,1\}^{\ell(n)-e(n)}$ for any function $e(n)$ such that $2^{-e(n)}$ is negligible.

Theorem 4.3 *If the DDH-Assumption (Assumption 3.1) holds, then $\tilde{F} = \{\tilde{F}_n\}_{n\in N}$ (as in Construction 4.2) is an efficiently computable pseudo-random function ensemble.*

Proof: The proof easily follows from Theorem 4.1 and Lemma 4.2. From Theorem 4.1 a function $f_{P,Q,g,\vec{a}}$ selected from F_n is indistinguishable from a uniform function with domain $\{0,1\}^n$ and range $\langle g\rangle$. The size of $\langle g\rangle$ is at least $2^{\ell-1}$. Therefore, from Lemma 4.2, for all but a negligible fraction of the hash functions h in H_n, the distribution of $h(x)$ where x is uniform in $\langle g\rangle$ is indistinguishable from the uniform distribution on $\lfloor \ell/2 \rfloor$-bit strings. We can therefore conclude that a distinguisher for $\tilde{F}$ can be used to distinguish F from truly random functions. $\Box$

Remark 4.2 *This proof implies that $\tilde{F} = \{\tilde{F}_n\}_{n\in N}$ 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

There are a few possible approaches to proving Theorem 4.1. One approach is related to the construction of [60] (and in particular to the concept of an n-dimensional synthesizer). Indeed, the construction of [60] has motivated the constructions of this paper (the connection is described in Section 5.2). However, the proof we give here for Theorem 4.1 follows an analogous line to the proof of security for the GGM-Construction of pseudo-random functions [35]. This may seem surprising since the two constructions look very different. Nevertheless, 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 similarities and differences between the two constructions.

Let G be a pseudo-random 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 pseudo-random function $f_s : \{0,1\}^n \rightarrow \{0,1\}^n$ is a uniformly chosen n-bit string s. For any n-bit input, $x = x_1 x_2 \cdots x_n$, 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 definition of f_s can be thought of as a recursive labeling process of a depth-n binary tree. The key s is the label of the root and it induces a labeling of all the nodes in the tree. The labels of the 2^n leaves correspond to the 2^n different outputs of the function.

In contrast, in our construction no tree appears in the design and no particular order is attached to the input bits. Nevertheless, we were able to relate the proof of security of the two constructions.

The DDH-Assumption implies a simple pseudo-random generator that practically doubles its input: $G_{P,Q,g,g^a}(b) \stackrel{\text{def}}{=} \langle g^b, g^{a\cdot b}\rangle$ (whose output is a pseudo-random 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:

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

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

1. $\tilde{G}_{P,Q,g,g^a}$ is efficiently computable (on any input), given the random bits that were used to sample it (in particular, given a).

2. For any $\tilde{G}_{P,Q,g,g^a}$, it is easy to generate the distribution of its output, $\tilde{G}_{P,Q,g,g^a}(g^b)$, on a uniformly chosen input (this fact implies Lemma 4.4).

We now obtain the pseudo-random 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:

$$f_{P,Q,g,a_0,a_1,\ldots,a_n}(x) \stackrel{\text{def}}{=} \tilde{G}^{x_n}_{P,Q,g,g^{a_n}}(\cdots(\tilde{G}^{x_2}_{P,Q,g,g^{a_2}}(\tilde{G}^{x_1}_{P,Q,g,g^{a_1}}(g^{a_0}))\cdots).$$

We turn to the formal proof of Theorem 4.1. First we show (in Lemma 4.4) that a polynomial sample, $\langle \tilde{G}_{P,Q,g,g^a}(g^{b_1}), \ldots \tilde{G}_{P,Q,g,g^a}(g^{b_t})\rangle$ is pseudo-random iff a single sample, $\tilde{G}_{P,Q,g,g^a}(g^b)$, is pseudo-random. In preliminary versions of this paper the proof of Lemma 4.4 used a hybrid-argument based on property (2) above (which is similar to the corresponding argument in [35]). However, Victor Shoup (personal communication) has pointed out that one can use the randomized-reduction of the DDH-Problem (see Section 3.3) for an alternative proof of the lemma. The new proof is both simpler and more security-preserving. Given a distinguisher for the polynomial-sample we get a distinguisher for the single sample that achieves *the same advantage*. Based on this property, the security of the functions in our proof of Theorem 4.1 does not significantly decrease when the *number of queries* the distinguisher makes increases (which is very different from the proofs of security for the functions in [35, 60]).

Definition 4.1 *Let n and t be any pair of positive integers. Define the two distributions $I^{n,t}_{\mathbf{R}}$ and $I^{n,t}_{\mathbf{PR}}$ as follows:*

$$I^{n,t}_{\mathbf{R}} \stackrel{\text{def}}{=} \langle P, Q, g, g^a, g^{b_1}, g^{c_1}, \ldots, g^{b_t}, g^{c_t}\rangle$$

and
$$I^{n,t}_{\mathbf{PR}} \stackrel{\text{def}}{=} \langle P,Q,g,g^a,g^{b_1},g^{a\cdot b_1},\ldots,g^{b_t},g^{a\cdot b_t}\rangle,$$
where $\langle P,Q,g\rangle$ is distributed according to $IG(1^n)$ and all the values in $\langle a,b_1,\ldots,b_t,c_1,\ldots,c_t\rangle$ are uniform in Z_Q.

Lemma 4.4 *(Indistinguishability of a Polynomial Sample) If the DDH-Assumption (Assumption 3.1) holds, then for every probabilistic polynomial-time algorithm $\mathcal{D}$, every polynomial $t(\cdot)$, every constant $\alpha > 0$ and all but a finite number of n's*

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

Moreover, if there exists a probabilistic algorithm with running time $p=p(n)$ that distinguishes $I^{n,t(n)}_{\mathbf{PR}}$ from $I^{n,t(n)}_{\mathbf{R}}$ (as above) with advantage $\epsilon=\epsilon(n)$. Then there exists a probabilistic algorithm with running time $poly(n)\cdot t(n)+p(n)$ that breaks the DDH-Assumption with advantage $\epsilon(n)$.

Proof: It is enough to prove the "moreover" part of the lemma as setting $\epsilon(n)=\frac{1}{n^\alpha}$ it implies the first part of the lemma.

Let $\epsilon=\epsilon(n)$ be any positive real-valued function. Assume that there exists a probabilistic algorithm $\mathcal{D}$ with running time $p=p(n)$ and a polynomial $t(\cdot)$ such that for infinitely many n's
$$\left|\Pr[\mathcal{D}(I^{n,t(n)}_{\mathbf{PR}})=1]-\Pr[\mathcal{D}(I^{n,t(n)}_{\mathbf{R}})=1]\right| > \epsilon(n).$$
We define a probabilistic algorithm $\mathcal{A}$ with running time $poly(n)\cdot t(n)+p(n)$ such that for infinitely many n's

$$\left|\Pr[\mathcal{A}(P,Q,g,g^a,g^b,g^{a\cdot b})=1]-\Pr[\mathcal{A}(P,Q,g,g^a,g^b,g^c)=1]\right| > \epsilon(n),$$

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 Z_Q.

Let the input of $\mathcal{A}$ be $\langle P,Q,g,g^a,g^b,g^{\tilde{c}}\rangle$, where P is n-bit long and $\tilde{c}$ is either $a\cdot b$ or uniform in Z_Q. Using a randomized reduction similar to that in the proof of Lemma 3.2, $\mathcal{A}$ generates $t(n)$ random pairs $g^{b_i},g^{\tilde{c}_i}$ such that $\forall i, \tilde{c}_i=a\cdot b_i$ iff $\tilde{c}=a\cdot b$. $\mathcal{A}$ now invokes $\mathcal{D}$ on these values to distinguish between the two possible distributions of its own input. More formally, $\mathcal{A}$ executes the following algorithm:

1. Define $t=t(n)$ and sample each one of the values in $\langle d_1,\ldots,d_t,e_1,\ldots,e_t\rangle$ uniformly at random in Z_Q.

2. Define the sequence I to be

$$\langle P,Q,g,g^a,\tilde{\mathcal{R}}_{d_1,e_1}(g^a,g^b,g^{\tilde{c}}),\ldots,\tilde{\mathcal{R}}_{d_t,e_t}(g^a,g^b,g^{\tilde{c}})\rangle,$$

   where
$$\forall i,\ \tilde{\mathcal{R}}_{d_i,e_i}(g^a,g^b,g^{\tilde{c}}) \stackrel{\text{def}}{=} \left(g^b\right)^{d_i}\cdot g^{e_i},\ \left(g^{\tilde{c}}\right)^{d_i}\cdot (g^a)^{e_i}.$$

3. Output $\mathcal{D}(I)$

Denote by $g^{b_i}, g^{\tilde{c}_i}$ the value $\tilde{\mathcal{R}}_{d_i,e_i}(g^a, g^b, g^{\tilde{c}})$. By the same arguments used in the proof of Lemma 3.2 we have that:

- If $\tilde{c} = a \cdot b$, then $b_1, \ldots, b_t$ are uniform in Z_Q (and independent of each other and of a) and $\forall i,\ \tilde{c}_i = a \cdot b_i$.
- If $\tilde{c} \neq a \cdot b$, then $b_1, \ldots, b_t, \tilde{c}_1, \ldots, \tilde{c}_t$ are *all* uniform in Z_Q (and independent of each other and of a).

Therefore, by the definitions of $\mathcal{A}$, $I^{n,t}_{\mathbf{PR}}$ and $I^{n,t}_{\mathbf{R}}$ it easily follows that:

$$\begin{array}{ll} & \Pr[\mathcal{A}(P,Q,g,g^a,g^b,g^{a\cdot b}) = 1] = \Pr[\mathcal{D}(I^{n,t}_{\mathbf{PR}}) = 1] \\ \text{and} & \Pr[\mathcal{A}(P,Q,g,g^a,g^b,g^c) = 1] = \Pr[\mathcal{D}(I^{n,t}_{\mathbf{R}}) = 1]. \end{array}$$

It is now immediate that infinitely many n's

$$\left|\Pr[\mathcal{A}(P,Q,g,g^a,g^b,g^{a\cdot b}) = 1] - \Pr[\mathcal{A}(P,Q,g,g^a,g^b,g^c) = 1]\right| > \epsilon(n),$$

where the probabilities are as above. $\Box$

The proof of Theorem 4.1 given Lemma 4.4 uses an hybrid argument, which is a proof-technique for showing that two distributions are indistinguishable. See [33, 34] for details on hybrid arguments. Loosely, the method for showing that D and D' are indistinguishable is to (1) Define a polynomial-length sequence of efficient distributions $D_0, D_1, \ldots, D_m$ with $D_0 = D$ and $D_m = D'$. (2) Show that any two neighboring distributions D_{j-1} and D_j are indistinguishable. In fact, in the uniform version of this argument (e.g. in the proof of Theorem 4.1) we usually show that it is hard to distinguish D_{J-1} and D_J where J is *uniformly chosen* in $[m]$. Furthermore, in the proof of Theorem 4.1 (as well as in the corresponding proofs in [35, 60]) the $n+1$ distributions that are (implicitly) defined are *not efficiently samplable.* For example, one of the two extreme distributions is of uniform functions (which is certainly not efficiently samplable). Nevertheless, a uniform function can be efficiently "simulated" by an algorithm that answers each query at random (under the restriction of keeping consistency of its answers for repeating queries). Since all other intermediate function distributions can be "simulated" in the same sense we can still apply the hybrid argument. We now turn to the formal proof (where the arguments described above are implicit).

Proof: (of Theorem 4.1) It is enough to prove the "moreover" part of the theorem as setting $\epsilon(n) = \frac{1}{n^\alpha}$ it implies the first part of the theorem.

Let $\epsilon = \epsilon(n)$ be any positive real-valued function. Assume that there exists a probabilistic oracle machine $\mathcal{M}$ with running time $t = t(n)$ such that for infinitely many n's

$$\left|\Pr[\mathcal{M}^{f_{P,Q,g,\vec{a}}}(P,Q,g) = 1] - \Pr[\mathcal{M}^{R_{P,Q,g}}(P,Q,g) = 1]\right| > \epsilon(n),$$

where the probabilities are as in Theorem 4.1. We define a probabilistic algorithm $\mathcal{D}$ with running time $poly(n) \cdot t(n)$, such that for infinitely many n's

$$\left|\Pr[\mathcal{D}(I^{n,t(n)}_{\mathbf{PR}}) = 1] - \Pr[\mathcal{D}(I^{n,t(n)}_{\mathbf{R}}) = 1]\right| > \frac{1}{n} \cdot \epsilon(n).$$

By Lemma 4.4, this completes the proof of the theorem.

On any input $\langle P, Q, g, g^a, g^{b_1}, g^{\tilde{c_1}}, g^{b_2}, g^{\tilde{c_2}}, \ldots, g^{b_t}, g^{\tilde{c_t}} \rangle$, where P is n bits long (and either each $\tilde{c}_i$ is $a \cdot b_i$ or each $\tilde{c}_i$ is uniform in Z_Q), $\mathcal{D}$ executes the following algorithm:

1. Sample J uniformly at random in $[n]$.

2. Sample each one of the values in $\langle a_{J+1}, a_{J+2}, \ldots, a_n \rangle$ uniformly at random in Z_Q.

3. Invoke $\mathcal{M}$ on input $\langle P, Q, g \rangle$ and answer its queries in the following way: Let the queries asked by $\mathcal{M}$ be $\langle x^1, x^2, \ldots x^m \rangle$. The i^{th} query x^i is an n-bit string. Denote $x^i = \bar{x}^i x^i_J x^i_{J+1} \cdots x^i_n$, where $\bar{x}^i$ is a $(J-1)$-bit string and $x^i_J, x^i_{J+1}, \ldots, x^i_n$ are single bits. To answer the i^{th} query define $\ell = \ell(i) = \min\{i' | \bar{x}^{i'} = \bar{x}^i\}$ and answer the query by
$$\begin{cases} (g^{\tilde{c}_\ell})^{\prod_{x^i_k=1, k>J} a_k} & \text{if } x^i_J = 1 \\ (g^{b_\ell})^{\prod_{x^i_k=1, k>J} a_k} & \text{if } x^i_J = 0 \end{cases}$$
These answers are well defined since $m \leq t$.

4. Output whatever $\mathcal{M}$ outputs.

From the definition of $\mathcal{D}$ we have that for $f_{P,Q,g,\vec{a}}$ and $R_{P,Q,g}$ as in Theorem 4.1,
$$\Pr[\mathcal{D}(I^{n,t}_{\mathbf{PR}}) = 1 \mid J = 1] = \Pr[\mathcal{M}^{f_{P,Q,g,\vec{a}}}(P, Q, g) = 1],$$
$$\Pr[\mathcal{D}(I^{n,t}_{\mathbf{R}}) = 1 \mid J = n] = \Pr[\mathcal{M}^{R_{P,Q,g}}(P, Q, g) = 1]$$
and for any $0 < j < n$
$$\Pr[\mathcal{D}(I^{n,t}_{\mathbf{R}}) = 1 \mid J = j] = \Pr[\mathcal{D}(I^{n,t}_{\mathbf{PR}}) = 1 \mid J = j+1].$$

By the assumption we get that for infinitely many n's
$$\begin{aligned}
& \left|\Pr[\mathcal{D}(I^{n,t}_{\mathbf{PR}}) = 1] - \Pr[\mathcal{D}(I^{n,t}_{\mathbf{R}}) = 1]\right| \\
= & \left|\frac{1}{n} \cdot \sum_{j=1}^{n} \Pr[\mathcal{D}(I^{n,t}_{\mathbf{PR}}) = 1 \mid J = j] - \frac{1}{n} \cdot \sum_{j=1}^{n} \Pr[\mathcal{D}(I^{n,t}_{\mathbf{R}}) = 1 \mid J = j]\right| \\
= & \frac{1}{n} \cdot \left|\Pr[\mathcal{D}(I^{n,t}_{\mathbf{PR}}) = 1 \mid J = 1] - \Pr[\mathcal{D}(I^{n,t}_{\mathbf{R}}) = 1 \mid J = n]\right| \\
= & \frac{1}{n} \cdot \left|\Pr[\mathcal{M}^{f_{P,Q,g,\vec{a}}}(P, Q, g) = 1] - \Pr[\mathcal{M}^{R_{P,Q,g}}(P, Q, g) = 1]\right| \\
> & \frac{1}{n} \cdot \epsilon(n).
\end{aligned}$$

This completes the proof of the theorem. $\square$

4.3 Efficiency of the Construction

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 (a product of polynomially many numbers), $y = a_0 \cdot \prod_{x_i=1} a_i$ (which can be performed modulo Q), and one modular exponentiation, g^y. This gives a pseudo-random 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. [16]). Actually, to compute the value of the pseudo-random function of Construction 4.2, we also need one application of a pair-wise independent hash function but this operation is very cheap compared with 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-time complexity of the pseudo-random functions. In order to do so, let us first recall the result of Beame, Cook and Hoover [2] who showed that division and related operations *including multiple product* are computable in NC^1. Based on this result, Reif and Tate [65, 66] showed that these operations are also computable in TC^0. The exact depth required for these operations was considered in [71, 72] where 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 (recall that for every integer d the class of functions computable by depth d circuits consisting of a polynomial number of threshold gates is denoted by TC^0_d).

By the results above, we immediately 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 (since all the necessary operations can be performed in TC^0):

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

The exact depth of the functions: As discussed above, Theorem 4.5 can be obtained by a naive application of the results in [71, 72]. In [58], we noted that a more detailed analysis of the function $f_{P,Q,g,\vec{a}}$ implies further optimization in the depth. We described several methods that enable to evaluate this function in TC^0_5 (using additional preprocessing): 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 $2^i \bmod P$. Using similar ideas and a much more careful analysis, Krause and Lucks [46] managed to further reduce the depth to *four*. This is

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

especially interesting as pseudo-random functions cannot be evaluated in TC_2^0 (see [46] for exact statements). This means that the depth required for evaluating the pseudo-random functions of this paper is almost the smallest possible. An natural question which remains open is whether there exist pseudo-random functions in TC_3^0.

Remark 4.3 *Similar analysis holds for efficiency and depth of the pseudo-random functions of Construction 5.1.*

5 Construction Based on Factoring or the GDH-Assumption

In this section we show an additional construction of pseudo-random 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. This construction is interesting 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 so called Blum-integer is not stronger than the assumption that *factoring* Blum-integers is hard. This gives an attractive construction of pseudo-random functions that is at least as secure as Factoring (which was recently improved in [61]).

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 pseudo-random functions based on additional assumptions (in fact, Construction 4.2 was obtained as a modification of Construction 5.1).

5.1 The GDH-Assumption

The GDH-Assumption was previously considered in the context of a key-exchange protocol for a group of parties (see e.g., [69, 74]). In this protocol, party $i \in [n]$ chooses a secret value, a_i. After executing the protocol, each of these parties can compute $g^{\prod_{i\in[n]} a_i}$ and this value defines their common key. While executing the protocol, an eavesdropper may learn values of the form $g^{\prod_{i\in I} a_i}$ for several proper subsets I of $[n]$. It is essential to assume that even with this knowledge it is hard to compute $g^{\prod_{i\in[n]} a_i}$. The GDH-Assumption is even stronger: Informally, this assumption says that it is hard to compute $g^{\prod_{i\in[n]} a_i}$ for an algorithm that can query $g^{\prod_{i\in I} a_i}$ for *any* proper subset, I of $[n]$ of its choice.

To remain consistent with the DDH-Assumption, we state the GDH-Assumption (Assumption 5.1) in a subgroup of Z_P^* of order Q (where P and Q are primes). In fact, the corresponding assumption in any other group implies a corresponding construction of pseudo-random functions. For example, since breaking the GDH-Assumption modulo a composite is at least as hard as factoring [6, 69], we obtain in Section 5.4 a construction of pseudo-random functions which is at least as secure as Factoring. Furthermore, in contrast with the DDH-Assumption, one can consider the GDH-Assumption in Z_P^* itself (i.e., when g is a generator of Z_P^*).

In order to formalize the GDH-Assumption, we use the following definition:

Definition 5.1 *Let $\langle P,Q,g\rangle$ be any possible output of $IG(1^n)$ and let $\tilde{a}=\langle\tilde{a}_1,\tilde{a}_2,\ldots\tilde{a}_n\rangle$ be any sequence of n elements of Z_Q. Define the function $h_{P,Q,g,\tilde{a}}$ with domain $\{0,1\}^n$ such that for any n-bit input, $x=x_1x_2\cdots x_n$,*

$$h_{P,Q,g,\tilde{a}}(x)\stackrel{\text{def}}{=}g^{\prod_{x_i=1}\tilde{a}_i}.$$

Define $h^r_{P,Q,g,\tilde{a}}$ to be the restriction of $h_{P,Q,g,\tilde{a}}$ to inputs $\{0,1\}^n\setminus\{1^n\}$.

Assumption 5.1 (Generalized Diffie-Hellman) *For every probabilistic polynomial-time oracle machine $\mathcal{A}$, every constant $\alpha>0$ and all but a finite number of n's*

$$\Pr[\mathcal{A}^{h^r_{P,Q,g,\tilde{a}}}(P,Q,g)=h_{P,Q,g,\tilde{a}}(1^n)]<\frac{1}{n^\alpha},$$

where the probability is 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 each of the values in $\tilde{a}=\langle\tilde{a}_1,\tilde{a}_2,\ldots\tilde{a}_n\rangle$ uniformly at random in Z_Q.

As a corollary of Theorem 4.1 we have that if the DDH-Assumption holds, then so does the GDH-Assumption. In fact, we get that the DDH-Assumption implies the *decisional* GDH-Assumption (this was also previously shown in [74]):

Corollary 5.1 *If the DDH-Assumption (Assumption 3.1) holds, then for every probabilistic polynomial-time oracle machine $\mathcal{A}$, every constant $\alpha>0$ and all but a finite number of n's*

$$\left|\Pr[\mathcal{A}^{h^r_{P,Q,g,\tilde{a}}}(P,Q,g,h_{P,Q,g,\tilde{a}}(1^n))=1]-\Pr[\mathcal{A}^{h^r_{P,Q,g,\tilde{a}}}(P,Q,g,g^c)=1]\right|<\frac{1}{n^\alpha},$$

where the probabilities are taken over the random bits of $\mathcal{A}$, the choice of $\langle P,Q,g\rangle$ according to the distribution $IG(1^n)$, the choice of each of the values in $\tilde{a}=\langle\tilde{a}_1,\tilde{a}_2,\ldots\tilde{a}_n\rangle$ uniformly at random in Z_Q and the choice of c uniformly at random in Z_Q.

5.2 Motivation to the construction

Construction 5.1 is motivated by the concept of pseudo-random synthesizers and the construction of pseudo-random functions using pseudo-random synthesizers as building blocks [60]. Informally, a pseudo-random 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 pseudo-random synthesizer. Informally, a k-dimensional pseudo-random 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 the 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 [60] can be viewed as first recursively applying a 2-dimensional synthesizer to get an n-dimensional synthesizer, S, and then defining the pseudo-random function, f, by:

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

However, using this construction, the depth of the n-dimensional synthesizer (and the pseudo-random 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 this section 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.

5.3 The Construction

We turn to the construction of pseudo-random functions:

Construction 5.1 *We define the function ensemble $F = \{F_n\}_{n\in N}$. For every n, a key of a function in F_n is a tuple, $\langle P,Q,g,\vec{a},r\rangle$, where P is an n-bit prime, Q a prime divisor of $P-1$, g an element of order Q in Z_P^*, $\vec{a} = \langle a_{1,0},a_{1,1},a_{2,0},a_{2,1},\ldots a_{n,0},a_{n,1}\rangle$ a sequence of $2n$ elements of Z_Q and r an n-bit string. For any n-bit input, $x = x_1x_2\cdots x_n$, the Binary-function, $f_{P,Q,g,\vec{a},r}$, is defined by:*

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

(where $\odot$ denotes the inner product mod 2*). The distribution of functions in F_n is induced by the following distribution on their keys: $\vec{a}$ and r are uniform in their range and the distribution of $\langle P,Q,g\rangle$ is $IG(1^n)$.*

Theorem 5.2 *If the GDH-Assumption (Assumption 5.1) holds, then $F = \{F_n\}_{n\in N}$ (as in Construction 5.1) is an efficiently computable pseudo-random function ensemble.*

In order to prove Theorem 5.2 we need the following corollary of the Goldreich-Levin hard-core-bit theorem [37] (more precisely, the setting of this corollary is somewhat different than the one considered in [37] but their result still applies):

Corollary 5.3 *If the GDH-Assumption (Assumption 5.1) holds, then for every probabilistic polynomial-time oracle machine $\mathcal{A}$, every constant $\alpha > 0$ and all but a finite number of n's*

$$\left|\Pr[\mathcal{A}^{h^r_{P,Q,g,\tilde{a}}}(P,Q,g,r,(h_{P,Q,g,\tilde{a}}(1^n))\odot r)=1] - \Pr[\mathcal{A}^{h^r_{P,Q,g,\tilde{a}}}(P,Q,g,r,\sigma)=1]\right| < \frac{1}{n^\alpha},$$

where the probabilities are taken over the random bits of $\mathcal{A}$, the choice of $\langle P,Q,g\rangle$ according to the distribution $IG(1^n)$, the choice of each of the values in $\tilde{a} = \langle \tilde{a}_1,\tilde{a}_2,\ldots\tilde{a}_n\rangle$ uniformly at random in Z_Q, the choice of r uniformly at random in $\{0,1\}^n$ and the choice of σ uniformly at random in $\{0,1\}$.

Proof:(of Theorem 5.2) Let $F = \{F_n\}_{n\in N}$ be as in Construction 5.1. It is clear that F is efficiently computable. Assume that F is not pseudo-random, then there exists a probabilistic polynomial-time oracle machine $\mathcal{M}$ and a constant $\alpha > 0$ such that for infinitely many n's

$$\left|\Pr[\mathcal{M}^{f_{P,Q,g,\vec{a},r}}(P,Q,g,r) = 1] - \Pr[\mathcal{M}^{R_n}(P,Q,g,r) = 1]\right| > \frac{1}{n^\alpha},$$

where in the first probability, $f_{P,Q,g,\vec{a},r}$ is distributed according to F_n, and in the second probability R_n is uniformly distributed over the set of $\{0,1\}^n \to \{0,1\}$ functions, $\langle P,Q,g\rangle$ is distributed according to $IG(1^n)$ and r is a uniformly chosen n bit string.

Let $t(\cdot)$ be a polynomial that bounds the running time of $\mathcal{M}$. We define a probabilistic polynomial-time oracle machine $\mathcal{A}$ such that for infinitely many n's

$$\left|\Pr[\mathcal{A}^{h^r_{P,Q,g,\tilde{a}}}(P,Q,g,r,(h_{P,Q,g,\tilde{a}}(1^n))\odot r) = 1] - \Pr[\mathcal{A}^{h^r_{P,Q,g,\tilde{a}}}(P,Q,g,r,\sigma) = 1]\right| > \frac{1}{n^\alpha \cdot t(n)},$$

where the probabilities are as in Corollary 5.3. By Corollary 5.3 this would contradict the GDH-Assumption and would complete the proof of the theorem.

Given access to $h^r_{P,Q,g,\tilde{a}}$ and on input $\langle P,Q,g,r,\tilde{\sigma}\rangle$ (where we expect $\tilde{\sigma}$ to either be uniformly chosen or to be $(h_{P,Q,g,\tilde{a}}(1^n))\odot r$), $\mathcal{A}$ executes the following algorithm:

1. Define $t = t(n)$ and sample J uniformly at random in $[t]$.
2. Sample each one of $\langle b_1, b_2, \ldots, b_n\rangle$ uniformly at random in Z_Q.
3. Invoke $\mathcal{M}$ on input $\langle P,Q,g,r\rangle$ and answer its queries in the following way: Let the queries asked by $\mathcal{M}$ be $\langle x^1, x^2, \ldots x^m\rangle$ and assume without loss of generality that all those queries are distinct.
   - Answer each one of the first $J-1$ queries with a uniformly chosen bit.
   - Answer the J^{th} query with $\tilde{\sigma}$.
   - Let x^i be the i^{th} query for $i > J$ and define the n-bit string $z = z_1 z_2 \ldots z_n$ such that z_k is 1 if the k^{th} bit of x^i and the k^{th} bit of x^J are equal and 0 otherwise. Since $x^i \neq x^J$ we have that $z \neq 1^n$. Finally, answer the i^{th} query with

$$\left(\left(h^r_{P,Q,g,\tilde{a}}(z)\right)^{\prod_{z_k=0} b_k}\right)\odot r.$$

4. Output whatever $\mathcal{M}$ outputs.

From the definition of $\mathcal{A}$ we have that all its answers to queries x^i for $i > J$ are $f_{P,Q,g,\vec{a},r}(x^i)$, where $\vec{a} = \langle a_{1,0}, a_{1,1}, a_{2,0}, a_{2,1}, \ldots a_{n,0}, a_{n,1}\rangle$ depends on the J^{th} query $x^J = x^J_1 x^J_2 \ldots x^J_n$ as follows: For every $1 \leq k \leq n$ if $x^J_k = 0$ then $a_{k,0} = \tilde{a}_k$ and $a_{k,1} = b_k$ and if $x^J_k = 1$ then $a_{k,1} = \tilde{a}_k$ and $a_{k,0} = b_k$. The first $J-1$ queries are answered by $\mathcal{A}$ uniformly at random. The only answer that depends on $\tilde{\sigma}$ is the J^{th} answer itself. This answer is of course uniformly distributed in case $\tilde{\sigma}$ is uniform. It is also not hard to verify that the J^{th} answer is $f_{P,Q,g,\vec{a},r}(x^J)$ in case $\tilde{\sigma} = (h_{P,Q,g,\tilde{a}}(1^n))\odot r$. We can therefore conclude that:

$$
\begin{aligned}
& \Pr[\mathcal{A}^{h^r_{P,Q,g,\tilde{a}}}(P,Q,g,r,(h_{P,Q,g,\tilde{a}}(1^n)) \odot r) = 1 \mid J = 1] \\
= \; & \Pr[\mathcal{M}^{f_{P,Q,g,\tilde{a},r}}(P,Q,g,r) = 1],
\end{aligned}
$$

as well as

$$
\begin{aligned}
& \Pr[\mathcal{A}^{h^r_{P,Q,g,\tilde{a}}}(P,Q,g,r,\sigma) = 1 \mid J = t(n)] \\
= \; & \Pr[\mathcal{M}^{R_n}(P,Q,g,r) = 1],
\end{aligned}
$$

and

$$
\begin{aligned}
& \Pr[\mathcal{A}^{h^r_{P,Q,g,\tilde{a}}}(P,Q,g,r,\sigma) = 1 \mid J = j] \\
= \; & \Pr[\mathcal{A}^{h^r_{P,Q,g,\tilde{a}}}(P,Q,g,r,(h_{P,Q,g,\tilde{a}}(1^n)) \odot r) = 1 \mid J = j+1],
\end{aligned}
$$

where the probabilities are as above. Therefore, by the standard hybrid argument we get from the assumption that for infinitely many n's

$$
\left| \Pr[\mathcal{A}^{h^r_{P,Q,g,\tilde{a}}}(P,Q,g,r,(h_{P,Q,g,\tilde{a}}(1^n)) \odot r) = 1] - \Pr[\mathcal{A}^{h^r_{P,Q,g,\tilde{a}}}(P,Q,g,r,\sigma) = 1] \right| > \frac{1}{n^\alpha \cdot t(n)}.
$$

□

Remark 5.1 *From the proof of Theorem 5.2 we get that F is pseudo-random even if the distinguisher (denoted by $\mathcal{M}$ in the proof) has access to P, Q, g and r.*

5.4 Pseudo-Random Functions at Least as Secure as Factoring

The proof of Theorem 5.2 does not rely on the specific group for which the GDH-Assumption is defined. Therefore, the corresponding assumption in any other group implies a corresponding construction of pseudo-random functions. An especially interesting example is taking the GDH-Assumption modulo a composite. Since breaking this assumption is at least as hard as factoring [6, 69], we obtain an attractive construction of pseudo-random functions which is at least as secure as Factoring. As mentioned in the introduction, this construction was recently improved in [61]. In this subsection, we repeat the definition of the GDH-Assumption and the construction of pseudo-random functions with the group set to Z^*_N, where N is a Blum-integer. The proof of security is practically the same as the proof of Theorem 5.2 (and is therefore omitted).

Similarly to the case of the DDH-Assumption, we keep our results general by letting the composite N be generated by *some* polynomial-time algorithm FIG (where FIG stands for factoring-instance-generator). We assume that on input 1^n of FIG its output, N, is distributed over $2n - bit$ integers, where $N = P \cdot Q$ for two $n - bit$ primes, P and Q, such that $P \equiv Q \equiv 3 \bmod 4$ (such an integer is known as a Blum-integer).

The GDH-Assumption Modulo a Composite:

Definition 5.2 *Let N be any possible output of $FIG(1^n)$, let g be any quadratic-residue in Z_N^* and let $\tilde{a} = \langle \tilde{a}_1, \tilde{a}_2, \ldots \tilde{a}_n \rangle$ be any sequence of n elements of $[N]$. Define the function $h_{N,g,\tilde{a}}$ with domain $\{0,1\}^n$ such that for any n-bit input, $x = x_1 x_2 \cdots x_n$,*

$$h_{N,g,\tilde{a}}(x) \stackrel{\text{def}}{=} g^{\prod_{x_i=1} \tilde{a}_i}.$$

Define $h^r_{N,g,\tilde{a}}$ to be the restriction of $h_{N,g,\tilde{a}}$ to inputs $\{0,1\}^n \setminus \{1^n\}$.

Assumption 5.2 (Generalized Diffie-Hellman in Z_N^*) *For every probabilistic polynomial-time oracle machine $\mathcal{A}$, for every constant $\alpha > 0$ and all but a finite number of n's*

$$\Pr[\mathcal{A}^{h^r_{N,g,\tilde{a}}}(N, g) = h_{N,g,\tilde{a}}(1^n)] < \frac{1}{n^\alpha},$$

where the probability is taken over the random bits of $\mathcal{A}$, the choice of N according to the distribution $FIG(1^n)$, the choice of g uniformly at random in the set of quadratic-residues in Z_N^* and the choice of each of the values in $\tilde{a} = \langle \tilde{a}_1, \tilde{a}_2, \ldots \tilde{a}_n \rangle$ uniformly at random in $[N]$.

The Construction and its Security:

Construction 5.2 *We define the function ensemble $F = \{F_n\}_{n \in N}$. For every n, a key of a function in F_n is a tuple, $\langle N, g, \vec{a}, r \rangle$, where N is a $2n$-bit Blum-integer, g is a quadratic-residue in Z_N^*, $\vec{a} = \langle a_{1,0}, a_{1,1}, a_{2,0}, a_{2,1}, \ldots a_{n,0}, a_{n,1} \rangle$ is a sequence of $2n$ values in $[N]$ and r is a $2n$-bit string. For any n-bit input, $x = x_1 x_2 \cdots x_n$, the Binary-function, $f_{N,g,\vec{a},r}$, is defined by:*

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

The distribution of functions in F_n is induced by the following distribution on their keys: $g, \vec{a}$ and r are uniform in their range and the distribution of N is $FIG(1^n)$.

In the same way Theorem 5.2 is proven, we get that:

Theorem 5.4 *If the GDH-Assumption in Z_N^* (Assumption 5.1) holds, then $F = \{F_n\}_{n \in N}$ (as in Construction 5.2) is an efficiently computable pseudo-random function ensemble.*

However, breaking the GDH-Assumption in Z_N^* is at least as hard as factoring N:

Theorem 5.5 *([6, 69]) If the GDH-Assumption in Z_N^* (Assumption 5.1)* does not hold, *then there exists a probabilistic polynomial-time oracle machine $\mathcal{A}$ and a constant $\alpha > 0$ such that for infinitely many n,*

$$\Pr[\mathcal{A}(P \cdot Q) = \langle P, Q \rangle] > \frac{1}{n^\alpha},$$

where the distribution of $N = P \cdot Q$ is $FIG(1^n)$.

Furthermore, the reduction is linear-preserving (see [49]): Assume that there exists a probabilistic algorithm $\mathcal{A}'$ with running-time $t(n)$ that breaks the GDH-Assumption in Z_N^* with probability $\epsilon(n)$. Then there exists a probabilistic algorithm $\mathcal{A}$ with running-time $t(n) \cdot poly(n)$ for factoring with success-probability $\epsilon(n)$.

We can therefore deduce that:

Corollary 5.6 *(of Theorem 5.4 and Theorem 5.5) Let $F = \{F_n\}_{n \in N}$ be as in Construction 5.2 and assume that F is* not *an efficiently computable pseudo-random function ensemble. Then there exists a probabilistic polynomial-time algorithm $\mathcal{A}$ and a constant $\alpha > 0$ such that for infinitely many n's:*

$$\Pr[\mathcal{A}(P \cdot Q) = \langle P, Q \rangle] > \frac{1}{n^\alpha},$$

where the distribution of $N = P \cdot Q$ is $FIG(1^n)$.

6 Additional Features and Further Research

This paper shows two, very efficient, constructions of pseudo-random functions. The first construction is based on the *decisional* DH-Assumption (Assumption 3.1) and the second construction is based on a generalization of the *computational* DH-Assumption (Assumption 5.1). 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.

The pseudo-random functions of Constructions 4.2 and 5.1 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 pseudo-random functions. In [58] we presented preliminary results in obtaining such features for our construction of pseudo-random functions: (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 pseudo-random 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 "oblivious evaluation" of the value of the function: Assume that a party, $\mathcal{A}$, knows a key s of a pseudo-random 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). Though there is much room for improving these designs, they are still a significant improvement over the protocols that are available for all previous constructions of pseudo-random functions (including commonly used block-ciphers such as DES) and they serve as a demonstration to the potential of our construction.

We consider the task of improving the protocols given in [58] and designing additional ones to be an interesting line for further research. A particularly interesting example arises by the work of Bellare and Goldwasser [3]. They suggest a way to design a digital-signature scheme that is very attractive given efficient pseudo-random 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 pseudo-random function f_s is available as part of the public-key). Another very attractive scheme one may desire is a *function-sharing* scheme for pseudo-random functions (in an analogous meaning to function-sharing schemes for trapdoor one-way permutations as defined in [25]). Two examples for applications of such schemes are efficient metering of web usage [55] and the distribution of KDCs (key-distribution centers) [57].

In Section 5.2 the concept of a k-dimensional pseudo-random synthesizer and the immediate construction of pseudo-random functions from n-dimensional synthesizers are described. Assumption 5.1 gives a simple construction of an n-dimensional synthesizer which indeed translates to a construction of pseudo-random functions (Construction 5.1). An interesting problem is to construct efficient n-dimensional synthesizers using other intractability assumptions.

Acknowledgments

We thank Ran Canetti, Joe Kilian, Kobbi Nissim and Amnon Ta-Shma for many helpful discussions and comments. Special thanks to Victor Shoup for suggesting the improved proof of Lemma 4.4 and for bringing [73] to our attention. Finally, we would like to thank the anonymous referees for many helpful comments.

References

[1] D. 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] Ben-Or M., Goldwasser S. and Wigderson A., Completeness theorems for non-cryptographic fault tolerant distributed computation, *Proc. 20th Ann. ACM Symp. on Theory of Computing*, 1988, pp. 1-9.

[6] E. Biham, D. Boneh and O. Reingold, Breaking generalized Diffie-Hellman modulo a composite is no easier than Factoring, *Theory of Cryptography Library*, Record 97-14 at: `http://theory.lcs.mit.edu/~tcryptol/homepage.html`

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

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

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

[10] 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.

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

[12] 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.

[13] 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.

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

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

[16] 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.

[17] R. Canetti, Towards realizing random oracles: hash functions that hide all partial information, *Proc. Advances in Cryptology - CRYPTO '97*, LNCS, Springer, 1997 pp. 455-469.

[18] R. Canetti, J. Friedlander and I. Shparlinski, On certain exponential sums and the distribution of Diffie-Hellman triples, *Research report, IBM T. J. Watson Research Center, Number RC 20915 (92645)*, July 1997.

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

[20] D. Chaum, C. Crepeau. and I. Damgard, Multiparty unconditionally secure protocols, *Proc. 20th Ann. ACM Symp. on Theory of Computing*, 1988, pp. 11-19.

[21] D. Chaum, and T. Pederson, "Wallet databases with observers", Advances in Cryptology - CRYPTO' 92, LNCS, 1992, pp. 89–105.

[22] B. Chor, A. Fiat and M. Naor, Tracing traitors, *Advances in Cryptology - CRYPTO' 94*, LNCS vol. 839, Springer, 1994, pp. 257-270.

[23] R. Cramer, I. Damgård, B. Schoenmakers, Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols, *Advances in Cryptology - CRYPTO' 94*, LNCS vol. 839, Springer, 1994, pp. 174–187.

[24] R. Cramer and V. Shoup, A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack, *Advances in Cryptology - CRYPTO '98*, LNCS vol. 1462, Springer, 1998, pp. 13-25.

[25] A. De Santis, Y Desmedt, Y. Frankel and M. Yung, How to share a function securely, *Proc. 26th ACM Symp. on Theory of Computing*, 1994, pp. 522-533.

[26] Y. Desmedt, Society and group oriented cryptography: A new concept *Proc. Advances in Cryptology - CRYPTO '87*, LNCS, Springer, 1987.

[27] Y. Desmedt and Y. Frankel, Threshold cryptosystems, *Advances in Cryptology – CRYPTO '89*, LNCS vol. 435, Springer, 1989, pp. 307–315.

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

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

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

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

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

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

[34] O. Goldreich, **Modern cryptography, probabilistic proofs and pseudo-randomness**. *Algorithms and Combinatorics*, vol. 17, Springer-Verlag, 1998.

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

[36] 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.

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

[38] O. Goldreich O., S. Micali and A. Wigderson, How to play any mental game, *Proc. 19th Ann. ACM Symp. on Theory of Computing*, 1987, pp. 218-229.

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

[40] O. Goldreich and R. Ostrovsky, Software Protection and Simulation on Oblivious RAMs. J. of the ACM vol. 43(3), 1996, pp. 431–473.

[41] J. Hastad, R. Impagliazzo, L. A. Levin and M. Luby, Construction of a pseudo-random generator from any one-way function, *SIAM Journal on Computing*, vol 28(4), 1999, pp. 1364-1396.

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

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

[44] 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.

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

[46] M. Krause and S. Lucks, On the minimal hardware complexity of pseudorandom function generators, *18th Annual Symposium on Theoretical Aspects of Computer Science*, 2001, pp. 419-430.

[47] M. Langberg, An implementation of efficient pseudo-random functions, 1998. At: `http://www.wisdom.weizmann.ac.il/~naor/p_r_func/abs/abs.html`

[48] 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.

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

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

[51] U. Maurer and S. Wolf, Towards the equivalence of breaking the Diffie-Hellman protocol and computing discrete logarithms, *SIAM Journal on Computing*, vol 28(5), 1999, pp. 1689–1721.

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

[53] K. McCurley, The discrete logarithm problem,*Cryptography and Computational Number Theory, Proc. Symp. Appl. Math.*, AMS Lecture Notes, vol. 42, 1990, pp. 49-74.

[54] M. Naor, Bit commitment using pseudorandomness, *J. of Cryptology* 4(2), 1991, pp. 151–158.

[55] M. Naor and B. Pinkas, Secure and efficient metering, *Advances in Cryptology - EUROCRYPT '98*, LNCS vol. 1462, Springer, 1998.

[56] M. Naor, B. Pinkas, Oblivious Transfer and Polynomial Evaluation. Proc. 31st ACM Symp. on Theory of Computing, 1999, pp. 245–254.

[57] M. Naor, B. Pinkas and O. Reingold, *Distributed Pseudo-Random Functions and KDCs*, Advances in Cryptology - Eurocrypt '99, LNCS vol. 1592, Springer, 1999, pp. 327–346.

[58] M. Naor and O. Reingold, Number-Theoretic constructions of efficient pseudo-random functions, *Proc. 38th IEEE Symp. on Foundations of Computer Science*, 1997, pp. 458-467. Full version at `http://www.wisdom.weizmann.ac.il/%7Enaor/PAPERS/gdh_abs.html`

[59] M. Naor and O. Reingold, On the construction of pseudo-random permutations: Luby-Rackoff revisited, *J. of Cryptology, vol. 12(1)*, 1999, pp. 29–66. (Prelim. version STOC'97.)

[60] M. Naor and O. Reingold, Synthesizers and their application to the parallel construction of pseudo-random functions, *J. Comput. System Sci.*, vol. 58(2), 1999, pp. 336–375. (Prelim. version FOCS'95).

[61] M. Naor, O. Reingold and A. Rosen, Pseudo-random functions and Factoring, *Proc. 32nd ACM Symp. on Theory of Computing*, 2000.

[62] A. M. Odlyzko, Discrete logarithms and smooth polynomials, *Contemporary Mathematics, AMS* 1993.

[63] B. Pinkas, private communication.

[64] A. Razborov and S. Rudich, Natural proofs, *J. of Computer and System Sciences*, vol. 55(1), 1997, pp. 24–35.

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

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

[67] C. P. Schnorr, Efficient identification and signatures for smart cards, *Advances in Cryptology - CRYPTO '89*, LNCS vol. 435, Springer, 1990, pp. 239–252.

[68] A. Shamir, How to share a secret, *Comm. ACM* vol. 22(11), 612-613.

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

[70] V. Shoup, Lower bounds for discrete logarithms and related problems, *Advances in Cryptology - EUROCRYPT '97*, LNCS vol. 1233, Springer, 1997, pp. 256–266.

[71] 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.

[72] 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.

[73] M. Stadler, Publicly verifiable secret sharing, *Proc. Advances in Cryptology - EUROCRYPT '96*, LNCS vol. 1070, Springer, 1996, pp. 190–199.

[74] 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.

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

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