A Primitive for Proving the Security of Every Bit and About Universal Hash Functions & Hard Core Predicates*

Eike Kiltz

Lehrstuhl Mathematik & Informatik, Fakultät für Mathematik,
Ruhr-Universität Bochum, 44780 Bochum, Germany.
February 4, 2002
kiltz@lmi.ruhr-uni-bochum.de

Abstract. In 1999, J. Håstad and M. Näslund [13] could prove that every bit is a hard core of the RSA function. From this work we extract an abstract theorem about the hidden number problem which can be used to prove that every bit is a hard core of *many specific cryptographic functions*. Applications are RSA, ElGamal, Rabin, a modified Diffie-Hellman function, Pailler's cryptosystem, the Diffie-Hellman function for elliptic curves and discrete exponentiation.
So far all in the literature known *general constructions* of hard core predicates for any one-way function (for example the famous Goldreich-Levin Bit [12]) are based on some set of universal hash functions (UHF). The natural open question was if there may be a nice connection between UHF and hard core predicates. We present an example providing a negative answer to that question. Furthermore, as an alternative to the Goldreich-Levin Bit, we give a new and efficient construction of a hard core predicate of any one-way function that is based on the hidden number problem.

Keywords: hard core predicates, bit-security, universal hash functions, ElGamal, RSA, one-way functions, hidden number problem

1 Introduction

The existence of one-way functions is one of the central aspects in modern cryptography. Loosely speaking one-way functions are functions that are easy to evaluate and hard to invert. But even if inverting f is hard, a one-way function may "leak" certain information about x in $f(x)$. For example given an instance $f(x)$ it may be hard (on the average) to efficiently compute the corresponding x, but what about the *least significant bit* of the binary representation of x? In fact, it is known that the discrete exponentiation function leaks this least significant bit but the RSA function does not.
The problem of showing that $f(x)$ hides at least one bit of information about x is called the *hard core predicate problem*. In 1989 Goldreich and Levin [12] proved that every one-way function can be modified to have such a hard core, the so called *Goldreich Levin Bit*. This modification has no substantial loss in either "security" or "efficiency". In 1996 Näslund [19] showed that every one-way function can be modified such that *every* single bit of a linear function on $\mathbb{Z}_p$ also is a hard core of that modified one-way function.
The expression "security of b" sometimes appears as a synonym for a hard core b in the literature. For instance, if a bit of the binary representation is a hard core of f we say that

* A 4-sheet version of this paper appeared in the proceedings of FCT'01 [15].

this bit is "secure".

There are numerous applications for the concept of hard core predicates in modern cryptography [11]. A hard core predicate for a permutation f gives rise to a:

- Pseudorandom generator.
- Secure bit-commitment scheme.
- Semantically secure cryptosystem.

In the *first part* of our work we consider the bitsecurity of concrete functions. As already noted the least significant bit is a hard core of a concrete function, the RSA function. In 1999, J. Håstad and M. Näslund [13] could shown that even every bit is a hard core of the RSA function. The proof of that result only makes use of the so called multiplicative structure of the RSA function. That is, given $\mathrm{RSA}(x) = x^e \bmod N$ and an integer c, one can compute $\mathrm{RSA}(cx \bmod N) = c^e \cdot x^e \bmod N$. Also in [13] it was shown that all (but the first least significant[1]) bits are a hard core of the discrete exponentiation function on $\mathbb{Z}_p^*$: $\mathrm{EXP}(x) = g^x \bmod p$. The proof again relies on the multiplicative structure, i.e. $\mathrm{EXP}(cx) = \mathrm{EXP}(x)^c \bmod p$.

In section 3 we present the Hidden Number Problem, a useful primitive to prove bitsecurity. One can conclude the bitsecurity of many cryptographic functions as a simple corollary. These cryptographic functions include RSA, ElGamal, Rabin, discrete exponentiation, a modified Diffie-Hellman function, the Diffie-Hellman function on elliptic curves and Paillier's function. Although there already exist similar proofs for most of these cryptographic functions one has to *individually adapt* them to prove the result for a concrete function. Because of the complexity of the proof this is a very unthankful work. The hidden number problem exploits the common structure of all those cryptographic functions. Once proven it lets us *easily* conclude the security of every bit of many cryptographic functions. This is illustrated in figure 1. The proofs of the bitsecurity of the ElGamal function, a modified Diffie-Hellman function (MDH), the elliptic curve Diffie-Hellman function (ECDH) as introduced in [4] and Paillier's function are new results (marked by darker leaves).

The Hidden Number problem (at least in a similar fashion) was first introduced by Boneh and Venkatesan [5] and successfully exploited to show that the collection of the $\sqrt{\log n}$ "unbiased" most significant bits are a hard core function of the ElGamal and the Diffie-Hellman function. An error in the proof in [5] was spotted and corrected by [24]. See also [20] how to apply lattice-based techniques to solve the Hidden Number problem.

In addition we exploit the hidden number problem to present a new general and efficient construction of hard-core predicates for any (modified) one-way function as an alternative to the Goldreich-Levin Bit [12]. When extracting $O(\log n)$ simultaneous hard core bits our method is slightly faster than the Goldreich-Levin method.

In the *second part* of our work we consider hard core predicates for any one-way function. The two most commonly used general constructions of hard core predicates [12, 19] mentioned

[1] The first least significant bits of EXP are known to be "easy". We will refer to that later.

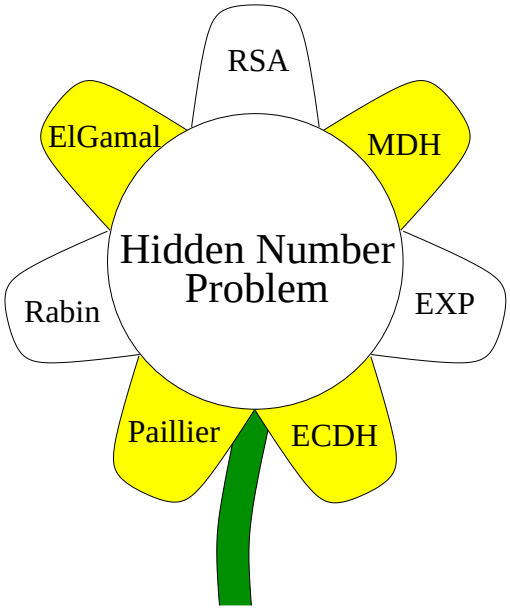

Fig. 1. The hidden number problem and its relation to cryptographic functions.

above, as well as some others, heavily rely on sets of universal hash functions. On the other hand, universal hash functions and hard core predicates have the same upper bounds of complexity [10, 9]. These two facts suggest a connection between universal hash functions and hard core functions. This connection was first conjectured by Näslund [18].
In section 4.2 we answer this question to the negative with a simple counterexample. This is done by modifying a set of universal hash functions in a certain way such that it cannot be a hard core of a (specific) one-way function.
In section 4.3 we use the hidden number problem to give an example of a set of hash functions that is only 2/3-universal (i.e. almost universal) but still can be used to give a hard core predicate. These two results seem to show that the concepts of universal hash functions and of hard core predicates are less correlated than expected. Based on our observations, in section 4.4 we present a structural result about the construction of hard core predicates for any (modified) one-way function, i.e. we characterize how to modify a given general hard core to get a new one.
An excellent overview of hard core functions is given in the article of González Vasco and Mats Näslund [23].

2 Notations and Preliminaries

All computations are done in the model of probabilistic polynomial time Turing machine (*pptm*). For a string $x \in \{0,1\}^*$, $|x|$ denotes the length of the binary representation of x. A function $f : \{0,1\}^* \to \{0,1\}^*$ is called *length-regular* if for every $x, y \in \{0,1\}^*$, if $|x| = |y|$, then $|f(x)| = |f(y)|$. For an $x = \sum_{i=0}^{n} x_i 2^i$, $\mathrm{bit}_i(x) := x_i$ denotes the i-th bit of the binary representation of x, i.e. $\mathrm{lsb}(x) = bit_0(x)$, the least significant bit.
A function α is called *negligible* in n if for every polynomial P and all sufficiently large n's

it holds that $\alpha(n) < \frac{1}{P(n)}$. A function α is called *non-negligible* in n if for a polynomial P and all sufficiently large n's it holds that $\alpha(n) \geq \frac{1}{P(n)}$. Note that there are functions that a neither negligible nor non-negligible.
A polynomial time computable function $f : \{0,1\}^* \to \{0,1\}^*$ is called *one-way function* if for every pptm $\mathcal{A}$ the function $\alpha(n) := \Pr[f(\mathcal{A}(f(x)) = f(x)]$ is negligible, probability taken over x, uniformly distributed on $\{0,1\}^n$, and internal coin tosses of $\mathcal{A}$.

Definition 1 (Hard core predicate). *Let $b : \{0,1\}^* \to \{0,1\}$ be a polynomial-time computable function. The predicate b is called a* hard core predicate *of a function g if for every pptm $\mathcal{D}$ (predictor),*

$$\Pr[\mathcal{D}(g(x)) = b(x)] - \frac{1}{2}$$

is negligible as a function in n, where x is chosen at random and uniformly distributed from $\{0,1\}^n$.

If the function b is a single bit of the binary representation, we say that the bit is *individually secure* for the function g.

Note that no function b can be a hard core of *every* one-way function. To see that fix an arbitrary one-way function f and a hard core b (of f). Now define $g(x) := (f(x), b(x))$. By definition, g is a one-way function. Now we have constructed a counterexample for this b, as $b(x)$ can easily be recovered given the tuple $(f(x), b(x)) = g(x)$. Therefore b is no hard core of g. Definition 1 can easily be generalized to the case of more than one bit.

Definition 2 (Hard core function). *A polynomial time computable function $h : \{0,1\}^n \to \{0,1\}^{l(n)}$ is called a* hard core function *of a function g if for pptm $\mathcal{D}$ (distinguisher),*

$$\left|\Pr[\mathcal{D}(g(x), h(x)) = 1] - \Pr[\mathcal{D}(g(x), r_{l(n)}) = 1]\right|$$

is negligible as a function in n, where x is chosen at random and uniformly distributed from $\{0,1\}^n$ and $r_{l(n)}$ is chosen at random and uniformly distributed from $\{0,1\}^{l(n)}$.

If the function h are some bits of the binary representation, we say that the bits are *simultaneous secure* for the function g.

2.1 Known Results

The following well known Theorem from Goldreich and Levin [12] states that every one-way function can be easily modified into a one-way function that has a hard core predicate. This modification has no substantial loss in either "security" or "efficiency".

Theorem 1 (Goldreich and Levin [12]). *Let f be an arbitrary length regular one-way function and let g be defined by $g(x, r) := (f(x), r)$ where $|x| = |r|$. Let $b(x, r)$ denote the inner-product* $\bmod 2$ *of the binary vectors x and r. Then the predicate b is a hard core of the function g.*

Note that if f is a one-way function then g is one.

Theorem 2 (Näslund [19]). *Let f be an arbitrary length-regular one-way function and let g be defined by $g(x,u,v) := (f(x),u,v)$, where $n = |x|$, p is a n-bit prime and $(u,v) \in \mathbb{Z}_p \times \mathbb{Z}_p^*$. Let $b_j(x,u,v) := \mathrm{bit}_j(ux+v \bmod p)$. Then for every $j = j(n) \in \{0,\ldots,n-1\}$ the predicate b_j is a hard core of the function g.*

For the previously given Theorem one has to use a slightly different definition of hard core predicates to fit the so called "bias of the upper bits" of the binary representation of a bit-string. We will, as for the rest of this paper, not concern about that. Note that, in particular, $b_0(x,r) = \mathrm{lsb}(ux+v \bmod p)$ is a hard core predicate of the one-way function g.

3 The Hidden Number Problem: a Useful Primitive to Prove Bitsecurity

3.1 Main Results and Motivation

In [13] it was shown that every single bit is a hard core of the RSA function. In particular, the same proof techniques were applied to show that all (but the first least significant) bits of the discrete exponentiation function are hard. It turned out as already noted in [13] that a special structure was needed to apply the techniques, a multiplicative structure. For instance, by the equation $\mathrm{RSA}(cx) = \mathrm{RSA}(c) \cdot \mathrm{RSA}(x)$ one can compute $\mathrm{RSA}(cx)$, given $\mathrm{RSA}(x)$ and c. Although the proofs are similar it seems that it is not possible to conclude, for instance, a result about the security of the discrete exponentiation bits from a result about the security of the RSA bits.

In this section we present a useful primitive, the hidden number problem, and study under which circumstances it is solvable. As a corollary we conclude the bitsecurity of many known cryptographic functions such as RSA and ElGamal. Figure 1 illustrates this connection.

Under some reasonable cryptographic assumptions[2], every bit is a hard core of the following functions:

1. The RSA encryption function
2. The ElGamal encryption function
3. A Modified Diffie-Hellman function (MDH)
4. The Diffie-Hellman function on Elliptic Curves (ECDH)
5. The Discrete Exponentiation function EXP[3]
6. The Rabin encryption function
7. Paillier's encryption function

Note that 1, 5 and 6 are known results whereas 2, 3, 4 and 7 are, at least for the results about the security of all bits, new contributions of this paper.
As a second corollary we give a new and very simple construction of hard core predicates for

[2] We will state the concrete assumptions later.
[3] For all but the first least significant bits.

every one-way function of the form $g(x,r) = (f(x),r)$. It only consists of a single multiplication modulo a prime p followed by a bit computation and gently improves the result from Theorem 2.

Theorem 3. *Let f be an arbitrary length-regular one-way function and let g be defined by $g(x,r) := (f(x),r)$, where $n = |x|$, p is a n-bit prime and $r \in \mathbb{Z}_p^*$. Let $b_j(x,r) := \mathrm{bit}_j(rx \bmod p)$. Then for every $j = j(n) \in \{0,\ldots,n-1\}$ the predicate b_j is a hard core of the function g.*

The results from Theorem 3 can easily be extended to $\mathcal{O}(\log n)$ simultaneous bits using standard techniques.

Theorem 4. *Let f be an arbitrary length-regular one-way function and let g be defined by $g(x,r) := (f(x),r)$, where $n = |x|$, p is a n-bit prime and $r \in \mathbb{Z}_p^*$. Let $h(x,r)$ be any combination of $\mathcal{O}(\log n)$ bits of $(rx \bmod p)$. Then the function h is hard core function of the function g.*

Remark 1. Theorems 3 and 4 also hold when the modulus p is a power of 2, i.e. if $p = 2^n$.

3.2 Discussion

Theorem 4 suggests a new method to get $O(\log n)$ simultaneous secure bits of any (modified) one-way function. For the special case considered in Remark 1 this method is more efficient than the original method given by Goldreich and Levin [12]. We will quickly discuss that.

To get $m := O(\log n)$ simultaneous secure bits for any (modified) one-way function, for the Goldreich-Levin method one multiplication of a binary vector (from $\{0,1\}^n$) with a $m \times n$ Toeplitz Matrix with entries from $\{0,1\}$ is needed. This can be done with $O(n\log(n))$ bit operations. Remark 1 says that using our method one can get $m = O(\log n)$ simultaneous hard core bits for any (modified) one-way function by the function $h(x,r) = xr \bmod p \bmod m$. This is the special case for the m least significant bits. Now take as the modulus a power of 2, i.e. $p = 2^n$. Then the function h simplifies to $h(x,r) = xr \bmod 2^m \bmod 2^n = xr \bmod 2^m = (x \bmod 2^m) \cdot (r \bmod 2^m) \bmod 2^m$. Hence, only one multiplication in $\mathbb{Z}_{2^m}^*$ is needed. This can be implemented in $O(m^2) = O((\log n)^2)$ bit operations. Hence, our new method nearly gains a factor of $O(n)$ in efficiency.

This leads to a efficient construction of a pseudorandom generator for *any* one-way permutation f which is due to Blum and Micali [2]. Let $h(x,r)$ be the $m := \mathcal{O}(\log n)$ least significant bits of $(rx \bmod 2^n)$, i.e. $h(x,r) = xr \bmod 2^m$.
Pick $x_0, r_0 \in \{0,1\}^n$ (the seed, or key) at random and let $(x_{i+1}, r_{i+1}) = (f(x_i), r_i)$. Now define the output of the pseudorandom generator as

$$G(x_0,r_0) = h(x_0,r_0)h(x_1,r_1),\ldots.$$

3.3 The Hidden Number Problem

We first informally introduce the Hidden Number Game as illustrated in Figure 2. It is

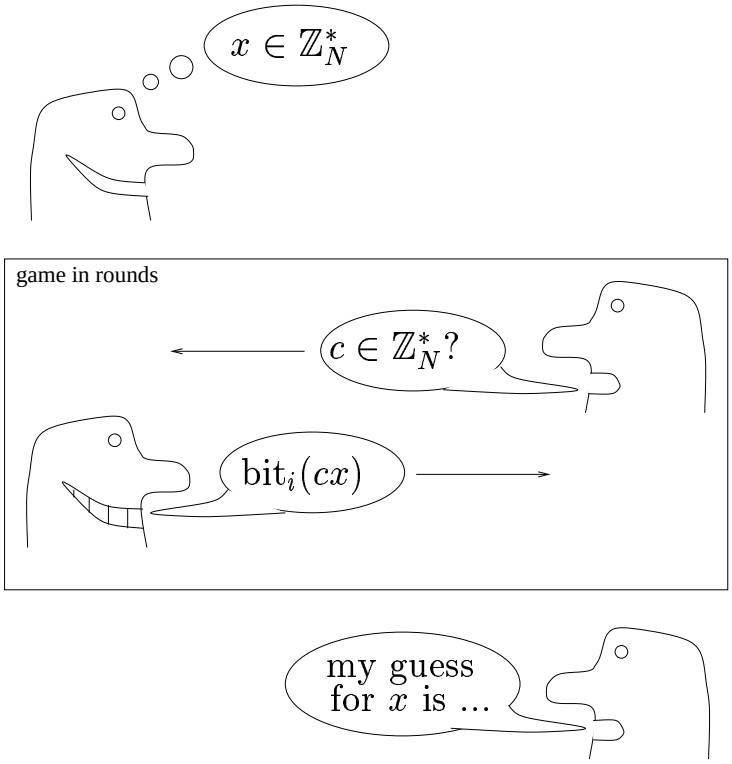

Fig. 2. The Hidden Number Game.

a game between Alice (left) and Bob (right). In the beginning both agree on a common modulus N and a fixed bit i. Alice takes a secret value $x \in \mathbb{Z}_N^*$. Now Bob is allowed to ask questions to Alice about $\mathrm{bit}_i(cx \bmod N)$ for any $c \in \mathbb{Z}_N^*$. After polynomial many rounds and after some internal computation Bob makes a guess for x. We say that he has won the game if he guesses the correct x with non-negligible probability of success.
The crucial question is:

Is there are winning strategy for Bob even if Alice is allowed to lie for nearly half of her answers?

The answer is yes.

We come to a more formal description.

Definition 3. *Let $I \subset \mathbb{N}$ be an infinite set of integers and P a polynomial. For all $N \in I$ and $x \in \mathbb{Z}_N^*$ let $(\mathcal{O}_{N,x})$ be a family of hidden number oracles for the i-th bit with*

$$\Pr[\mathcal{O}_{N,x}(c) = \mathrm{bit}_i(cx \bmod N)] - \frac{1}{2}$$

is a non-negligible function in the bitlength n of N, where the probability is taken over all $c \in \mathbb{Z}_N^*$, uniformly distributed and over all coin tosses of the oracle.

The Hidden Number Problem (HNP) associated with I and $(\mathcal{O}_{N,x})$ is to compute x. More formal we say that the HNP is efficiently solvable *for bit i if there is an oracle pptm $\mathcal{D}^{(\mathcal{O}_{N,x})}$*

allowed to make queries to $\mathcal{O}_{N,x}$ such that that for every $N \in I$ and $x \in \mathbb{Z}_N^*$, $\mathcal{D}^{\mathcal{O}_{N,x}}$ outputs x with non-negligible probability of success (taken over the internal coin tosses of $\mathcal{D}^{\mathcal{O}_{N,x}}$).

We state our main theorem about the Hidden Number Problem.

Theorem 5. *1. If I is the set of all odd primes, then HNP is efficiently solvable for all bits.*
2. If I is the set of all odd *integers, then HNP is efficiently solvable for all bits or one gets a non-trivial factor of the integer.*
3. If I is the set of all integers N of the form $N = 2^l \cdot q$, q odd*, then HNP is efficiently solvable for the bits $l < i < n$ or one gets a non-trivial factor of the integer q.*

As already pointed out, the proof of Theorem 5 is basically the same as in the Håstad and Näslund paper [13] about the security of every RSA bit. It is sketched in subsection 3.10.

Now, as a warm-up, Theorem 3 can easily be proven. To see this let f be a one-way function. Assume we have a hidden number oracle that, given $(f(x), r)$, predicts $\mathrm{bit}_i(rx \bmod p)$ with probability significantly better than random guessing. Now apply Theorem 5 to that oracle in order to compute x with non-negligible probability of success. This shows that either f is not a one-way function or the oracle cannot exist.

3.4 RSA and Rabin Bits

Let $I_{\mathrm{RSA}} = \{(N, e) : N = p \cdot q, |p| = |q|, p, q \text{ prime }, \gcd(e, \phi(N) = 1\}$ be the set of all possible RSA instances, where $\phi(x)$ denotes the Euler Totient Function and $I_{\mathrm{RSA}}(n) := I_{\mathrm{RSA}} \cap (\{0,1\}^n \times \{0,1\}^n)$ the set of all possible n-bit RSA instances.
Given $(N, e) \in I_{\mathrm{RSA}}$ and $x \in \mathbb{Z}_N^*$, the RSA function is defined as $\mathrm{RSA}_{N,e}(x) := x^e \bmod N$. Knowing the factorization of N, one can compute a d such that $ed \bmod \phi(N) = 1$. This is done using Euclids algorithm, for instance. Then $\mathrm{RSA}_{N,e}(x)^d = x^{de} = x \bmod N$. In order to apply the definition of a hard core predicate one has to consider RSA as a collection of functions. We leave the details, see [11] for more information.
The RSA function is believed to be one-way:
Strong RSA assumption: For every pptm $\mathcal{A}$ it holds that $\Pr[\mathcal{A}(N, e, \mathrm{RSA}_{N,e}(x)) = x]$ is a negligible function in n, where the probability is taken over all $(N, e) \in I_{\mathrm{RSA}}(n)$, $x \in \mathbb{Z}_N^*$ and the internal coin tosses of $\mathcal{A}$.

Theorem 6. *Under the (strong) RSA assumption, every bit is a hard core of the RSA function.*

The Theorem is a direct corollary of the following proposition.

Proposition 1. *If there exists an oracle $\mathcal{O}_{\mathrm{RSA}}$ satisfying*

$$\Pr[\mathcal{O}_{\mathrm{RSA}}(N, e, \mathrm{RSA}_{N,e}(x)) = \mathrm{bit}_i(x)] - \frac{1}{2} \text{ is non-negligible,}$$

probability taken over random choices of $(N, e) \in I_{\mathrm{RSA}}(n)$ and $x \in \mathbb{Z}_N^*$. Then there exists an oracle pptm $\mathcal{D}^{\mathcal{O}_{\mathrm{RSA}}}$ (inverting algorithm) satisfying

$$\Pr[\mathcal{D}^{\mathcal{O}_{\mathrm{RSA}}}(N, e, \mathrm{RSA}_{N,e}(x)) = x] \text{ is non-negligible,}$$

probability taken over all random choices of $(N, e) \in I_{\mathrm{RSA}}(n)$, $x \in \mathbb{Z}_N^*$ and the internal coin tosses of $\mathcal{D}^{\mathcal{O}_{\mathrm{RSA}}}$.

Proof. Let $N, e, \mathrm{RSA}_{N,e}(x)$ be given. Using the RSA oracle $\mathcal{O}_{\mathrm{RSA}}$ we will simulate a Hidden Number oracle $\mathcal{O}_{N,x}$ with x in the role of the Hidden Number. This allows us to apply Theorem 5 and therefore compute x with non-negligible probability. Hence, we have broken the strong RSA assumption. There are some technical details we have to take care of.

Let $T_{\mathrm{RSA}} \subseteq I_{\mathrm{RSA}}$ be the non-negligible fraction of pairs (N, e) such that for every fixed $(N, e) \in T_{\mathrm{RSA}}$ the oracle $\mathcal{O}_{\mathrm{RSA}}$ has a non-negligible advantage over random guessing the i-th bit of $x \in \mathbb{Z}_N^*$. The existence of such a fraction follows by a simple averaging argument and is a standard technique. Let us assume the pair (N, e) is such a "good instance".
Now simulate a Hidden Number oracle for the hidden number x exploiting the multiplicative structure of the RSA function:

$$\begin{aligned}\mathcal{O}_{N,x}(c) &:= \mathcal{O}_{\mathrm{RSA}}(N, e, c^e \cdot \mathrm{RSA}_{N,e}(x)) \\ &= \mathcal{O}_{\mathrm{RSA}}(N, e, \mathrm{RSA}_{N,e}(c \cdot x)) \\ &= \mathrm{bit}_i(cx)\end{aligned}$$

where the last equation holds if the RSA oracle did not lie. For a random $c \in \mathbb{Z}_N^*$, $\mathrm{RSA}_{N,e}(cx)$ is a random element, too. Hence, for all $(N, e) \in T_{\mathrm{RSA}}$ the oracles answer is correct with non-negligible advantage (probability over all c) over random guessing. According to Theorem 5 then there exists an oracle pptm $\mathcal{D}^{(\mathcal{O}_{N,x})}$ that computes the Hidden Number x with non-negligible probability, or we get a non-trivial factor of the modulus N which nevertheless enables us to reveal x. Thus, computing the cleartext x from the cyphertext $\mathrm{RSA}(x)$ is the same as computing the Hidden Number.
Now when choosing a random (N, e) from I_{RSA}, $\mathcal{D}^{(\mathcal{O}_{N,x})}$ still outputs x with non-negligible probability for all $x \in \mathbb{Z}_N^*$. We can now define $\mathcal{D}^{\mathcal{O}_{\mathrm{RSA}}}(N, e, \mathrm{RSA}_{N,e}(x)) := \mathcal{D}^{(\mathcal{O}_{N,x})}$ as output of the inverting algorithm which inverts RSA with non-negligible probability. □

The Rabin encryption function is RSA with exponent 2, i.e. $\mathrm{RABIN}(x) = x^2 \bmod N$. For that reason all RSA results above carry over to Rabin with some minor adoptions. Due to space limitations we will not go into detail.

3.5 A Modified Diffie-Hellman Function and its Bits

Define $I_{\mathrm{DH}} := \{(p, g) : p \text{ is a prime and } g \text{ is a primitive element of } \mathbb{Z}_p^*\}$ and $I_{\mathrm{DH}}(n) := I_{\mathrm{DH}} \cap (\{0,1\}^n \times \{0,1\}^n)$. The Diffie-Hellman function $\mathrm{DH}_{p,g}$ with respect to $(p, g) \in I_{\mathrm{DH}}$ is defined as $\mathrm{DH}_{p,g}(g^a, g^b) = g^{ab} \bmod p$. To the best of our knowledge, it is an open problem if computing a *single bit* of the Diffie-Hellman function is as hard as computing all of $\mathrm{DH}_{p,g}$.
The only known result concerning the bitsecurity of the Diffie-Hellman function is from Boneh and Venkatesan [5]. It shows that the *collection* of the $\sqrt{\log n}$ unbiased most significant bits of $\mathrm{DH}_{p,g}$ are as hard to compute as all of $\mathrm{DH}_{p,g}$. Furthermore it seems that the oracle machines used in [5] can only cope with almost faulty-free oracles.
In the same paper [5], Boneh and Venkatesan also present a modification of the Diffie-Hellman function with provable security of the unbiased most significant bit. We will present a modification with provable security of *every* bit. Its security will rely on the following assumption:

CDH: The Computational Diffie-Hellman (CDH) assumption for $\mathbb{Z}_p^*$ is the following:
For every pptm $\mathcal{A}$ it holds that $\Pr[\mathcal{A}(p, g, g^a, g^b) = g^{ab}]$ is a negligible function in n, where the probability refers to randomly chosen $(p, g) \in I_{\mathrm{DH}}(n)$, g^a, g^b from $\mathbb{Z}_p^*$ and the internal coin tosses of $\mathcal{A}$.
The CDH assumption for the group $\mathbb{Z}_p^*$ is believed to be true. See [3] for instance.
For $(p, g) \in I$ define

$$\mathrm{MDH}_{p,g}(g^a, g^b, u) := ug^{ab}.$$

Theorem 7. *Computing any bit of the Modified Diffie-Hellman function* MDH *is as hard as computing all of* MDH.

The Theorem is a direct corollary of the following proposition.

Proposition 2. *If there exists an oracle* $\mathcal{O}_{\mathrm{MDH}}$ *such that*

$$\Pr[\mathcal{O}_{\mathrm{MDH}}(p, g, g^a, g^b, u) = \mathrm{bit}_i(\mathrm{MDH}_{p,g}(g^a, g^b, u))] - \frac{1}{2} \text{ is non-negligible,}$$

probability taken over random choices of $(p, g) \in I_{\mathrm{DH}}(n)$ *and* $a, b, u \in \mathbb{Z}_p^*$. *Then there exists an oracle pptm* $\mathcal{D}^{\mathcal{O}_{\mathrm{MDH}}}$ *such that*

$$\Pr[\mathcal{D}^{\mathcal{O}_{\mathrm{MDH}}}(p, g, g^a, g^b) = g^{ab}] \text{ is non-negligible,}$$

probability taken over all random choices of $(p, g) \in I_{\mathrm{DH}}(n)$, $a, b \in \mathbb{Z}_p^*$ *and the internal coin tosses of* $\mathcal{D}^{\mathcal{O}_{\mathrm{MDH}}}$.

Proof. Let p, g, g^{ab} be an instance of the Diffie-Hellman function. We again only concentrate on the non-negligible fraction of (p, g, a, b) for which the oracle $\mathcal{O}_{\mathrm{MDH}}$ has an non-negligible advantage over random guessing $bit_i(ug^{ab})$. Now to simulate the hidden number oracle for the hidden number $x := g^{ab}$ set

$$\mathcal{O}_{N,g^{ab}}(c) := \mathcal{O}_{\mathrm{MDH}}(p, g, g^a, g^b, c),$$

which is $bit_i(cg^{ab})$ in case the oracle did not lie. According to Theorem 5, we get the hidden number $x = g^{ab}$ with non-negligible probability of success. □

3.6 The Diffie-Hellman function on Elliptic Curves

At Crypto 2001 Boneh and Shparlinski proved the individual security of every of the $O(\log n)$ least significant bits for the Diffie-Hellman function on elliptic curves [4]. In fact, what they showed is the bitsecurity of a modified Diffie-Hellman similar to that mentioned in the previous subsection that exploits some special structure of elliptic curves. In their paper computing this Diffie-Hellman function is reduced to the so called Squaring Hidden Number problem. We will sketch that our results also apply to this type of Hidden Number problem. Hence, computing any bit of the Diffie-Hellman function on elliptic curves is as hard as computing the whole. We will not go into mathematical details of elliptic curves and describe the problem at a very informal level.

Let p be a prime and let $\mathbb{F}_p$ be the finite field of size p. Let $\mathbb{E}$ be an elliptic curve over $\mathbb{F}_p$, given by the affine Weierstrass equation of the form

$$y^2 = x^3 + Ax + B, \quad 4A^2 + 27B^2 \neq 0.$$

It is well known that the set $\mathbb{E}(\mathbb{F}_p)$ of $\mathbb{F}_p$-rational points of $\mathbb{E}$ form an Abelian group under an appropriate composition rule and with the point of infinity $\mathcal{O}$ as the neutral element. Let $I_{\overline{\mathrm{DH}}}(n)$ be the family of sets of all possible pairs $(\mathbb{E}, G)$, where G is an element of prime order from $\mathbb{E}$ and whose bitlength is of order n.
For a point $P = (x, y) \in \mathbb{E}$ it is not clear how $\mathrm{bit}_i(P)$ can be defined. In fact, (at least) two choices are possible. We define $\mathrm{bit}_i(P)$ as $\mathrm{bit}_i(x)$, the i-th bit of the x-coordinate of the point P. Setting $\mathrm{bit}_i(P) := \mathrm{bit}_i(y)$ leads to similar results.
Now the Diffie-Hellman function for a point $G \in \mathbb{E}$ (of prime order q) is defined as

$$\mathrm{DH}_{\mathbb{E},G}(aG, bG) = abG.$$

For a $P = (x, y) \in \mathbb{E}$ and for a $\lambda \in \mathbb{F}_p^*$, the mapping ϕ is defined as $\phi_\lambda(P) = (x\lambda^2, y\lambda^3) \in \phi_\lambda(\mathbb{E})$. For a point $P \in \mathbb{E}$ define $P_\lambda := \phi_\lambda(P)$. From [4] we have that for the mapping ϕ_λ and for a point $(x, y) \in \mathbb{E}$ it holds that $xG_\lambda = (xG)_\lambda$ and $yG_\lambda = (yG)_\lambda$. Therefore the mapping $\phi_\lambda : \mathbb{E} \to \phi_\lambda(\mathbb{E})$ homomorphism. In fact it is easy to show that ϕ_λ is an isomorphism. Hence, computing the Diffie-Hellman function for all curves $\{\phi_\lambda(\mathbb{E})\}_{\lambda \in \mathbb{F}_p^*}$ is as hard as computing the Diffie-Hellman function in $\mathbb{E}$.
We define the modified Diffie-Hellman function $\overline{\mathrm{DH}}$ as

$$\overline{\mathrm{DH}}_{\mathbb{E},g}(P, Q, \lambda) := \mathrm{DH}_{\phi_\lambda(\mathbb{E}),G_\lambda}(P_\lambda, Q_\lambda).$$

Note that this function basically uses λ as an index indicating in which group to evaluate the Diffie-Hellman function. Is has a similar function as the u in the definition of the modified Diffie-Hellman function of the last subsection.

Theorem 8. *Computing any bit of the modified Diffie-Hellman function $\overline{\mathrm{DH}}$ is as hard as computing all of $\overline{\mathrm{DH}}$.*

The theorem follows from the following proposition.

Proposition 3. *If there exists an oracle $\mathcal{O}_{\overline{\mathrm{DH}}}$ such that*

$$\Pr[\mathcal{O}_{\overline{\mathrm{DH}}}(\mathbb{E}, G, \overline{\mathrm{DH}}_{\mathbb{E},G}(aG, bG, \lambda)) = \mathrm{bit}_i(abG)] - \frac{1}{2} \text{ is non-negligible,}$$

probability taken over random choices of $(\mathbb{E}, G) \in I_{\overline{\mathrm{DH}}}(n)$ and $\lambda, a, b \in \mathbb{F}_p^*$. Then there exists an oracle pptm $\mathcal{D}^{\mathcal{O}_{\overline{\mathrm{DH}}}}$ such that

$$\Pr[\mathcal{D}^{\mathcal{O}_{\overline{\mathrm{DH}}}}(\mathbb{E}, G, \overline{\mathrm{DH}}_{\mathbb{E},G}(aG, bG, \lambda)) = abG] \text{ is non-negligible,}$$

probability taken over all random choices of $(\mathbb{E}, G) \in I_{\overline{\mathrm{DH}}}(n)$, $\lambda, a, b \in \mathbb{F}_p^*$ and the internal coin tosses of $\mathcal{D}^{\mathcal{O}_{\overline{\mathrm{DH}}}}$.

We give a basic idea of the proof. It essentially uses the following multiplicative structure of the $\overline{\mathrm{DH}}$ function as shown in [4]:

$$\overline{\mathrm{DH}}_{\mathbb{E},G}(aG, bG, \lambda) = \mathrm{DH}_{\phi_\lambda(\mathbb{E}),\phi_\lambda(G)}(\phi_\lambda(aG), \phi_\lambda(bG)) = \phi_\lambda(abG) = (\lambda^2 x, \lambda^3 y),$$

where $abG = (x, y)$. Setting x as the hidden number, we have a similar structure as needed by Theorem 5 to compute x with overwhelming probability of success. The main difference is that we have an oracle that outputs $\mathrm{bit}_i(c^2 x)$ instead of $\mathrm{bit}_i(cx)$ on the input c. In [4] this is refered as the Quadratic Hidden Number problem. Clearly this is just a problem of computing roots what can be done efficiently. Hence, for the Quadratic Hidden Number problem Theorem 5 also holds. Note that once we have computed the coordinate x of $abG = (x, y)$, the value y can also be computed by the Weierstrass equation. Hence, we have computed the point $abG = (x, y)$.

3.7 ElGamal bits

Define $I_{\mathrm{El}} := \{(p, g) : p \text{ is a prime and } g \text{ is a primitive element of } \mathbb{Z}_p^*\}$ and $I_{\mathrm{El}}(n) := I_{\mathrm{El}} \cap (\{0,1\}^n \times \{0,1\}^n)$. Let $(p, g) \in I_{\mathrm{El}}$. The ElGamal public key cryptosystem encrypts a message $x \in \mathbb{Z}_p$ given a public key g^a by computing the tuple (g^b, xg^{ab}). Here b is chosen uniformly and at random from the set $\mathbb{Z}_p^*$. Decryption using the private key a is done by computing $(g^b)^a = g^{ab}$ and then dividing to obtain the plaintext x.
Thus in order to break this cryptosystem one has to "invert" the ElGamal encryption

$$\mathrm{El}_{p,g,a,b}(x) := (g^a, g^b, xg^{ab}),$$

in a sense that given given (g^a, g^b, xg^{ab}) one has to compute x. Clearly, the security of the ELGamal cryptosystem relies on the CDH assumption for $\mathbb{Z}_p^*$.

Theorem 9. *Under the CDH assumption for $\mathbb{Z}_p^*$, every bit is a hard core of the ElGamal function $\mathrm{El}_{p,g,a,b}$.*

The Theorem is a direct corollary of the following proposition.

Proposition 4. *If there exists an oracle $\mathcal{O}_{\mathrm{El}}$ such that*

$$\Pr[\mathcal{O}_{\mathrm{El}}(p, g, \mathrm{El}_{p,g,a,b}(x)) = \mathrm{bit}_i(x)] - \frac{1}{2} \text{ is non-negligible,}$$

probability taken over random choices of $(p, g) \in I_{\mathrm{El}}(n)$ and $x, a, b \in \mathbb{Z}_p^*$. Then there exists an oracle pptm $\mathcal{D}^{\mathcal{O}_{\mathrm{El}}}$ such that

$$\Pr[\mathcal{D}^{\mathcal{O}_{\mathrm{El}}}(p, g, \mathrm{El}_{p,g,a,b}(x)) = x] \text{ is non-negligible,}$$

probability taken over all random choices of $(p, g) \in I_{\mathrm{El}}(n)$, $x, a, b \in \mathbb{Z}_p^*$ and the internal coin tosses of $\mathcal{D}^{\mathcal{O}_{\mathrm{El}}}$.

Note that given x and xg^{ab}, g^{ab} can be computed and hence the CDH assumption is broken.

Proof. Let p, g, g^a, g^b, xg^{ab} be given. We again only concentrate on the non-negligible fraction of (p, g) for that the oracle $\mathcal{O}_{\mathrm{El}}$ has an non-negligible advantage over random guessing $bit_i(x)$. Now to simulate the hidden number oracle choose random $r, s \in \mathbb{Z}_p^*$ and set

$$\begin{aligned}\mathcal{O}_{N,x}(c) &:= \mathcal{O}_{\mathrm{El}}(p, g, g^{a+r}, g^{b+s}, c \cdot xg^{(a+r)(b+s)}) \\ &= \mathcal{O}_{\mathrm{El}}(p, g, \mathrm{El}_{p,g,a+r,b+s}(cx)) \\ &= \mathrm{bit}_i(cx),\end{aligned}$$

where the last equation holds if the oracle did not lie. According to Theorem 5, we get the hidden number x with non-negligible probability of success. □

3.8 Discrete Exponentiation Bits

Define $I_{\mathrm{EXP}} := \{(p, g) : p \text{ is a prime and } g \text{ is a primitive element of } \mathbb{Z}_p^*\}$ and $I_{\mathrm{EXP}}(n) := I_{\mathrm{EXP}} \cap (\{0,1\}^n \times \{0,1\}^n)$. Let $(p, g) \in I_{\mathrm{EXP}}$. The discrete exponentiation function on $\mathbb{Z}_p^*$ is defined by $\mathrm{EXP}_{p,g}(x) := g^x \bmod p$. Inverting the $\mathrm{EXP}_{p,g}$ function is believed to be computationally hard on $\mathbb{Z}_p^*$ (DL assumption).

Theorem 10. *Let p be a prime and $p-1 = 2^l q$ and $l < i \leq \log p$. Under the DL assumption, bit_i is a hard core of the discrete exponentiation function on $\mathbb{Z}_p^*$ for almost all primes p.*

By "almost all primes" it is to be understood that if p is chosen at random, the result holds with probability $1 - o(1)$. Note that given $\mathrm{EXP}_{p,g}(x)$ all the l least significant bits of x are "easy" since they can be found by the Pohlig-Hellman algorithm [22].

Proof. Let $\mathrm{EXP}_{p,g}(x) = g^x \bmod p$ and an oracle $\mathcal{O}_{\mathrm{EXP}}$ such that

$$\Pr[\mathcal{O}_{\mathrm{EXP}}(p, g, \mathrm{EXP}_{p,g}(x)) = \mathrm{bit}_i(x)] - \frac{1}{2} \text{ is non-negligible,}$$

probability over random $(p, g) \in I_{\mathrm{EXP}}(n)$, $x \in \mathbb{Z}_p^*$ and the internal coin flips of the oracle. Again we only need to concentrate on the non-negligible fraction of (p, g) for that the oracle $\mathcal{O}_{\mathrm{EXP}}$ has an non-negligible advantage over random guessing $\mathrm{bit}_i(x)$. Simulate a Hidden Number oracle for the hidden number x as follows:

$$\begin{aligned}\mathcal{O}_{p,x}(c) &:= \mathcal{O}_{\mathrm{EXP}}(p, g, \mathrm{EXP}_{p,g}(x)^c \bmod p) \\ &= \mathcal{O}_{\mathrm{EXP}}(p, g, \mathrm{EXP}_{p,g}(c \cdot x \bmod p - 1)) \\ &= \mathrm{bit}_i(cx \bmod p - 1),\end{aligned}$$

where the last equation holds if the oracle did not lie. According to Theorem 5 we get x with non-negligible probability or a non-trivial factor of q, if $p - 1 = 2^l q$ for odd q. This factor of q does not help us much, but it is shown in [13] that this only happens with probability $1 - o(1)$ when each n-bit prime is chosen at random. □

3.9 Paillier's Bits

At Eurocrypt'99 Paillier [21] proposed a new encryption scheme based on higher residuosity classes.
Let $I_{\text{PAL}} := \{(N, g) : N = p \cdot q, |p| = |q|, p, q \text{ prime }, g \in \mathbb{Z}^*_{N^2}, \text{ord}(g) \text{ is a non zero multiple of } N\}$ and $I_{\text{PAL}}(n) := I_{\text{PAL}} \cap (\{0,1\}^n \times \{0,1\}^{2n})$. It can be shown that the pair $(N, g) \in I_{\text{PAL}}$ induces the following bijection:

$$\begin{aligned} \mathcal{E}_{N,g} : \mathbb{Z}_N \times \mathbb{Z}^*_N &\to \mathbb{Z}^*_{N^2} \\ (x, z) &\mapsto g^x \cdot z^N \bmod N^2 \end{aligned}$$

That means that for every $\omega \in \mathbb{Z}^*_{N^2}$ there exists an unique pair $(x, z) \in \mathbb{Z}_N \times \mathbb{Z}^*_N$ such that $\omega = g^x \cdot z^N \bmod N^2$. We denote the first component with the uniquely defined $\text{Class}_{N,g}(\omega) := x$. Computing $\text{Class}_{N,g}(\cdot)$ is considered to be a hard problem.
Paillier's encryption scheme works as follows: To encrypt a message $x \in \mathbb{Z}^*_N$, select a random $\omega \in \mathbb{Z}^*_{N^2}$ such that $\text{Class}_{N,g}(\omega) = x$. This can be done efficiently. $\omega \in \mathbb{Z}^*_{N^2}$ is the cyphertext. It is shown in [21] that, knowing the factorization of N, computing $\text{Class}_{N,g}(\omega) = x$ is easy. It was already mentioned in [8] that the function $\text{Class}_{N,g}(\cdot)$ has a homomorphic property which is:

$$\text{Class}_{N,g}(xy \bmod N^2) = \text{Class}_{N,g}(x) + \text{Class}_{N,g}(y) \bmod N. \tag{1}$$

This property was then exploited to show that, if computing the function $\text{Class}_{N,g}(\cdot)$ is hard, then the least significant bit is a hard core of the function $\mathcal{E}_{N,g}$. Under some non-standard computational assumption it was then shown in the same paper that $O(N)$ individual bits and the collection of $O(N)$ bits (simultaneously) are a hard core of the function $\mathcal{E}_{N,g}$. We will improve the result about the $O(n)$ individual bits to "every bit is secure". Furthermore, we do not need the non-standard computational assumption.
Unfortunately our result can not be used to show the security of the collection of more than $O(\log N)$ bits. It is an open problem if for more than $O(\log N)$ simultaneous bits the assumption made in [8] is crucial.
From equation (1) it follows that for every $k \in \mathbb{Z}^*_N$

$$\text{Class}_{N,g}(x^c \bmod N^2) = c \cdot \text{Class}_{N,g}(x) \bmod N. \tag{2}$$

This establishes our multiplicative property.

Theorem 11. *If computing the function* $\text{Class}_{N,g}(\cdot)$ *is hard, then every bit is a hard core of the function* $\mathcal{E}_{N,g}$.

Proof. Let $\mathcal{E}_{N,g}(x, z) = g^x z^n \bmod N^2$ be given and assume there is an oracle $\mathcal{O}_{\mathcal{E}}$ such that

$$\Pr[\mathcal{O}_{\mathcal{E}}(N, g, \mathcal{E}_{N,g}(x, z)) = \text{bit}_i(x)] - \frac{1}{2} \text{ is non-negligible,}$$

probability over random $(N, g) \in I_{\text{PAL}}(n)$, $(x, z) \in \mathbb{Z}_N \times \mathbb{Z}^*_N$ and the internal coin flips of the oracle. Again we only need to concentrate on the non-negligible fraction of (N, g, z) for

that the oracle $\mathcal{O}_\mathcal{E}$ has an non-negligible advantage over random guessing $\text{bit}_i(x)$. We set $\omega := \mathcal{E}_{N,g}(x,z)$ and simulate a Hidden Number oracle for the hidden number x as follows:

$$\mathcal{O}_{N,x}(c) := \mathcal{O}_\mathcal{E}(N, g, \omega^c \bmod N^2),$$

which is $\text{bit}_i(\text{Class}_{N,g}(\omega^c))$ in case the oracle did not lie. Exploiting equation (2) this is $\text{bit}_i(cx \bmod N)$.
According to Theorem 5 we get x with non-negligible probability or a non-trivial factor of N which nevertheless allows us to compute x. This is a contradiction to the assumption that computing $\text{Class}_{N,g}(\cdot)$ is hard and thus the oracle $\mathcal{O}_\mathcal{E}$ cannot exist. □

3.10 Proof sketch of Theorem 5.

As already stated in the introduction, the proof of this Theorem is analogous to the one given in [13] to show the security of every RSA bit. Going through the details of the RSA proof one encounters that the oracle is only queried for $\mathcal{O}(c \cdot \text{RSA}(x))$ for a given c and $\text{RSA}(x)$. This is exactly the structure of the Hidden Number Problem. Due to space limitations we will only can **briefly sketch** the proof of the security of the least significant bit and for the special case when I is the set of all odd primes. This proof is due to Alexi et al. [1]. Now, what happens if N is not a prime? First, when N is *even* then one can use the Chinese remainder Theorem to concentrate only on the odd part of the modulus. So lets assume N is *odd*. When going deeper into the proof technique of [13] it becomes clear that one has to invert a special function φ modulo N. At this point we either can invert this function *or* we get a non-trivial factor of N. Again the reader is encouraged to read [13] for more details.

Proof sketch of Theorem 5, special case $i = 0$, the lsb. Fix a prime $p = N \in I$ and $x \in \mathbb{Z}_p^*$. Let there be a polynomial P and a hidden number oracle such that

$$\Pr[\mathcal{O}_{N,x}(c) = \text{lsb}(cx \bmod p)] \geq \frac{1}{2} + \frac{1}{P(n)},$$

probability taken over all $a \in \mathbb{Z}_p^*$ and the internal coin tosses. We have to show that we can compute x with non-negligible probability of success.
We first describe an inverting algorithm using a parity subroutine. Then we describe how to implement this parity subroutine using the hidden number oracle for the lsb.

Inverting Algorithm:

1. Chose at random $a, b \in \mathbb{Z}_p^*$.
2. Apply a Brent-Kung binary gcd procedure [6] to the instances ax and bx. This binary gcd procedure queries a *parity subroutine* to get parities of the form $\text{par}(k_1 a \cdot x)$ and $\text{par}(k_2 b \cdot x)$ for known k_1 and k_2. If the gcd algorithm is successful we get a d with $dx = \gcd(ax, bx)$.
3. If we were lucky in choosing a and b, then ax and bx are relatively prime, what happens with asymptotical probability of $6/\pi^2$. In this case we have $dx \bmod p = 1$ and therefore $x = d^{-1} \bmod p$.

There are two sources of errors: first the parity subroutine can err and secondly the elements a and b can be chosen "unluckily". But a detailed analysis shows that the success probability of the inverting algorithm still is non-negligible.

It leaves to show how to implement the *parity subroutine*. The parity of an element x of $\mathbb{Z}_p^*$ is defined as $\mathrm{lsb}(x)$, if $x < p/2$, and as $lsb(p-x)$ otherwise. We only want to compute parities of the form $\mathrm{par}(dx)$ for a given d. We further assume that dx is "small".

Parity subroutine:
Pick a random $r \in \mathbb{Z}_p^*$ and query the oracle for the lsb of rx and $(r+d)x$. Since dx is small, with very high probability no wrap around 0 occurs when adding rx to dx. If no such wrap around occurs, then $\mathrm{par}(dx) = 1 \iff \mathrm{lsb}(rx) \neq \mathrm{lsb}((r+d)x)$. Now do a majority decision over many randomly chosen r to determine, with overwhelming probability of success, the parity of dx.

The problem is that the lsb oracle now may err on *both* ends $\mathrm{lsb}(rx)$ and $\mathrm{lsb}((r+d)x)$. To get around with this a technique called "pairwise independent sampling" may be applied. See [1] for more details. □

4 Hard Core Predicates and Universal Hash Functions

4.1 Definitions

Definition 4 (hash family). *An $(N; n, m)$-hash family is a set $\mathcal{H}$ of N functions $h : X \to Y$ where $|X| = n$ and $|Y| = m$.*

There will be no loss in generality assuming $n \geq m$.

Definition 5 (ε-universal hash family). *An $(N; n, m)$-hash family is called* ε-universal *provided that for any two distinct elements $x_1, x_2 \in X$, there exist at most εN functions $h \in \mathcal{H}$ such that $h(x_1) = h(x_2)$.*

Using the notation of probability we write for two distinct $x_1, x_2 \in X$:

$$\Pr[h(x_1) = h(x_2)] \leq \varepsilon,$$

where the function h is picked at random and uniformly from the set $\mathcal{H}$. The special case $\varepsilon = \frac{1}{m}$ is known as *universal hashing*. This definition was first given in 1979 by Carter and Wegman [7].

Three examples of universal hash functions are common in the literature:

- Multiplication by a randomly chosen boolean matrix over GF[2].
- Linear functions on GF[2^n].
- Linear functions on $\mathbb{Z}_p$.

Theorems 1 and 2 show that all these universal hash functions give hard core predicates. Goldmann and Näslund [9] investigated how efficiently general hard core functions can be

computed. As a measure of simplicity/complexity the model of *small depth circuits* is used. That is, how large/deep a circuit of Boolean AND/OR/NOT-gates is needed to compute a hard core function. All of the constructions of general hard core functions mentioned above can be computed with circuits of depth logarithmic (in n), polynomial size, and constant fan-in. In [9] it is shown that these are the simplest possible constructions. Another approach to determine the complexity of general hard core functions in terms of bounds on Fourier spectra was taken by Goldmann and Russell [10]. It is an interesting fact that precisely the same lower bounds hold for universal hash functions, see the paper by Mansour, Nisan and Tiwari [17].

In summary, on one hand all known constructions of general hard core functions rely on universal hash functions, and on the other hand hard core functions and universal hash functions share the same complexity lower bounds. This raised the question first asked by Näslund [18] in his 1995 paper "Universal Hash Functions & Hard Core Bits" and later in [10, 19], whether there is a "nice connection" between universal hash functions and hard core functions. We formalize this conjecture as follows, where $\langle r \rangle$ denotes some integer encoding of the bitstring r.

Question 1. Let $H_n = \{h^1, \ldots, h^{N_n}\}$ be a set of universal hash functions, where $h^i : \{0,1\}^n \to \{0,1\}$. Let f be an arbitrary length-regular one-way function, and let g be defined by $g(x,r) := (f(x), r)$ where $\log_2 N_{|x|} = |r| = c \cdot |x|$ for a constant c. Let $b(x,r) := h^{\langle r \rangle}(x)$.
Question: Is the predicate b always a hard core of the function g?

In the next section we will construct a simple example that gives a negative answer to this question. We modify a given set of universal hash functions that gives a hard core predicate and get a new set by first applying a one-way permutation f and then the universal hash function. Then we show that the constructed set of universal hash functions is not a hard core for this specific one-way permutation f.

4.2 Universal Hash Functions Giving no Hard Cores

For the counterexample we will use a slightly modified version of the discrete exponentiation on the group $\mathbb{Z}_p^*$ as the one-way function. For $(p, \alpha) \in I_{\text{EXP}}$, the discrete exponentiation function is defined as $\text{EXP}_{p,\alpha} = \alpha^x \bmod p$. We denote some integer encoding of u and v by $\langle u, v \rangle$.

Construction 1 *For a fixed n-bit prime p and a primitive element α of $\mathbb{Z}_p^*$ let $h_p^{\langle u,v \rangle}(x) := (ux + v \bmod p) \bmod 2$ and let $S_p = \{h_p^{\langle u,v \rangle}; u \in \mathbb{Z}_p^*, v \in \mathbb{Z}_p\}$ be the $(N_p; p, 2)$-set of universal hash functions where $N_p = p(p-1)$. Let $f_{p,\alpha} : \mathbb{Z}_p \to \mathbb{Z}_p$ be the one-way permutation defined as*

$$f_{p,\alpha}(x) := \begin{cases} 0 & : \quad x \equiv 0 \\ \alpha^x \bmod p & : \quad \textit{otherwise} \end{cases}$$

and let $g_{p,\alpha}$ be defined by $g_{p,\alpha}(x,u,v) := (f_{p,\alpha}(x), u, v)$.
Define the $(N_p; p, 2)$-set of universal hash functions

$$H_{p,\alpha} = \{h_p^{\langle u,v \rangle} \circ f_{p,\alpha}; u \in \mathbb{Z}_p^*, v \in \mathbb{Z}_p\}.$$

Further let

$$b_{p,\alpha}(x,u,v) := h_p^{\langle u,v\rangle} \circ f_{p,\alpha}(x) = \begin{cases} v \bmod 2 & : \quad x \equiv 0 \\ u\alpha^x + v \bmod p \bmod 2 & : \quad \textit{otherwise,} \end{cases}$$

where $u \in \mathbb{Z}_p^*$, $v \in \mathbb{Z}_p$.

Proposition 5. *Let* $b_{p,\alpha}$ *and* $g_{p,\alpha}$ *be defined as in Construction 1. Then the predicate* $b_{p,\alpha}$ *is not a hard core of the function* $g_{p,\alpha}$.

Proof. Since the discrete exponentiation function $\mathrm{EXP}_{p,\alpha}$ is a bijective one-way function on $\mathbb{Z}_p^*$, the function $f_{p,\alpha}$ is a natural extension of $\mathrm{EXP}_{p,\alpha}$ to a one-way permutation on $\mathbb{Z}_p$. S_p is a set of universal hash functions (see for example Carter and Wegman [7]). Because $f_{p,\alpha}$ is a permutation on $\mathbb{Z}_p$, $H_{p,\alpha}$ is a set of universal hash functions, too.
Now, to follow definition 1, the algorithm $\mathcal{D}$ (predictor), given $g_{p,\alpha}(x,u,v) = (f_{p,\alpha}(x),u,v)$ and (the coding of) the mapping $b_{p,\alpha}$, has to compute $b_{p,\alpha}(x,u,v)$ with success probability significantly greater than $1/2$. The predicate $b_{p,\alpha}$ is constructed as $b_{p,\alpha}(x) = h_p^{\langle u,v\rangle} \circ f_{p,\alpha}(x)$. Because $S_p = \{h_p^{\langle u,v\rangle}; u \in \mathbb{Z}_p^*, v \in \mathbb{Z}_p\}$ is a set of universal hash functions there must be an efficient algorithm $\mathcal{S}_p$ that computes $h_p^{\langle u,v\rangle}(x)$ for every input tuple (x,u,v).
Now define the output of algorithm $\mathcal{D}$ as $\mathcal{D}_{p,\alpha}(x,u,v)) := \mathcal{S}_p(f_{p,\alpha}(x),u,v) = b_{p,\alpha}(x,u,v)$. Then for all n $\Pr[\mathcal{D}(g_{p,\alpha}(x)) = b_{p,\alpha}(x)] = 1$, where x is chosen from $\{0,1\}^n$ at uniform distribution. Therefore $b_{p,\alpha}$ is not a hard core predicate for the one-way function $g_{p,\alpha}$. □

Note that proposition 5 now is giving a negative answer to Question 1 on the relation between universal hash functions and hardcore predicates.
This example can be generalized to any set S of universal hash functions where $s : X \to \{0,1\}$ for $s \in S$ and any one-way *permutation* f on X.

Construction 2 *Let* $S_n = \{h^1, \ldots, h^{N_n}\}$ *be a set of universal hash functions where* $h^i : \{0,1\}^n \to \{0,1\}$ *for every* i. *Let* f *be an arbitrary one-way permutation, and let* g *be defined by* $g(x,r) := (f(x),r)$ *where* $\log_2 N_{|x|} = |r| = c \cdot |x|$ *for a constant* c.
Define the set of universal hash functions $H_n = \{h^1 \circ f, \ldots, h^{N_n} \circ f\}$ *and let* $b(x,r) := h^{\langle r\rangle} \circ f(x)$.

Proposition 6. *Let* b *and* g *be defined as in Construction 2. Then the predicate* b *is not a hard core of the function* g.

The simple proof is analogous to that of proposition 5.

4.3 A 2/3-universal Hash Function Giving a Hard Core

In this section we present an "almost universal" set of hash functions giving a hard core. For a prime p, let $\mathcal{H}_{p,2}^{\mathrm{mult}} = \{h_r(x) = (rx \bmod p) \bmod 2, r \in \mathbb{Z}_p^*\}$. From Theorem 3 it follows that $b(x,r) := (rx \bmod p) \bmod 2$ is a hard core of any one-way function of the form $g(x,r) = (f(x),r)$.

Proposition 7. $\mathcal{H}_{p,2}^{\mathrm{mult}}$ *is* $\frac{2}{3}$*-universal. Moreover, for infinitely many primes, this set is not* ε*-universal for any* $\varepsilon < \frac{2}{3}$.

To this end, we introduce the following notation:

$$\tau_p(h_0, h_1) := \frac{1}{p-1} \cdot |\{z \in \mathbb{Z}_p^* : \mathrm{lsb}(h_0 z) = \mathrm{lsb}(h_1 z)\}|.$$

In other words, $\tau_p(h_0, h_1)$ is the fraction of prime residues z on which $h_0 z$ and $h_1 z$ share the least significant bit. Trivial equalities are $\tau_p(h, h) = 1, \tau_p(h, -h) = 0$ and $\tau_p(h_0, -h_1) = 1 - \tau_p(h_0, h_1)$. We may therefore focus on the case $h_1 \neq h_0, h_1 \neq p - 1 - h_0$. Another (more or less) easy observation is that

$$\tau_p(h_0, h_1) = \tau_p(1, h_0^{-1} h_1).$$

We therefore focus on the case $h_0 = 1$ and $h_1 \neq 1, h_1 \neq p - 2$. For the sake of simplicity, we set $\tau_p(h) := \tau_p(1, h)$. Note that $\tau_p(-h) = 1 - \tau_p(h)$.

Lemma 1 ([14]). *For each prime p and for each $h \in \mathbb{Z}_p^* \setminus \{1, p-2\}$, $\frac{1}{3} \leq \tau_p(h_0, h_1) \leq \frac{2}{3}$. Moreover, for each prime of the form $p = 6q + 1$, for each $h \in \{3, 4q + 1\}$ and each $h' \in \{6q - 2, 2q\}$, we have $\tau_p(h) = \frac{2}{3}$ and $\tau_p(h') = \frac{1}{3}$.* [4]

Proof of proposition 7. For two distinct elements $x, y \in \mathbb{Z}_p^*$ we get from lemma 1:

$$\Pr[h(x) = h(y)] = \Pr[\mathrm{lsb}(rx) = \mathrm{lsb}(ry)] = \tau_p(x, y) = \tau_p(x^{-1} y) \leq \frac{2}{3},$$

where the first probability refers to a uniformly chosen $h \in \mathcal{H}_{p,2}^{\mathrm{mult}}$ and the second to a uniformly chosen $r \in \mathbb{Z}_p^*$. This equation (and again lemma 1) also shows that for infinitely many primes $\frac{2}{3}$ is minimal regarding the universality of $\mathcal{H}_{p,2}^{\mathrm{mult}}$. □

4.4 A new Construction of Hard Cores of any One-way Permutation

Let h be a hard core predicate for any (modified) one-way function. In this subsection, we pursue the following question: Which permutations $\varphi(x)$ lead to hard-core predicates $h(\varphi(x), r)$, given a one-way function of the form $g(x, r) = (f(x), r)$?
A function $\phi : \{0,1\}^* \to \{0,1\}^*$ is said to be *hard to compute*, if for every pptm $\mathcal{A}$, $\Pr[\mathcal{A}(x) = \phi(x)]$ is negligible in n, where x is chosen at random from $\{0,1\}^n$ according to the uniform distribution.

Proposition 8 (General construction). *Let $h(y, r)$ be a hard core predicate for any length-regular one-way function of the form $g'(y, r) = (f'(y), r)$.*
Let $f : \{0,1\}^* \to \{0,1\}^*$ be a permutation that is hard to invert and $\varphi : \{0,1\}^* \to \{0,1\}^*$ be an arbitrary permutation such that $f \circ \varphi^{-1}$ is computable in polynomial time. Define $g(x, r) := (f(x), r)$ and $b(x, r) := h(\varphi(x), r)$.
Then the predicate b is a hard core of g iff $\varphi \circ f^{-1}$ is hard to compute.

[4] We briefly note that, according to a Theorem of Dirichlet [16], infinitely many primes actually are of the form $cq + d$, if (c, d) is a fixed pair of pairwise prime positive integers (e.g., $c = 6$, $d = 1$).

Proof. Let $\varphi \circ f^{-1}$ be not hard to compute, i.e. assume there exists a pptm $\mathcal{A}$ for which $\varepsilon_n := \Pr[A(y) = \varphi \circ f^{-1}(y)]$ is non-negligible, where the probability refers to a uniformly chosen $y \in \{0,1\}^n$. Since $f \circ \varphi^{-1}$ is computable in polynomial time, we can test if a given output of $\mathcal{A}$ is correct. Now, to follow Definition 1, define the output of algorithm $\mathcal{D}$ (predictor) as

$$\mathcal{D}(g(x,r)) := \begin{cases} h(\mathcal{A}(f(x)), r) & : \text{ if } \mathcal{A} \text{ is correct} \\ \text{random bit} & : \text{ otherwise.} \end{cases}$$

If $\mathcal{A}$ is correct, $\mathcal{D}$ outputs $h(\mathcal{A}(f(x)), r) = h(\varphi \circ f^{-1} \circ f)(x), r) = h(\varphi(x), r) = b(x, r)$. If $\mathcal{A}$ was wrong, $\mathcal{D}$ outputs a random bit which is the correct answer with probability at least $\frac{1}{2}$. Thus $\Pr[\mathcal{D}(g(x,r)) = b(x,r)] \geq \varepsilon_n \cdot 1 + (1 - \varepsilon_n) \cdot \frac{1}{2} = \frac{1}{2} + \frac{1}{2} \cdot \varepsilon_n$ and b is not a hard core of g.
Now let $\varphi \circ f^{-1}$ be hard to compute. Assume that b is not a hard core of g. Then there is an oracle $\mathcal{A}$ that, given $g(x,r) = (f(x), r)$, computes the predicate $b(x,r) = h(\varphi(x), r)$ with probability significantly better than random guessing.
For any y set $x = \varphi^{-1}(y)$ and use $\mathcal{A}(f \circ \varphi^{-1}(y), r)$ to compute $h(\varphi(\varphi^{-1}(y)), r) = h(y, r)$ with probability significantly better than random guessing. This contradicts to the fact that $h(y,r)$ is a hard core of the one-way permutation $g'(y,r) = (f \circ \varphi^{-1}(y), r)$. □

Corollary 1. *Let $f : \{0,1\}^* \to \{0,1\}^*$ be a one-way permutation and $\varphi : \{0,1\}^* \to \{0,1\}^*$ be an arbitrary permutation such that $f \circ \varphi^{-1}$ is computable in polynomial time. For every $j = j(n) \in \{0, \ldots, n-1\}$ define $b_j(x,r) := \mathrm{bit}_j(r\varphi(x) \bmod p)$, where $n = |x|$, p is a n-bit prime and $r \in \mathbb{Z}_p^*$. Let g be the one-way permutation defined by $g(x,r) := (f(x), r)$.*
Then the predicate b_j is a hard core of the one-way permutation g iff $\varphi \circ f^{-1}$ is hard to compute.

Example 1. Let f be an arbitrary length-regular one-way function, and let g be defined by $g(x,r) := (f(x), r)$ where $n = |x|$, p is a n-bit prime and $r \in \mathbb{Z}_p^*$. For every $j = j(n) \in \{0, \ldots, n-1\}$ define $b_j(x,r) := \mathrm{bit}_j(\frac{r}{x} \bmod p)$. Then the predicate b_j is a hard core of the function g.

This follows from corollary 1 using $\varphi(x) := x^{-1}$ and the fact that $\frac{1}{f^{-1}(y)}$ is hard to compute iff $f^{-1}(y)$ is.

5 Conclusions

We introduced a cryptographic primitive, the hidden number problem, which has multitude of applications in the context of bitsecurity. This primitive can be use to prove that *every individual bit* as well as *every $O(\log n)$ simultaneous bits* are a hard core of the following well known cryptographic functions: The RSA encryption function, the ElGamal encryption function, a Modified Diffie-Hellman function, the Diffie-Hellman function on Elliptic Curves, the Discrete Exponentiation function, the Rabin encryption function and Paillier's encryption function.
Whereas the results for the RSA, Rabin and Discrete Exponentiation case where known beforehand the other cases are completely new results.

Furthermore the hidden number problem provides a new method to get $O(\log n)$ simultaneous hard bits of any (modified) one-way function. This method could be shown to be more efficient than the original Goldreich-Levin method.
Furthermore we could disprove a conjecture made by Näslund about the connection between universal hash functions and hard core predicates. We gave an example of a UHF giving no hard core predicate. On the other hand we gave a hash function that is "only" 2/3 universal and still giving a hard core predicate.

Acknowledgment. This paper greatly benefited from the helpful insight of Hans Ulrich Simon. Furthermore I send handshakes to Jürgen Forster and Mats Näslund for interesting discussions and fruitful comments.

References

1. W. Alexi, B. Z. Chor, O. Goldreich, and C.-P. Schnorr. RSA and Rabin functions: Certain parts are as hard as the whole. *SIAM Journal on Computing*, 17(2):194–209, April 1988. Special issue on cryptography.
2. M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudorandom bits. *SIAM Journal on Computing*, 13(4):850–864, 1984.
3. D. Boneh. The decision Diffie-Hellman problem. *Lecture Notes in Computer Science*, 1423:48–63, 1998.
4. D. Boneh and I. Shparlinski. Hard core bits for the elliptic curve Diffie-Hellman secret. In *Advances in Cryptology – CRYPTO ' 2001*, Lecture Notes in Computer Science. Springer-Verlag, Berlin Germany, 2001.
5. D. Boneh and R. Venkatesan. Hardness of computing the most significant bits of secret keys in Diffie-Hellman and related schemes. *Proc. of CRYPTO 1996*, pages 129–142, 1996.
6. R. P. Brent and H. T. Kung. Systolic VLSI arrays for polynomial gcd computation. *IEEE Transactions on Computers*, 33(8):731–736, August 1984.
7. J. L. Carter and M. N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18(2):143–154, April 1979.
8. D. Catalano, R. Gennaro, and N. A. Howgrave-Graham. The bit security of Paillier's encryption scheme and its applications. *Proc. of EUROCRYPT'01. Lecture Notes in Computer Science*, 2045:229–244, 2001.
9. M. Goldmann and M. Näslund. The complexity of computing hard core predicates. *Proc. of CRYPTO'97. Lecture Notes in Computer Science*, 1294:1–15, 1997.
10. M. Goldmann and A. Russell. Spectral bounds on general hard core predicates. In *Proc. of STACS*, pages 614–625, 2000.
11. O. Goldreich. *Foundations of Cryptography (Fragments of a Book).* http://theory.lcs.mit.edu/~oded/frag.html, 1995.
12. O. Goldreich and L. Levin. A hard-core predicate for all one-way functions. *Proc. of STOC*, pages 25–32, 1989.
13. J. Håstad and M. Näslund. The security of all RSA and discrete log bits. *ECCC Report TR99-037*, pages 1–48, 1999.
14. E. Kiltz and Hans Ulrich Simon. manuscript. *unpublished*, 2001.
15. Eike Kiltz. A useful primitive to prove security of every bit and about hard core predicates and universal hash functions. *Proc. of FCT'01. Lecture Notes in Computer Science*, 2138:388–392, 2001.
16. E. Landau. *Handbuch der Lehre von der Verteilung der Primzahlen.* Teubner Verlag, 1909.
17. Y. Mansour, N. Nisan, and P. Tiwari. The computational complexity of universal hashing. *Theoretical Computer Science*, 107(1):121–133, January 1993.
18. M. Näslund. Universal hash functions & hard core bits. *Lecture Notes in Computer Science*, 921:356–366, 1995.
19. M. Näslund. All bits in $ax + b \bmod p$ are hard (extended abstract). In *Proc. of CRYPTO '96*, pages 114–128, 1996.
20. P. Q. Nguyen and J. Stern. The two faces of lattices in cryptology. *Proc. of Cryptography and Lattices Conference. Lecture Notes in Computer Science*, 2146, 2001.
21. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. *Proc. of EUROCRYPT'99*, pages 223–238, 1999.
22. S. C. Pohlig and M. Hellman. An improved algorithm for computing logarithms over GF(p). *IEEE Trans. on Information Theory*, pages 106–110, 1978.

23. M. I. González Vasco and M. Näslund. A survey of hard core functions. *Proc. Workshop on Cryptography and Computational Number Theory*, pages 177–195, 2001.
24. M. I. González Vasco and I. Shparlinski. On the security of diffie-hellman bits. *Proc. Workshop on Cryptography and Comp. Number Theory (CCNT'99)*, 2000.