Number-Theoretic Constructions of Efficient Pseudo-Random Functions

Preliminary Version

Moni Naor * Omer Reingold †

Abstract

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

*Incumbent of the Morris and Rose Goldman Career Development Chair, Dept. of Applied Mathematics and Computer Science, Weizmann Institute of Science, Rehovot 76100, Israel. Research supported by grant no. 356/94 from the Israel Science Foundation administered by the Israeli Academy of Sciences and by BSF grant no. 94-00032. E-mail: naor@wisdom.weizmann.ac.il.

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

1 Introduction

This paper studies the efficient construction of several fundamental cryptographic primitives and reduces their security to the *decisional* version of the Diffie-Hellman assumption (**DDH-Assumption**). Our major construction is of pseudo-random functions. We also show a simple randomized-reduction between the worst-case and average-case of the DDH-Assumption that may increase our confidence in this assumption. In addition, we show a related construction of pseudo-random functions that is at least as secure as the *generalized* DH-Assumption (**GDH-Assumption**). Breaking the GDH-Assumption modulo a composite would imply an efficient algorithm for factorization (see [5, 53]). Therefore, we get an attractive construction of pseudo-random functions that is at least as secure as factoring.

Pseudo-random functions were introduced by Goldreich, Goldwasser and Micali [28] and have innumerable applications (e.g., [3, 7, 19, 25, 26, 29, 39, 44, 46]). 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. Our new construction of pseudo-random functions is:

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

Shallow Given appropriate preprocessing, the value of the functions at any given point can be computed in TC^0, compared with TC^1 previously (in [43]). 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 [47]) and to computational learning-theory.

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

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

The DDH-Assumption

In the following few paragraphs we briefly describe the DDH-Assumption, its different applications and the current knowledge on its security. A more detailed description appears in Section 3.1.

The DH-Assumption was introduced in the context of the Diffie and Hellman [21] key-exchange protocol (among quite a few of the fundamental ideas and concepts of public-key

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

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

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

It turns out that the DDH-Assumption was assumed in quite a few previous works (both explicitly and implicitly). All these applications rely on the average-case assumption described above. In Section 3.3, we show that for any given P and Q the DDH-assumption can be reduced to its worst-case version[1]:

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

The randomized reduction we describe may strengthen our confidence in the DDH-Assumption and in the security of its many applications. Additional evidence to the validity of the DDH-Assumption lies in the fact that it endured the extensive research of the related CDH-Assumption. To some extent, the DDH-Assumption is also supported by the results on the strength of the CDH-Assumption in several groups [12, 40, 41, 53] and by additional results [12, 17, 54]. For instance, Shoup [54] showed that the DDH-Problem is hard for any "generic"-algorithm. However, a main conclusion of this paper is that the DDH-Assumption deserve more attention since it implies the security of many attractive cryptographic constructions.

The most obvious application of the DDH-Assumption is to the Diffie-Hellman key-exchange protocol and to the related public-key cryptosystem [22]. 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 get a probabilistic encryption-scheme whose semantic security (see [31]) is equivalent to the DDH-Assumption[2]. The price of encrypting many bits using the ElGamal cryptosystem is a single (or two) exponentiation. This is comparable with the Blum-Goldwasser cryptosystem [9]. Other applications that previously appeared are [4, 13, 16, 23, 55].

To previous applications one can add a pseudo-random generator [10, 57] that practically doubles the input length and a pseudo-random synthesizer (see definition in [43]) 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.

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

[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.

The pseudo-random function is defined by $n+1$ values, $\langle a_0, a_1 \ldots a_n \rangle$, chosen uniformly in $\mathbb{Z}_Q$. The value of the function on an n-bit input $x = x_1 x_2 \cdots x_n$ is essentially

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

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

The GDH-Assumption

In Section 5, we suggest a related construction of 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., [53, 55]). The GDH-Assumption is implied by the DDH-Assumption (as shown in [55] and in this paper) but the assumptions are not known to be equivalent. In addition, the GDH-Assumption modulo a composite is not stronger than the assumption that factoring is hard (see [5, 53]). This implies an attractive construction of pseudo-random functions such that their security can be reduced to factoring:

Let N be distributed over Blum-integers ($N = P \cdot Q$, where P and Q are primes and $P = Q = 3 \bmod 4$) and assume that (under this distribution) it is hard to factor N. Let g be a uniformly distributed quadratic residue in $\mathbb{Z}_N^*$, 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. Where 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:

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

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

Previous Work

In addition to introducing pseudo-random functions, Goldreich, Goldwasser and Micali [28] provided a construction of such functions (GGM-Construction) based on pseudo-random generators. Naor and Reingold [43] 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 [43] 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 [33], based on the hardness of subset sum (and factoring), and by Blum et. al. [8], based on hard-to-learn problems.

The construction of this paper is not only more parallelizable than the concrete constructions based on [28, 43], but it is also much more efficient. Though this construction seems very different than the constructions of [28, 43], we were able to relate the proof of security of this construction to both [28] and [43] (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 [24]. 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 CDH-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 pseudo-random functions. In Section 7, we suggest directions for further research.

2 Preliminaries

2.1 Notation and Conventions

- For any integer N the multiplicative group modulo N is denoted by $\mathbb{Z}_N^*$ and the additive group modulo N is denoted by $\mathbb{Z}_N$.
- For any integer k, denote by $[k]$ the set of integers — $\{1, 2, \ldots, k\}$. For any two 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 $x \odot y$.
- All exponentiations in this paper (apart of Section 5.4) are in $\mathbb{Z}_P^*$ (the definition of P will be clear by the context).

2.2 Pseudo-Random Functions

As mentioned in the introduction, our main result is a construction of a pseudo-random function that is *efficient, shallow and simple*. We devote this section to motivating such a construction and to describing previous constructions and applications of pseudo-random functions. Good references on pseudo-random functions and pseudo-randomness in general are Goldreich [27] and Luby [38].

The concept of a pseudo-random function-ensemble was introduced by Goldreich, Goldwasser and Micali [28]. 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 [10, 57] as building blocks, which in turn can

be obtained from any one-way function (as shown by Hastad, Impagliazzo, Levin and Luby [32]).

A pseudo-random function is a powerful cryptographic primitive that can replace uniformly-chosen functions 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 [14, 29, 38]. As shown by Luby and Rackoff [39], it is possible to efficiently construct pseudo-random *permutations* (which, in particular, can be used as block-ciphers) from pseudo-random functions (also see [44] 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 to construct digital-signatures. For some of the additional applications of pseudo-random functions see [7, 19, 25, 26, 46].

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 [43] 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 (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 [43], 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 [43] 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 pseudo-random functions imply reduced latency in computing those functions which, for some applications (as the encryption of messages on a network), is essential.

- As was observed by Valiant [56], 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, 36, 35]. It is also interesting to compare with the result of Linial, Mansour and Nisan [37] who showed that AC^0 *can* be efficiently learned.

- Another application of pseudo-random functions in complexity was suggested by Razborov and Rudich [47]. 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 known lower bound techniques) for separating this class from $P/poly$. We therefore get from our construction that *if the*

GDH-Assumption holds against a subexponential-time adversary (and in particular if factoring is hard), then there are no Natural Proofs for separating TC^0 from $P/poly$.

We note that one can extract a similar result (assuming the hardness of factoring) from the work of Kharitonov [36], which is based on the pseudo-random generator of Blum, Blum and Shub [6].

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. 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 [43]. 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 [20]. It seems that for such properties to be possible we need a simple construction of pseudo-random functions as the one introduced in this paper.

   In Section 6 we consider some of the possible features of pseudo-random functions. We also present preliminary results in this direction 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 "blind computation" 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 [27]:

Definition 2.1 (efficiently computable pseudo-random function ensemble) *Let $\{A_n, B_n\}_{n\in\mathbb{N}}$ be a sequence of domains and let $F = \{F_n\}_{n\in\mathbb{N}}$ be a function ensemble such that the random variable F_n assumes values in the set of $A_n \mapsto B_n$ functions. Then F is called* 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 sufficiently large 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 $R = \{R_n\}_{n\in\mathbb{N}}$ is the corresponding uniform function ensemble (i.e., $\forall n$, R_n is uniformly distributed over the set of $A_n \mapsto B_n$ 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 (see e.g. Remark 4.1). For a discussion on security preserving reductions see [38].*

3 The Decisional Diffie-Hellman Assumption

3.1 Background

As mentioned in the introduction, the DH-Assumption was introduced in the context of the Diffie and Hellman [21] 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 $\mathbb{Z}_P^*$, the Diffie-Hellman key-exchange protocol goes as follows: $\mathcal{A}$ chooses an integer a uniformly at random in $[P-2]$ and sends g^a to $\mathcal{B}$.[3] In return $\mathcal{B}$ chooses an integer b uniformly at random in $[P-2]$ and sends g^b to $\mathcal{A}$. Both $\mathcal{A}$ and $\mathcal{B}$ can now compute $g^{a\cdot b}$ and their common key, $K_{\mathcal{A},\mathcal{B}}$, is defined by $g^{a\cdot b}$ in some public way. For this protocol to be secure we must have, at the minimum, that the CDH-Assumption holds:

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

[3] Recall that all exponentiations here and in the rest of the paper (apart of Section 5.4) are in $\mathbb{Z}_P^*$ (the definition of P will be clear by the context).

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 [30] hard-core function[4] to $g^{a \cdot b}$ (an important improvement on the security of such an application was made by Shoup [54]). If the CDH-Assumption holds, then this method indeed gives a pseudo-random key. However, the proof in [30] only implies the pseudo-randomness of the key in case its length is at most logarithmic in the security parameter. A much more ambitious method is to take $g^{a \cdot b}$ itself as the key. For instance, in the ElGamal cryptosystem, given the public key g^a the encryption of a message m is $\langle g^b, g^{a \cdot b} \cdot m \rangle$. The security of the key-exchange protocol now relies on the DDH-Assumption:

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

However, when g is a generator of $\mathbb{Z}_P^*$, we have that g^a and g^b do give some information on $g^{a \cdot b}$. For example, if either g^a or g^b is a quadratic residue, then so is $g^{a \cdot b}$. A standard solution for this problem is to take g to be a generator of the subgroup of $\mathbb{Z}_P^*$ of order Q, where Q is a large prime divisor of $P - 1$. In fact, for most applications, taking g to be an element of order Q is an advantage. This is the case since all known subexponential algorithms for computing the discrete-log will be subexponential in the length of P even when applied to the subgroup of size Q generated by g (see, [42, 45] for surveys on algorithms for the discrete-log; the best known algorithm for general groups has time square root of the size of the group). Therefore, we can use a rather small prime Q (say, 160 bits long) which results in a substantial improvement in efficiency.

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 [40] and Boneh and Lipton [11] show that in several settings these two problems are in fact equivalent. For example, Maurer showed that given some information which only depends on P and an efficient algorithm for computing the DH-Problem in $\mathbb{Z}_P^*$, one can efficiently compute the discrete-log in $\mathbb{Z}_P^*$ (so in some nonuniform sense these problems are equivalent). Shoup [54] showed that there are no efficient "generic"-algorithms for computing the discrete-log or the DH-Problem, where 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.

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

To some extent, the DDH-Assumption is supported by the work of Shoup [54] and the work of Boneh and Venkatesan [12]. Shoup showed that the DDH-Problem is hard for any "generic"-algorithm (as 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 [17].

We have discovered an attractive feature of the DDH-Problem: There is quite a simple randomized reduction between its worst-case and its average-case. Such a reduction may strengthen our belief in the DDH-Assumption since it implies that:

> *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 $\mathbb{Z}_P^*$, a, b and c are uniformly chosen in $\mathbb{Z}_Q$.*
> - *It is easy to decide with overwhelming success probability for $\langle P, Q, g, g^a, g^b, g^c \rangle$ whether or not $c = a \cdot b \bmod Q$. Where a, b and c are* any *three elements in $\mathbb{Z}_Q$ and g is* any *element of order Q in $\mathbb{Z}_P^*$.*

For most applications of the DDH-Assumption (including ours) there is no reason to insist on working in a subgroup of $\mathbb{Z}_P^*$ (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) $\mathbb{Z}_N^*$ where N is a composite. McCurley and Shmuely [41, 53] 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 does not work if the order of g needs to be a secret (as is the case for $\mathbb{Z}_N^*$).

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 [31]) 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 [9]. 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 [13] pointed out that several suggestions for undeniable signatures (as the one in [18] where this concept was introduced) implicitly rely on the DDH-Assumption. If this assumption does not hold then such schemes are in fact *digital* signatures.

- Canetti [16] gave a simple construction based on the DDH-Assumption for a new cryptographic primitive called "Oracle Hashing". Loosely, these are hash functions that "hide all partial information" on their input.

- Franklin and Haber [23] showed a construction of a joint encryption scheme based on the the DDH-Assumption modulo a *composite*. Using this scheme they showed how to get an efficient protocol for secure circuit computation.

- Steiner, Tsudik and Waidner [55] 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).

To previous applications we can add:

- A pseudo-random generator that practically doubles the input length. Essentially, the generator is defined by $G_{P,g,g^a}(b) = \langle g^b, g^{a \cdot b} \rangle$. This gives a pseudo-random pair of values in the subgroup generated by g. In order to get a pseudo-random value in $\{0,1\}^\ell$, for ℓ of approximately twice the length of b, 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. 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 [43]) whose output length is similar to its arguments length, defined by $S_{P,g}(a,b) = g^{a \cdot b}$.

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 $\mathbb{Z}_P^*$).

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

Q and g be generated by *some* polynomial-time algorithm IG (where IG stands for instance generator).

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

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 weakened if g (generated by IG) is taken to be a uniformly chosen element of order Q in $\mathbb{Z}_P^*$.

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

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

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 $\mathbb{Z}_Q$.*

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.

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

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

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

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

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

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

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

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

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

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

Our randomized reduction is closely related to other known reductions. Blum and Micali [10] showed that for any specific prime P and generator g, the discrete-log problem is random-self-reducible: given $\langle P, g, g^a \rangle$ for *any* a it is easy to generate a random instance $\langle P, g, g^{a+r} = g^a \cdot g^r \rangle$ (where r is uniform in $[P-1]$). Given the solution for the random instance (i.e., $a+r$) it is easy to compute the solution for the original instance (i.e., a). A similar property was shown for the CDH-Problem (e.g. Maurer [40]): given $\langle P, g, g^a, g^b \rangle$ for *any* a and b it is easy to generate a random instance $\langle P, g, g^{a+r}, g^{b+s} \rangle$ (where r and s are uniform in $[P-1]$). Given the solution for the random instance (i.e., $z = g^{(a+r)\cdot(b+s)}$) it is easy to compute the solution for the original instance (i.e., $g^{a\cdot b} = z \cdot (g^a)^{-s} \cdot (g^b)^{-r} \cdot g^{-s\cdot r}$).

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

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

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

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

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

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

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

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

and outputs

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

Let $c = a \cdot b + e \bmod Q$ for e in $\mathbb{Z}_Q$ then:

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

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

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 3.2. 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 $\mathbb{Z}_Q$.

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 get that for *any* a, b and $c \neq a \cdot b \bmod Q$ in $\mathbb{Z}_Q$:

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

Now the probabilities are *only* taken over the random bits of $\mathcal{A}$ and $\mathcal{R}$. Therefore, by standard methods of amplification we can define a probabilistic algorithm $\mathcal{A}'$ such that for *any* a, b and $c \neq a \cdot b \bmod Q$ in $\mathbb{Z}_Q$:

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

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

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

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

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

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

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 $\mathbb{Z}_P^*$ generated by g).

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

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

Given Theorem 4.1, we have that F is "almost" an efficiently computable 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 set 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 [32, 34]:

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

Lemma 4.2, suggests the following construction:

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

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

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

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

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

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

4.2 Proof of Security

The proof of Theorem 4.1 bares a close resemblance to the proof of security for the GGM-Construction of pseudo-random functions [28]. This may seem surprising since, in our construction, no tree is described and the order of the input bits does not really matter. However, in some sense, one may view our construction as a careful application (or a generalization) of the GGM-Construction. In the following few paragraphs we describe the resemblance and differences between the two constructions. However, since we cannot get Theorem 4.1 as a corollary of the GGM-Theorem, we include a complete proof. We note that there is also a close relation between the construction of this section and the construction of [43] (which indeed motivated our construction as described in Section 5.2).

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 $f_s : \{0,1\}^n \mapsto \{0,1\}^n$ is a uniformly chosen n-bit string, s. For any n-bit input, $x = x_1x_2\cdots x_n$, the function f_s is defined by:

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

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

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

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

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

2. For any G_{P,g,g^a}, it is easy to generate the distribution of its output, $G_{P,g,g^a}(g^b)$, on a uniformly chosen input, g^b (we use this fact in the proof of Lemma 4.4).

We now get the 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:

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

We turn to the formal proof of Theorem 4.1. The proof makes extensive usage of hybrid-arguments (both in the proof of Lemma 4.4 and in the proof of Theorem 4.1 given Lemma 4.4) which is quite a standard proof-technique by now. Informally, this technique suggests a method of showing that two distributions are indistinguishable: (1) Define a polynomial-length sequence of distributions such that the two extreme distributions are the original ones. (2) Show that any two neighboring distributions are indistinguishable. More details can be found in [27].

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

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

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

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

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

Proof: (of Lemma 4.4) Assume that there exists a probabilistic polynomial-time algorithm $\mathcal{D}$, a polynomial $t(\cdot)$ and a constant $\alpha > 0$ such that for infinitely many n's

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

We define a probabilistic polynomial-time algorithm $\mathcal{A}$ such that for infinitely many n's

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

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 $\mathbb{Z}_Q$. This would contradict the DDH-Assumption and would complete the proof of the lemma.

On any input $\langle P, Q, g, g^a, g^b, g^{\tilde{c}}\rangle$, where P is n-bit long (and we expect $\tilde{c}$ to either be $a \cdot b \bmod Q$ or uniform in $\mathbb{Z}_Q$), $\mathcal{A}$ executes the following algorithm:

1. Define $t = t(n)$ and sample J uniformly at random in $[t]$.
2. Sample each one of the values in $\langle b_1, \ldots, b_{J-1}, b_{J+1}, \ldots, b_t, c_{J+1}, \ldots, c_t\rangle$ uniformly at random in $\mathbb{Z}_Q$.

3. Define the sequence I to be

$$\langle P, Q, g, g^a, g^{b_1}, g^{a \cdot b_1}, \ldots, g^{b_{J-1}}, g^{a \cdot b_{J-1}}, g^b, g^{\tilde{c}}, g^{b_{J+1}}, g^{c_{J+1}}, \ldots, g^{b_t}, g^{c_t} \rangle$$

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

It is immediate from the definition of $I_i^{n,t}$ and $\mathcal{A}$ that

$$\Pr[\mathcal{A}(P, Q, g, g^a, g^b, g^{a \cdot b}) = 1 \mid J = j] = \Pr[\mathcal{D}(I_j^{n,t}) = 1]$$

and

$$\Pr[\mathcal{A}(P, Q, g, g^a, g^b, g^c) = 1 \mid J = j] = \Pr[\mathcal{D}(I_{j-1}^{n,t}) = 1]$$

where $\langle P, Q, g \rangle$ is distributed according to $IG(1^n)$ and a, b and c are uniform in $\mathbb{Z}_Q$. Therefore, from our assumption, we get (by the standard hybrid-argument) that for infinitely many n's

$$\begin{aligned}
& \left| \Pr[\mathcal{A}(P, Q, g, g^a, g^b, g^{a \cdot b}) = 1] - \Pr[\mathcal{A}(P, Q, g, g^a, g^b, g^c) = 1] \right| \\
= \; & \frac{1}{t} \cdot \left| \sum_{j=1}^{t} \Pr[\mathcal{D}(I_j^{n,t}) = 1] - \sum_{j=1}^{t} \Pr[\mathcal{D}(I_{j-1}^{n,t}) = 1] \right| \\
= \; & \frac{1}{t} \cdot \left| \Pr[\mathcal{D}(I_t^{n,t}) = 1] - \Pr[\mathcal{D}(I_0^{n,t}) = 1] \right| \\
> \; & \frac{1}{t \cdot n^{\alpha}}
\end{aligned}$$

where $\langle P, Q, g \rangle$ is distributed according to $IG(1^n)$ and a, b and c are uniform in $\mathbb{Z}_Q$. □

Proof: (of Theorem 4.1) Assume that there exists a probabilistic polynomial-time oracle machine $\mathcal{M}$ and a constant $\alpha > 0$, such that for infinitely many n's

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

where the probabilities are as in Theorem 4.1. Let $t(\cdot)$ be a polynomial that bounds the running time of $\mathcal{M}$. We define a probabilistic polynomial-time algorithm $\mathcal{D}$, such that for infinitely many n's

$$\left| \Pr[\mathcal{D}(I_{t(n)}^{n,t(n)})) = 1] - \Pr[\mathcal{D}(I_0^{n,t(n)})) = 1] \right| > \frac{1}{n^{\alpha+1}}$$

By Lemma 4.4, this would contradict the DDH-Assumption and would complete 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-bit long (and we expect each $\tilde{c}_i$ to either be $a \cdot b_i \bmod Q$ or uniform in $\mathbb{Z}_Q$), $\mathcal{D}$ 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 $\mathbb{Z}_Q$.

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 is an n-bit string $x^i = \bar{x}^i x^i_J x^i_{J+1} \cdots x^i_n$, 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,

$$\Pr[\mathcal{D}(I^{n,t}_t) = 1 \mid J = 1] = \Pr[\mathcal{M}^{f_{P,Q,g,\vec{a}}}(P, Q, g) = 1],$$

$$\Pr[\mathcal{D}(I^{n,t}_0) = 1 \mid J = n] = \Pr[\mathcal{M}^{R_{P,Q,g}}(P, Q, g) = 1]$$

and for any $0 < j < n$

$$\Pr[\mathcal{D}(I^{n,t}_0) = 1 \mid J = j] = \Pr[\mathcal{D}(I^{n,t}_t) = 1 \mid J = j+1]$$

By the assumption we get that for infinitely many n's

$$\begin{aligned} & \left|\Pr[\mathcal{D}(I^{n,t}_t) = 1] - \Pr[\mathcal{D}(I^{n,t}_0) = 1]\right| \\ = & \frac{1}{n} \cdot \left| \sum_{j=1}^{n} \Pr[\mathcal{D}(I^{n,t}_t) = 1 \mid J = j] - \sum_{j=1}^{n} \Pr[\mathcal{D}(I^{n,t}_0) = 1 \mid J = j] \right| \\ = & \frac{1}{n} \cdot \left|\Pr[\mathcal{D}(I^{n,t}_t) = 1 \mid J = 1] - \Pr[\mathcal{D}(I^{n,t}_0) = 1 \mid J = n]\right| \\ = & \frac{1}{n} \cdot \left|\Pr[\mathcal{M}^{f_{P,Q,g,\vec{a}}}(P, Q, g) = 1] - \Pr[\mathcal{M}^{R_{P,Q,g}}(P, Q, g) = 1]\right| \\ > & \frac{1}{n^{\alpha+1}} \end{aligned}$$

□

4.3 Efficiency of the Pseudo-Random Functions

Consider a function $f_{P,Q,g,\vec{a}} \in F_n$ (where $\vec{a} = \langle a_0, a_1, \ldots a_n\rangle$) as in Construction 4.1. Computing the value of this function at any given point, x, involves one multiple product, $y = a_0 \cdot \prod_{x_i=1} a_i$ (which can be performed modulo Q), and one modular exponentiation, g^y. This gives a 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

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

Brickell et. al. [15]). Actually, in order to get the value of the pseudo-random function as in Construction 4.2, we also need one application of a pair-wise independent hash function but this operation is much cheaper than a multiple product or a modular exponentiation.

As described in the Introduction and in Section 2.2, we are also interested in finding the parallel complexity of the pseudo-random functions. In order to do so, let us first recall the result of Beame, Cook and Hoover [2], who showed that devision and related operations *including multiple product* are computable in NC^1. Based on this result Reif and Tate [48, 49] showed that these operations are also computable in TC^0. The exact depth required for these operations was considered in [51, 52] were it was shown that multiple sum is in TC^0_2, multiplication and division in TC^0_3 and multiple product in TC^0_4.

By the results above, we get that after preprocessing (i.e., computing the values g^{2^i}), it is possible to evaluate the function $f_{P,Q,g,\vec{a}}$ in TC^0. A more detailed analysis reveals that some further optimizations in the depth are possible: Using additional preprocessing, this function is in fact in TC^0_5. We defer this detailed analysis to the final version of this paper and only briefly comment on a few facts regarding $f_{P,Q,g,\vec{a}}$ that allow us to reduce the depth (compared with a naive application of [51, 52]). First, note that in both multiple products we can assume any preprocessing of the values in the multiplication (since these values are taken from the sequence $\langle a_0, a_1, \ldots a_n \rangle$ or from the set $\{g^{2^i}\}$). Second, we don't need the actual value of the first multiple product, $y = \prod_{x_i=1} a_i$: Computing values r_i (obtained by the CRT-representation) for which $y = \sum m_i \cdot r_i$ (where the values m_i are known in advance and can be preprocessed) is just as good. Finally, the value P is also known in advance. Therefore, the depth of the final modular reduction can be reduced by precomputing the values $2^i \bmod P$. In conclusion:

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

Remark 4.3 *The same analysis hold for efficiency and depth of the pseudo-random functions of Construction 5.1.*

5 A Related Construction Based on 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. We include this construction for two main reasons:

1. The GDH-Assumption is implied by the DDH-Assumption but they are not known to be equivalent. Therefore, Construction 5.1 may still be valid even if the DDH-Assumption does not hold. In addition, the GDH-Assumption modulo a composite is not stronger than the assumption that factoring is hard. This gives an attractive construction of pseudo-random functions that is at least as secure as factoring.

2. Construction 5.1 is based on a somewhat different methodology than Construction 4.2. It may be easier to apply this methodology in order to construct 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., [53, 55]). 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, $I \subsetneq [n]$. It is essential that even with this knowledge it would be 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, $I \subsetneq [n]$ of its choice.

To remain consistent with the DDH-Assumption, we state the GDH-Assumption (Assumption 5.1) in a subgroup of $\mathbb{Z}_P^*$ 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 [5, 53], 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 $\mathbb{Z}_P^*$ itself (i.e., when g is a generator of $\mathbb{Z}_P^*$).

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 $\vec{a} = \langle a_1, a_2, \ldots a_n\rangle$ be any sequence of n elements of $\mathbb{Z}_Q$. Define the function $h_{P,Q,g,\vec{a}}$ with domain $\{0,1\}^n$ such that for any n-bit input, $x = x_1 x_2 \cdots x_n$*

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

Define $h^r_{P,Q,g,\vec{a}}$ to be the restriction of $h_{P,Q,g,\vec{a}}$ 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 sufficiently large n's*

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

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 $\vec{a} = \langle a_1, a_2, \ldots a_n\rangle$ uniformly at random in $\mathbb{Z}_Q$.

As a corollary of Theorem 4.1 we have that if the DDH-Assumption holds, then so thus the GDH-Assumption. In fact, we get that the DDH-Assumption imply the *decisional* GDH-Assumption (this was also previously shown in [55]):

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 sufficiently large n's*

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

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 $\vec{a}=\langle a_1,a_2,\ldots a_n\rangle$ uniformly at random in $\mathbb{Z}_Q$ and the choice of c uniformly at random in $\mathbb{Z}_Q$.

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 [43]. 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, $M=\left(S(x^1_{i_1},x^2_{i_2},\ldots,x^k_{i_k})\right)_{i_1,i_2,\ldots,i_k=1}^m$, cannot be efficiently distinguished from uniform by an algorithm that can access M at points of its choice.*

The construction of [43] 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 $F = \{F_n\}_{n\in\mathbb{N}}$. 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 $\mathbb{Z}_P^*$, $\vec{a} = \langle a_{1,0}, a_{1,1}, a_{2,0}, a_{2,1}, \ldots a_{n,0}, a_{n,1}\rangle$ a sequence of $2n$ elements of $\mathbb{Z}_Q$ and r an n-bit string. For any n-bit input, $x = x_1 x_2 \cdots x_n$, 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 $F = \{F_n\}_{n\in\mathbb{N}}$ (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 [30] (to be more precise, the setting of this corollary is somewhat different than the one considered in [30] 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 sufficiently large n's*

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

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 $\vec{a} = \langle a_{1,0}, a_{1,1}, a_{2,0}, a_{2,1}, \ldots a_{n,0}, a_{n,1}\rangle$ uniformly at random in $\mathbb{Z}_Q$, 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 $F = \{F_n\}_{n\in\mathbb{N}}$ 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 $\{0,1\}^n \mapsto \{0,1\}$ 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

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

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 $h^r_{P,Q,g,\vec{a}}$ and on input $\langle P,Q,g,r,\tilde{\sigma}\rangle$ (where we expect $\tilde{\sigma}$ to either be uniformly chosen or to be $(h_{P,Q,g,\vec{a}}(1^n)) \odot r$), $\mathcal{A}$ 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 $\mathbb{Z}_Q$.
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
   $$\left(\left(h^r_{P,Q,g,\vec{a}}(z)\right)^{\prod_{z_k=0} b_k}\right) \odot r$$
4. Output whatever $\mathcal{M}$ outputs.

From the definition of $\mathcal{A}$ we have that

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

and

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

where the probabilities are as above. We also have that for any $0 < j < t(n)$

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

Therefore, by the standard hybrid argument we get from the assumption that for infinitely many n's

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

□

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 [5, 53], we obtain an attractive construction of pseudo-random functions which is at least as secure as factoring. In this subsection, we repeat the definition of the GDH-Assumption and the construction of pseudo-random functions with the group set to $\mathbb{Z}_N^*$, 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):

Definition 5.2 (FIG) *The factoring-instance-generator, FIG, is a probabilistic polynomial-time algorithm such 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 5.3 *Let N be any possible output of $FIG(1^n)$, let g be any quadratic-residue in $\mathbb{Z}_N^*$ and let $\vec{a} = \langle a_1, a_2, \ldots a_n \rangle$ be any sequence of n elements of $[N]$. Define the function $h_{N,g,\vec{a}}$ with domain $\{0,1\}^n$ such that for any n-bit input, $x = x_1 x_2 \cdots x_n$,*

$$h_{N,g,\vec{a}}(x) \stackrel{\text{def}}{=} g^{\prod_{x_i=1} a_i} \bmod N$$

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

Assumption 5.2 (Generalized Diffie-Hellman in $\mathbb{Z}_N^*$) *For every probabilistic polynomial-time oracle machine $\mathcal{A}$, every constant $\alpha > 0$ and all sufficiently large n's*

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

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 $\mathbb{Z}_N^*$ and the choice of each of the values in $\vec{a} = \langle a_1, a_2, \ldots a_n \rangle$ uniformly at random in $[N]$.

The Construction and its Security:

Construction 5.2 *We define the function ensemble $F = \{F_n\}_{n \in \mathbb{N}}$. 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 $\mathbb{Z}_N^*$, $\vec{a} = \langle a_{1,0}, a_{1,1}, a_{2,0}, a_{2,1}, \ldots a_{n,0}, a_{n,1} \rangle$ 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:*

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

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

Since breaking the GDH-Assumption in $\mathbb{Z}_N^*$ is at least as hard as factoring N we can deduce that:

Corollary 5.5 *(of Theorem 5.4 and [5, 53]) Let $F = \{F_n\}_{n\in\mathbb{N}}$ 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 $FIG(1^n)$.

Furthermore, the reduction is linear-preserving (see [38]): Assume that there exists a probabilistic algorithm with running-time $t(n)$ that distinguishes F from the uniform function-ensemble with advantage $\epsilon(n)$. Then there exists a probabilistic algorithm with running-time $t(n) \cdot poly(n)$ for factoring with success-probability $\epsilon(n)$.

6 Additional Properties of the Pseudo-Random Functions

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. An interesting example aroused by the work of Bellare and Goldwasser [3]. They suggested a way to design a digital-signature scheme which would be very attractive given efficient 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).

In this section, we suggest preliminary designs for several protocols. These designs are probably not efficient enough for practical implementation but still they serve as a demonstration to the potential of our construction. We hope this work will stimulate further research both in improving these designs and in suggesting designs for other protocols (as the non-interactive zero-knowledge proof mentioned above and a function-sharing scheme for pseudo-random functions). We defer the formal definitions and proofs of this section to the final version of the paper. In addition, we focus on the construction of Section 4 (almost the same designs work for the construction of Section 5).

A point worth noticing is that we describe our designs for the functions of Construction 4.1. That is, we ignore the pair-wise independent hashing introduced in Construction 4.2. However, as mentioned in Section 4, for many applications (as the undeniable signatures suggested below) the extra hashing in Construction 4.2 is unnecessary. In addition, recall Remark 4.2 that the pseudo-random functions of Construction 4.2 remain

pseudo-random even if the hash function is public. Therefore, the protocols we design in the following subsections for Construction 4.1 imply similar protocols for Construction 4.2. For example, when distributing a pseudo-random function to a set of parties, we can distribute the function obtained by ignoring the hash function and supply each party with the value of this hash function.

6.1 Zero-Knowledge Proof for the Value of the Function

We mentioned that a non-interactive zero-knowledge proof for claims of the form $y = f_s(m)$ is required by the Bellare-Goldwasser digital-signature scheme. Similarly a simple (interactive) zero-knowledge proof for claims of the form $y = f_s(m)$ and $y \neq f_s(m)$ implies a simple construction of undeniable signatures. Informally, undeniable signatures, which were introduced by Chaum and Antwerpen [18], are public-key schemes that allow a party to sign messages such that his participation is required in order to verify a signature. We can let the public key for an undeniable signature be a commitment to a key, s, of a pseudo-random function, f_s. A signature for a message m can simply be $f_s(m)$ (or $f_s(H(m))$ for a collision-intractable hash function, H). Now the confirmation protocol for a message m and a signature y is simply a zero-knowledge proof that $y = f_s(m)$ whereas the denial protocol is a zero-knowledge proof that $y \neq f_s(m)$.

In this section we describe such zero-knowledge proofs for the values of the functions of Construction 4.1. To be a bit more accurate, both the commitment for a function and the zero-knowledge proofs *do reveal some of the values of the function.* Therefore, these are zero-knowledge proofs for a *restriction* of the function to a subset of its inputs (those with their last two bits set to 1), when the value of the function on the rest of the inputs is publicly available (this can be formalized by allowing the zero-knowledge simulator access to all other values of its choice).

The protocol for $y = f_s(x)$ is essentially several parallel applications of a zero-knowledge proof for the result of the Diffie-Hellman protocol. These proofs are strongly related to Schnorr's identification protocol [50]. In order to make his proof into a zero-knowledge proof we use a strong-receiver commitment scheme in a rather standard way (using a *strong-sender* commitment scheme gives perfect zero-knowledge *arguments*). We denote the commit (resp. reveal) phases of this protocol by COMMIT (resp. REVEAL).

Definition 6.1 *(a commitment for f_s) Let $F = \{F_n\}_{n\in\mathbb{N}}$ be as in Construction 4.1 and let $f_{P,Q,g,\vec{a}}$ be some function in F_n, where $\vec{a} = \langle a_0, a_1, \ldots a_n\rangle$. A commitment for $f_{P,Q,g,\vec{a}}$ is*

$$\langle P, Q, g, g^{a_0}, g^{a_0 \cdot a_1}, g^{a_0 \cdot a_2}, \ldots, g^{a_0 \cdot a_n}\rangle$$

Protocol 6.1 *(ZK-Proof for $y = f_s(x)$)*
The prover, $\mathcal{P}$, knows the key $s = \langle P, Q, g, \vec{a}\rangle$ and the verifier $\mathcal{V}$ knows the commitment for f_s. The common input is a pair $\langle x, y\rangle$, where $x = x_1 x_2 \cdots x_n \in \{0,1\}^n$ satisfies that $x_{n-1} = x_n = 1$ and y is in the subgroup of $\mathbb{Z}_P^*$ generated by g. Denote by $\tilde{g}$ the value g^{a_0} and assume wlog that for some k, $x_1 = x_2 = \cdots = x_k = 0$ whereas $x_{k+1} = \cdots = x_n = 1$. The protocol for proving $y = f_s(x)$ is defined as follows:

1. *$\mathcal{V}$ chooses a value e uniformly at random in $\mathbb{Z}_Q^*$ and sends COMMIT(e) to $\mathcal{P}$.*

2. *For each* $i \in [(k+1)..n]$, $\mathcal{P}$ *chooses* r_i *uniformly at random in* $\mathbb{Z}_Q$, *computes* $c_i = \left(\prod_{j=1}^{i-1} a_j\right) \bmod Q$ *and sends* $\langle y_i = \tilde{g}^{c_i}, \tilde{g}^{r_i}, (y_i)^{r_i} \rangle$ *to* $\mathcal{V}$. *Denote by* y_{n+1} *the value* y.

3. $\mathcal{V}$ *sends* $REVEAL(e)$ *to* $\mathcal{P}$.

4. *For each* $i \in [(k+1)..n]$, $\mathcal{P}$ *sends* $d_i = e \cdot a_i + r_i \bmod Q$ *to* $\mathcal{V}$.

5. $\mathcal{V}$ *accepts if the following conditions hold: (1)* $y_{k+1} = \tilde{g}$. *(2)* $\forall i \in [(k+1)..n]$, $(y_i)^{r_i} \cdot (y_{i+1})^e = (y_i)^{d_i}$ *and* $(\tilde{g}^{a_i})^e \cdot \tilde{g}^{r_i} = \tilde{g}^{d_i}$.

In addition to the ideas used by the protocol for $y = f_s(x)$, the protocol for $y \neq f_s(x)$ uses the randomized-reduction of Section 3.3 in order to prove that a value is *not* the result of the Diffie-Hellman protocol.

Protocol 6.2 *(ZK-Proof for* $y \neq f_s(x)$*)*
Let the setting and notation be as in Protocol 6.1. The protocol for proving $y \neq f_s(x)$ *is defined as follows:*

1. $\mathcal{V}$ *chooses a value* e *uniformly at random in* $\mathbb{Z}_Q^*$ *and a uniform subset,* J, *of* $[n]$. $\mathcal{V}$ *sends* $COMMIT(e, J)$ *to* $\mathcal{P}$.

2. *For each* $i \in [(k+1)..n]$, $\mathcal{P}$ *chooses* r_i *uniformly at random in* $\mathbb{Z}_Q$, *computes* $c_i = \left(\prod_{j=1}^{i-1} a_j\right) \bmod Q$ *and sends* $\langle y_i = \tilde{g}^{c_i}, \tilde{g}^{r_i}, (y_i)^{r_i} \rangle$ *to* $\mathcal{V}$.

   Also, for every $i \in [n]$, $\mathcal{P}$ *chooses* u_i *and* v_i *uniformly at random in* $\mathbb{Z}_Q$ *and sends* $\langle \ell_i^1 = \tilde{g}^{u_i \cdot a_n + v_i}, \ell_i^2 = y^{u_i} \cdot \tilde{g}^{v_i} \rangle$ *to* $\mathcal{V}$.

3. $\mathcal{V}$ *sends* $REVEAL(e, J)$ *to* $\mathcal{P}$.

4. *For each* $i \in [(k+1)..(n-1)]$, $\mathcal{P}$ *sends* $d_i = e \cdot a_i + r_i \bmod Q$ *to* $\mathcal{V}$.

   Also, for every $i \in J$, $\mathcal{P}$ *sends* u_i *and* v_i *to* $\mathcal{V}$ *and for every* $i \in [n] \setminus J$, $\mathcal{P}$ *sends* $u_i \cdot a_n + v_i$ *to* $\mathcal{V}$.

5. $\mathcal{V}$ *accepts if the following conditions hold: (1)* $y_{k+1} = \tilde{g}$. *(2)* $\forall i \in [(k+1)..(n-1)]$, $(y_i)^{r_i} \cdot (y_{i+1})^e = (y_i)^{d_i}$ *and* $(\tilde{g}^{a_i})^e \cdot \tilde{g}^{r_i} = \tilde{g}^{d_i}$. *(3)* $\forall i \in J$, $\ell_i^1 = (\tilde{g}^{a_n})^{u_i} \cdot \tilde{g}^{v_i}$ *and* $\ell_i^2 = y^{u_i} \cdot \tilde{g}^{v_i}$ *(i.e.,* $\mathcal{P}$ *sent the correct values). (4)* $\forall i \in [n] \setminus J$, *we have that* $\ell_i^1 = \tilde{g}^{u_i \cdot a_n + v_i}$ *(i.e.,* $\mathcal{P}$ *sent the correct value) and* $(y_n)^{u_i \cdot a_n + v_i} \neq \ell_i^2$.

6.2 Function Sharing

For many applications (as the undeniable signature-scheme described above), it is desirable to have a simple function-sharing scheme for pseudo-random functions. De-Santis et. al. [20] defined function-sharing schemes for trapdoor-permutations. However, one can formulate and analogous definition of function-sharing for pseudo-random functions. Informally, given some (monotone) access structure for ℓ parties and a key s of a pseudo-random function f_s, we want a way to give party i a function Sh_i such that the following two conditions hold: (1) For any value x and any authorized subset of the parties $J \subset [\ell]$, it is easy to compute $f_s(x)$ given $\{Sh_j(x)\}_{j \in J}$. (2) Let $J \subset [\ell]$ be any unauthorized subset of the parties and

$\mathcal{A}$ any efficient algorithm that is given the shares $\{Sh_j\}_{j\in J}$ as input. Then even after $\mathcal{A}$ adaptively queries all the shares $\{Sh_j\}_{j\notin J}$ at points $\langle x^1 \ldots x^m\rangle$ of its choice, $\mathcal{A}$ cannot tell apart the restriction of f_s to all other inputs (apart of $\langle x^1 \ldots x^m\rangle$) from a random function.

Unfortunately, we do not know of a function-sharing scheme for the pseudo-random functions that are constructed in this paper. We do however know how to distribute these functions in a weaker sense. The main difference is that now an authorized subset of the parties is required to engage in a protocol in order to compute $f_s(x)$. Such a scheme has several disadvantages compared with a function-sharing scheme (see [20] for a discussion). In particular, the definition of security is more delicate (since it should consider executions of the protocol when an unauthorized subset is cheating). However, such a scheme might still be useful.

As an example, we briefly describe how to distribute a function $f_s = f_{P,Q,g,\vec{a}} \in F_n$, where $F = \{F_n\}_{n\in\mathbb{N}}$ is as in Construction 4.1, $\vec{a} = \langle a_0, a_1, \ldots a_n\rangle$ and the only authorized subset is $[\ell]$ itself: Party i gets the key of the function $Sh_i = f_{P,Q,g,\vec{a}^i} \in F_n$, where $\vec{a}^i = \langle a_0^i, a_1^i, \ldots a_n^i\rangle$ and the values $\{\vec{a}^i\}_{i\in[\ell]}$ are uniformly chosen, subject to the condition that $\forall 0 \leq j \leq n,\ a_j = \prod_{i\in[\ell]} a_j^i \bmod Q$. Now, for every input x there exists values $c, c_1, c_2, \ldots, c_\ell$, such that $Sh_i(x) = g^{c_i}$, $f_s(x) = g^c$ and $c = \prod_{i\in[\ell]} c_i \bmod Q$. Therefore, the computation of $f_s(x)$, can be done in n steps. At step 1 the first party publishes g^{c_1}. At step i party i can compute and publish $g^{\prod_{i=1}^{i} c_i}$ (and perhaps also prove this value). Additional access structures for which we can distribute the function $f_{P,Q,g,\vec{a}}$ are (1) "t out of ℓ": the authorized subsets are the ones with at least t elements (2) Access structures that can be described by a small monotone formula. We omit details from this preliminary version.

6.3 Blind Computation

We also suggest a new and attractive feature for a pseudo-random function — a protocol for "blind-computation" of its value: 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). One possible application of such a protocol is for "blind-authentication". It is also interesting to compare with a protocol for oblivious-transfer, during which $\mathcal{B}$ learns one of *two* possible values (whereas in blind-computation $\mathcal{B}$ learns one of exponential many possible values).

We came up with a preliminary design for blind-computation of the functions of Construction 4.1. This is an inefficient protocol since it requires $\Theta(n)$ rounds, still it shows that such a protocol is possible. The main idea of the protocol is the following: Let $u_j = (g^{a_0})^{\prod_{x_i=1, i\leq j} a_i}$ and $v_j = (g^{a_0})^{\prod_{x_i=o, i\leq j} a_i}$ The output of step j is $(u_j)^r$ and $(v_j)^{r'}$ for two values r and r', known to $\mathcal{B}$. Now $\mathcal{B}$ chooses t and t' uniformly at random, computes $(u_j)^{r\cdot t}$ and $(v_j)^{r'\cdot t'}$ and send this pair to $\mathcal{A}$ in a uniformly chosen order (in fact, $\mathcal{B}$ also has to prove that he knows such values t and t'). If $x_{j+1} = 1$ $\mathcal{B}$ asks for $((u_j)^{r\cdot t})^{a_{j+1}}$ and otherwise for $\left((v_j)^{r'\cdot t'}\right)^{a_{j+1}}$. We omit the formal description from this version.

7 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.

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.

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

Acknowledgments

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

References

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

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

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

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

[5] E. Biham, D. Boneh and O. Reingold, Generalized Diffie-Hellman modulo a composite is not weaker than factoring, *Theory of Cryptography Library*, Record 97-14 at:
`http://theory.lcs.mit.edu/~tcryptol/homepage.html`

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

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

[8] 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.

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

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

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

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

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

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

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

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

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

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

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

[20] 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.

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

[22] 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.

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

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

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

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

[27] 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).

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

[29] 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.

[30] 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.

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

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

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

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

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

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

[37] 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.

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

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

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

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

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

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

[44] M. Naor and O. Reingold, On the construction of pseudo-random permutations: Luby-Rackoff revisited, To appear in: *J. of Cryptology*. Preliminary version in: *Proc. 29th Ann. ACM Symp. on Theory of Computing*, 1997. pp. 189-199.

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

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

[47] A. Razborov and S. Rudich, Natural proofs, *J. of Computer and System Sciences*, vol. 55(1), 1997, pp. 24-35. Preliminary version: *Proc. 26th Ann. ACM Symp. on Theory of Computing*, 1994, pp. 204-213.

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

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

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

[51] 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.

[52] 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.

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

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

[55] 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.

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

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