Universal Hash Functions & Hard Core Bits

Mats Näslund

Royal Institute of Technology,
Dept. of Numerical Analysis and Computing Science,
S-100 44 Stockholm, Sweden
email: matsn@nada.kth.se

Abstract. In this paper we consider the bit-security of two types of universal hash functions: linear functions on GF[2^n] and linear functions on the integers modulo a prime. We show individual security for all bits in the first case and for the $O(\log n)$ least significant bits in the second case. Both types of functions are shown to have $O(\log n)$ simultaneous secure bits. For the second type of functions, primes of length $\Omega(n)$ are needed.
Together with the Goldreich-Levin theorem, this shows that all the common types of universal hash functions provide so called hard-core bits.

1 Introduction

Most cryptographic protocols are based on the access to some source of random bits. Examples of such protocols are private key crypto systems, authentication schemes, commitment schemes etc. For practical purposes it is desirable to reduce the number of true random bits needed. Instead we would like to deterministically expand a short truly random sequence into a longer one that is "just as good" as a truly random sequence of the same length. In other words we would like to deterministically produce some "extra" bits that "look" totally random. So called *hard-core* bits can serve as these extra bits. Intuitively, a hard-core bit is a 0–1 function that cannot be approximated essentially better than simply guessing it.

Another common technique is to try to make "slightly random" sources more random looking. This can be achieved by means of *universal hash functions*, first introduced by Carter & Wegman in [2]. Such hash functions will map elements pairwise independently and the image of each element will be uniformly distributed. Universal hash functions have been used extensively in the construction of pseudo random number generators (PRG's), see for instance [5].

This paper is concerned with the relation between universal hash functions and hard-core bits. The first function that was shown to provide hard-core bits is the usual inner product taken modulo 2. This was done by Goldreich and Levin in [4]. This inner product was from [2] known to be a universal hash function. The natural question is therefore: Can we obtain hard-core bits from the other known universal hash functions as well? More generally, does *every* universal hash function have such hard bit(s)? Not very surprisingly we will answer the first question positively. Even if the second question also could be answered

L.C. Guillou and J.-J. Quisquater (Eds.): Advances in Cryptology - EUROCRYPT '95, LNCS 921, pp. 356-366, 1995.

positively, some new proof technique seems to be needed and we must leave this an open problem.

After giving some notation we first go about hash functions given by linear functions in a finite field of characteristic 2. Here we show that for randomly chosen $a, x, b \in \mathrm{GF}[2^n]$, any single bit of the function $x \mapsto ax + b$ is a hard-core bit. Also, the $O(\log n)$ least significant bits are shown to be simultaneously hard-core. Next we study hash functions obtained as linear functions on the integers modulo a prime. Using an adaptation of the techniques used by Chor et al. in [1], [3], we are able to prove both individual and simultaneous hardness for $O(\log n)$ bits. Primes of length $\Omega(n)$ are needed though.

2 Preliminaries

The model of computation used is that of probabilistic Turing machines. We will only be interested in such machines that run in time polynomial in the length of the input, pptm's for short. The length of the input to such a machine is referred to by n. In general, we denote by $|y|$ the length of the binary string y. By $x \in_U S$ we mean an x chosen from the set S according to the uniform distribution. If S is a set, $||S||$ is the cardinality of S.

We call a function $g(n)$ *negligible* if for every constant $c > 0$ and for every sufficiently large n, $g(n) < n^{-c}$.

By a *one-way function* we mean a function f such that for every pptm, M, the probability that M on input $f(x)$ finds an $x' \in f^{-1}(x)$ is negligible. The probability is taken over $x \in_U \{0,1\}^n$ and M's random coin flips. For simplicity all one-way function in this paper are assumed to be *length-preserving*, i.e. $|f(x)| = |x|$.

Let H be an efficiently sampleable family of functions were each $h \in H$ is computable in deterministic polynomial time and maps $\{0,1\}^n \mapsto \{0,1\}^{l(n)}$, $l(n) \leq n$. Let f be a one-way function. An *approximation algorithm* for H is a pptm that on input $f(x)$ and the description of $h \in H$ tries to compute $h(x)$. We call H a family of *hard-core functions for f with security $s(n)$* if for all approximation algorithms A: $Pr[A(f(x), h) = h(x)] < 2^{-l(n)} + s(n)$. The probability is taken over $x \in_U \{0,1\}^n$, $h \in_U H$ and A's random choices. (When $l(n) = 1$ we have a family of hard-core predicates.) We will sometimes just say that H is $s(n)$-secure and if H is $s(n)$-secure for *all* non-negligible $s(n)$ we simply call H a family of hard-core functions. If indeed for some A and $s(n)$, $Pr[A(f(x), h) = h(x)] \geq 2^{-l(n)} + s(n)$ holds, we call A an $s(n)$-oracle for H.

The following fact (a version of the Goldreich-Levin theorem from [4]) will be useful:

Fact 1. *Let $\langle r, x \rangle$ denote the inner product, $\sum_{i=1}^{n} r_i x_i$. The family of functions: $b_r(x) = \langle x, r \rangle \pmod 2$ for $r \in_U \{0,1\}^n$ is a family of hard-core predicates for any one-way function.*

Generalizing, any one-way function has a family of hard-core functions, B, defined as follows: Let $k \in O(\log n)$ and let $r_1, r_2, \ldots, r_k \in_U \{0,1\}^n$. Then $B_{r_1, r_2, \ldots, r_k}(x) = b_{r_1}(x) \circ b_{r_2}(x) \circ \cdots \circ b_{r_k}(x)$, where $\circ$ means concatenation.

Finally, a family of *(strong) universal hash functions* (UHF's) is a set of functions, $H_{n,m}$, with each $h \in H_{m,n}$ such that $h : \{0,1\}^n \mapsto \{0,1\}^m$, $m \leq n$, and for any $x_1 \neq x_2 \in \{0,1\}^n$ and any $y_1, y_2 \in \{0,1\}^m$:

$$Pr_{h \in H_{n,m}}[h(x_1) = y_1 \wedge h(x_2) = y_2] = 2^{-2m}.$$

Throughout this paper $y_{<k>}$ denotes the k least significant bits in y. For the special case $y_{<1>}$, we write $\mathrm{lsb}(y)$.

3 Hash functions given by linear functions on GF[2^n]

3.1 Notation

We will here study the family $\{h_{A,B}(X) = A(t)X(t) + B(t) \mid A, B \in_U \mathrm{GF}[2^n]\}$. As usual, GF[$2^n$] is the field of 2^n elements. We assume that we have a representation of the field as $\mathbb{Z}_2[t]/Q(t)$ where Q is an irreducible polynomial of degree n, $\sum_{i=0}^{n} Q_i t^i$. We map elements $x \in \{0,1\}^n$ to GF[2^n] in the natural way by

$$\phi(x_{n-1}x_{n-2}\cdots x_0) = \sum_{i=0}^{n} x_i t^i = X(t).$$

We will use small letters (such as x) when we refer to values as binary strings and capital letters (such as X) when we have polynomials.

The notion of lsb is not well defined in GF[2^n]. However it turns out that what bit we interpret as least significant really does not matter so let us for the moment define it as the constant term in the polynomials. Finally, note that the addition below is modulo 2.

3.2 Security of lsb($A(t)X(t) + B(t)$)

We aim to show

Theorem 2. *The family* $\{lsb(h_{A,B}(X)) \mid A, B \in_U GF[2^n]\}$ *is a family of hard-core predicates, i.e. it is* n^{-c} *secure for any* $c > 0$.

The idea behind the proof is simple and quite intuitive. We will set up a 1–1 correspondence between the usual inner-product bit mod 2 and the lsb in the above representation.

Observe that for any two polynomials $P(t), S(t)$ we have $\mathrm{lsb}(P(t)+S(t)) = \mathrm{lsb}(P(t)) + \mathrm{lsb}(S(t))$ so that we, for the moment, can forget about $B(t)$ above. As before, let $\langle x, r\rangle$ be the inner product of the n-vectors r and x.

Lemma 3. *Given any* $r \in \{0,1\}^n$ *there is a unique polynomial* $R(t) \in GF[2^n]$ *such that for all* $x \in \{0,1\}^n$ *we have*

$$\langle x, r\rangle \pmod 2 = lsb(\phi(x)R(t)). \tag{1}$$

Furthermore, $R(t)$ *can be found in polynomial time.*

Proof. Let L_1 be the set of all linear functions $\{0,1\}^n \mapsto \{0,1\}$ and L_2 the set of all linear functions $\mathrm{GF}[2^n] \mapsto \{0,1\}$. First note that distinct r's define distinct linear functions: $\langle x, r\rangle \pmod 2$ and distinct polynomials $R(t)$ define distinct linear functions $\mathrm{lsb}(R(t)X(t))$. Since $||L_1|| = ||L_2|| = 2^n$, all function in L_1 can be expressed as $\langle x, r\rangle \pmod 2$ for some $r \in \{0,1\}^n$ and all functions in L_2 can be written as $\mathrm{lsb}(R(t)X(t))$ for some $R(t) \in \mathrm{GF}[2^n]$. We conclude that there is a bijection between the respective set of linear functions. Finally, given the values of $\langle x, r\rangle \pmod 2$ for n lineary independent x's we can in polynomial time, using standard linear algebra methods, find a corresponding polynomial $R(t) \in \mathrm{GF}[2^n]$ such that (1) is satisfied for all $\phi(x) = X(t) \in \mathrm{GF}[2^n]$. □

We can now reduce the problem of approximating the inner product to the problem of approximating $\mathrm{lsb}(h_{A,B}(X))$. Theorem 2 then follows from Fact 1:

Proof. (Of Theorem 2.) Let T be an $\delta(n)$-oracle for $\mathrm{lsb}(A(t)X(t) + B(t))$. On input $r \in_U \{0,1\}^n, y = f(x)$, do the following: Using Lemma 3 find the unique $R(t) \in \mathrm{GF}[2^n]$ such that $\langle r, x\rangle \pmod 2 = \mathrm{lsb}(R(t)\phi(x))$ for all $x \in \{0,1\}^n$. Choose $B(t) \in_U \mathrm{GF}[2^n]$ and run T on input (R, B, y). Suppose that T outputs γ. Output $\mathrm{lsb}(B(t)) + \gamma$.

If r is uniformly distributed in $\{0,1\}^n$, R will be uniformly distributed in $\mathrm{GF}[2^n]$. Thus the success probability is exactly the same as that of T. The reduction is clearly polynomial time.

Summing up, if $\delta(n)$ is non-negligible, we now have an approximation algorithm for the inner product bit, also with non-negligible success probability, a contradiction to the Goldreich-Levin theorem. □

In fact there is nothing special about the least significant bit.

Corollary 4. *Any single fixed bit position (coefficient) in the family* $\{h_{A,B}(X) \mid A, B \in_U GF[2^n]\}$ *is a family of hard-core predicates.*

Proof. The proof is the same as before. Just note that each single bit of $R(t)X(t)$ is a linear function $\mathrm{GF}[2^n] \mapsto \{0,1\}$. □

3.3 Simultaneous security

The same technique as above can be used to prove

Theorem 5. *Let c be a constant. The family* $\{(A(t)X(t)+B(t))_{<c \log n>} \mid A, B \in_U GF[2^n]\}$ *is a family of hard-core functions. In general any set of* $O(\log n)$ *bits constitute a family hard-core functions.*

The theorem will follow from the following two lemmas, the first being the well known XOR-lemma, see [6]. A function, $h : \{0,1\}^n \mapsto \{0,1\}^{l(n)}$, is called *length-regular* if $l(n)$ increases with n.

Lemma 6. *Let* $\{h\}$ *be a set family of length-regular function with* $|h(x)| \in O(\log n)$. *Then* $\{h\}$ *is a family of hard-core functions if, and only if, the exclusive-or of any non-empty subset of its bits is a family hard-core predicates.*

Next we show that the problem of approximating $\mathrm{lsb}(A(t)X(t)+B(t))$ can be reduced (in polynomial time) to that of approximating the exclusive-or of any non-empty subset of the bits of $A(t)X(t)+B(t)$.

Lemma 7. *Let S be a non-empty subset of $\{0,1,2,\ldots,n-1\}$ and let $l_i(z)$ be the i:th bit of z. Given $R(t) \in GF[2^n]$ there is a unique polynomial $P(t) \in GF[2^n]$ such that for all $X(t) \in GF[2^n]$:*

$$lsb(R(t)X(t)) = \sum_{i \in S} l_i(P(t)X(t))$$

and where P can be found in time polynomial in n.

Proof. For each $i = 0,1,\ldots,n-1$, $l_i(P(t)X(t))$ is a linear function $\mathrm{GF}[2^n] \mapsto \{0,1\}$. Being a sum (mod 2) of such linear functions, $\sum_{i\in S} l_i(P(t)X(t))$ is also a linear function. Arguing as before it will suffice to show that distinct $R(t)$ define distinct functions.

Assume that for some $R(t) \neq 0$ we have $\sum_{i\in S} l_i(R(t)X(t)) = 0$ for all $X(t) \in \mathrm{GF}[2^n]$. Let $i_0 \in S$. Since we are working in a field there is some $W(t) \in \mathrm{GF}[2^n]$ such that $R(t)W(t) = t^{i_0}$. By assumption $\sum_{i\in S} l_i(R(t)W(t)) = 0$ but $\sum_{i\in S} l_i(R(t)W(t)) = \sum_{i\in S} l_i(t^{i_0}) = 1$, a contradiction. We must conclude that if $\sum_{i\in S} l_i(R(t)X(t)) = 0$ is to hold for all $X(t) \in \mathrm{GF}[2^n]$ then $R(t) = 0$ and hence, distinct $R(t)$ give distinct linear functions. □

4 Hash functions given by linear functions in $\mathbb{Z}_p$

4.1 Notation

As usual, $\mathbb{Z}_p$ denotes the field of integers modulo a prime, p. By $\mathcal{P}_k$, $k > 0$, we mean the set of primes p of length n/k. Here, $n = |x|$, the security parameter of some one-way function $f(x)$. As in [1], [3] we devide $\mathbb{Z}_p$ into "positive" and "negative" elements. The positive being $\{1,2,\ldots,\frac{p-1}{2}\}$ and the negative $\{\frac{p-1}{2}+1, \frac{p-1}{2}+2,\ldots,p-1\}$. Thus it is natural to define an absolute value for each $x \in \mathbb{Z}_p$ by $|x|_p = x$ if $x \leq \frac{p-1}{2}$ and $|x|_p = p - x$ otherwise. If $|y|_p \leq \frac{p}{\gamma}$, we say that y is (γ,p)-small.

We shall need a notion of "oddness" and "eveness" and so we define the parity of $x \in \mathbb{Z}_p$ by $\mathrm{parity}(x,p) = \mathrm{lsb}(|x|_p)$. In this way the odd/even concept agrees with the intuition both for positive and negative x.

The hash functions we study here is the set $\{h_{a,b,p}(x) = ax + b \pmod p \mid p \in_U \mathcal{P}_k,\ a,b \in_U \mathbb{Z}_p\}$ Note that this set is not totally universal. The least significant bit has a $\frac{1}{p}$ "preference" for attaining the value 0. However this deviation tends to zero exponentially in $|p|$. For large p we can for all practical purposes consider the set to be universal. No polynomial time algorithm can distinguish this distribution from that of a "totally" universal hash function.

4.2 Security of $ax + b \pmod p$

In this section of the paper we show:

Theorem 8. *Let k be any positive constant, let $p \in_U \mathcal{P}_k$. The family*

$$\{lsb(h_{a,b,p}(x)) \mid a, b \in_U \mathbb{Z}_p\}$$

is a family of hard-core predicates for any one-way function.

The idea behind the proof is to show that an n^{-c}-oracle for $\text{lsb}(h_{a,b,p}(x))$ can be used to retrieve $x \pmod p$. We then repeat this process for several distinct p and combine these results using the Chinese remainder theorem to find $x \pmod{2^n}$.

For the first part we will use a modification of the ideas used by Chor et al. in [1], [3]. To get some motivation we review some of that work.

Previous work. Chor et al. in [1], [3] use a parity oracle to invert the RSA function by computing the gcd of $E_N(ax), E_N(bx)$ for random a, b. E_N is the RSA encryption function with composite modulus N. This can be done by observing that $E_N(ax) = E_N(a)E_N(x)$ and by using the well known bit-gcd procedure that only makes parity tests. We get a gcd of the form $E_N(lx)$ (with l known) and with probability $\frac{6}{\pi^2}$ for large N this gcd equals 1. Since $E_N(1) = 1$, x is easily found.

Using an lsb-oracle for parity. The first step is to convert an lsb-oracle to a parity-oracle. Suppose we have a known integer j and an unknown integer i and that we somehow are able to obtain information on $\text{lsb}(j)$ and $\text{lsb}(j + i)$. We can deduce the parity of i from this since the parity of i is 0 if and only if the lsb of j equals the lsb of $j + i$.

Is this true also in $\mathbb{Z}_p$? Not in general. If $j + i$ causes "overflow" mod p, the parity will be missrepresented. However it is easy to see that the probability of such overflow is $\frac{|i|_p}{p}$. Thus, if i is "small" mod p, we can deduce that

$$\text{parity}(i, p) = 0 \Leftrightarrow \text{lsb}(j) = \text{lsb}(j + i) \text{ for most } j.$$

(We shall shortly specify what is required to be considered small.) This gives a simple way to approximate $\text{parity}(ax + b \pmod p)$ using a lsb-oracle, O_l: On input a, b, p choose $c, d \in_U \mathbb{Z}_p$ and ask O_l about $\text{lsb}(cx + d \pmod p)$ and $\text{lsb}((a + c)x + (b + d) \pmod p)$. Output 0 if O_l answers the same to both questions, 1 otherwise.

To improve performance we can choose many (c, d)-pairs and take a majority decision. However, calling O_l twice for each (c, d)-pair has its drawbacks which we have reason to return to later.

We next show how we can use a "very good" parity oracle and then show how to obtain such from a "fairly good" lsb-oracle.

$(\frac{1}{2} - \frac{\alpha}{n})$-oracles for parity. Suppose we are given a prime p, $|p| = n/k$, chosen uniformly at random in $\mathcal{P}_k$ and that we have access to a $(\frac{1}{2} - \frac{\alpha}{n})$-oracle for parity, O_{par}, with $\alpha < k/2$. We can now try to retrieve $x \pmod p$ in the following fashion: Choose $a, b \in_U \mathbf{Z}_p$ and assume that $ax + b$ is "small". For instance, assume $ax + b$ is $(2n^c, p)$-small. This happens with probability $\frac{1}{2n^c}$. We now make the following observations: If parity$(ax + b, p) = 0$ then $ax + b \pmod p$ is divisible by 2, and $|2^{-1}(ax + b)|_p$ is even smaller than $|ax + b|_p$. On the other hand, If parity$(ax + b, p) = 1$ then $ax + b - 1 \pmod p$ is divisible by two and $|2^{-1}(ax + b - 1)|_p$ will be small. Eventually, after $|p|$ parity-tests (if they all gave correct answers), we end up with a representation $y \equiv a''x + b'' \pmod p$ with y, a'' and b'' known, and can easily find $x \pmod p$. Since we make exactly $|p| = n/k$ parity calls, the probability of getting one (or more) incorrect parity answer is at most $\frac{n}{k}\frac{\alpha}{n} < 1/2$. Finally note that if the initial $ax + b \pmod p$ is small and the oracle makes no errors, all successive $a'x + b' \pmod p$ will also be small. We have proved:

Lemma 9. *For a randomly chosen prime $p \in_U \mathcal{P}_k$ a $(\frac{1}{2} - \frac{\alpha}{n})$-oracle, $\alpha < k/2$, for parity$(ax + b \pmod p)$ for $(2n^c, p)$-small $ax + b$, can be used to find $x \pmod p$ with probability at least $\frac{1}{4n^c}$.*

Comment: There is in fact another way of proving this lemma by using the same gcd-technique as in [1], [3]: Choose $a, b, c, d \in_U \mathbf{Z}_p$ and use the bit-gcd algorithm (which only uses parity tests) to compute $e, f \in \mathbf{Z}_p$ such that $ex + f = \gcd(ax + p \pmod p, cx + d \pmod p)$. With probability greater than $\frac{1}{2}$, $ex + f \equiv 1 \pmod p$ and x can easily be found. The same analysis as in [1], [3] show that we make at most $6|p| + 3 = \frac{6}{k}n + 3$ parity calls on an execution of this algorithm. Thus a $(\frac{1}{2} - \frac{\alpha}{n})$-oracle with $\alpha < k/18$ would suffice. This can however be accomplished.

Why couldn't Chor et al. use the simpler, first technique? The reason lies in properties of the RSA function. The function $h_{a,b,p}(x) = ax + b \pmod p$ is multiplicative: Even if x is unknown we can still use the identity $h_{ac,bc,p}(x) \equiv ch_{a,b,p}(x) \pmod p$. We have already mentioned that the RSA function has similar properties. However the first method above uses *additive* properties of $h_{a,b,p}(x)$, namely $h_{a,b,p}(x) \pm c \equiv h_{a,b\pm c,p}(x) \pmod p$. The RSA function however, does not have such properties.

Using n^{-c}-oracles for lsb. To allow more erroneous oracles we use a modification of the techniques from [1], [3].

We first set out to describe the parity oracle. As mentioned, we can get a fairly reliable oracle for parity$(ax+b, p)$ using an oracle for lsb$(ax+b \pmod p)$. The flaw is that this does not work for arbitrary n^{-c}-oracles since we get the phenomenon of "error-doubling" (see [1],[3]) in asking for two points each time. The cure is in these two lemmas, which are slight modifications of those in [1], [3]. For the sake of self containment we sketch them as well as their proofs.

Lemma 10. *It is possible to generate (in polynomial time) a list, P, of $m \in poly(n)$ points of the form $rx + s$ and a set of lists, $L^{(i)}$, $i = 1, 2, \ldots 4m^3$, each $L^{(i)}$ containing m 0-1 elements. The lists satisfy:*

- *The points in P are pairwise independent and uniformly distributed in $\mathbf{Z}_p$.*
- *At least one of the lists $L^{(i)}$ satisfy: $lsb(P_j) = L_j^{(i)}$ for all but a $\frac{2}{\sqrt{m}}$ fraction of the j's.*

Since we now have points of the form $rx + s \pmod p$ with "known" lsb, we only need to ask the oracle about $\text{lsb}((a + r)x + (b + s) \pmod p)$ to deduce $\text{parity}(ax + b, p)$.

Proof. We go about generating these in the following way: Let $m \in poly(n)$ and divide $\mathbf{Z}_p$ into $m^{3/2}$ intervals of equal length, $I_i = [ip/m^{3/2}, (i + 1)p/m^{3/2})$. Select $r_1, r_2, s_1, s_2 \in_U \mathbf{Z}_p$ and let the j:th point, P_j, be $r_1x + s_1 + j(r_2x + s_2) \pmod p$, $j = 1, 2, \ldots, m$. It is easy to see that these points are uniformly distributed and pairwise independent.

Suppose we knew intervals I_{i_1}, I_{i_1} (of length $p/m^{3/2}$) such that $y = r_1x + s_1 \pmod p \in I_{i_1}$ and $z = r_2x + s_2 \pmod p \in I_{i_2}$. Assume that we in addition knew $\text{lsb}(y)$ and $\text{lsb}(z)$. Then, since y, z would now be known within $p/m^{3/2}$, $P_j = y + jz$, would be known within $(1 + j)p/m^{3/2} \leq 2p/\sqrt{m}$. Then, since $\lfloor P_j/p \rfloor$ would be determined by j, i_1, i_2, $\text{lsb}(P_j)$ would be determined by this and $\text{lsb}(y), \text{lsb}(z)$, unless P_j would happen to lie in an interval of length $2p/\sqrt{m}$ containing a multiple of p. But the later occurs only with probability $\frac{2}{\sqrt{m}}$.

Now, we do not actually "know" all the things assumed above, but there are m^3 possibilities for the pair (i_1, i_2) that determine the intervals and there are 4 choices for the pair $(\text{lsb}(y), \text{lsb}(z))$. All in all we have a set of $4m^3$ possibilities so we let $L^{(i)}$ consist of our calculated lsb's for each point, based on the i:th possibility for the quadruple $(i_1, i_2, \text{lsb}(x), \text{lsb}(y))$. Now, exactly one these quadruples is the correct one and hence, the corresponding list will contain m 0-1 elements, the j:th one being equal to $\text{lsb}(P_j)$ with probability $1 - \frac{2}{\sqrt{m}}$. □

We can now use these points to get a good parity oracle.

Lemma 11. *Let $\epsilon(n) = n^{-c}$ and let $\alpha > 0$ be any constant. Given an $\epsilon(n)$-oracle for $lsb(ax + b \pmod p)$ we can in polynomial time construct a set of $4m^3$ oracles, $m \in poly(n)$, such that at least one of them is a $\left(\frac{1}{2} - \frac{\alpha}{n}\right)$-oracle for $parity(ax + b, p)$ for randomly chosen a, x, b, p such that $ax + b$ is $(2\epsilon(n), p)$-small.*

Proof. Same as in [1], [3]. Each oracle gets the sample points P and the j:th oracle gets the list $L^{(j)}$ as created in Lemma 10. In order to get a vote for the parity of some $ax + b \pmod p$, the oracle uses a point $P_i = ux + v \pmod p$ in P and then only needs to ask the lsb-oracle about $\text{lsb}((a + u)x + (b + v) \pmod p)$. Recall that $\text{lsb}(ux + v \pmod p)$ is already "known" as $L_i^{(j)}$. The oracle that gets the correct choices for $L^{(j)}$ can be shown to be a $\left(\frac{1}{2} - \frac{\alpha}{n}\right)$-oracle for $\text{parity}(ax + b, p)$, provided $ax + b$ is $(2\epsilon(n), p)$-small.

The number of sample points, m, needed to make the majority decision reliable enough depends on $\epsilon(n), \alpha$ but can be shown to be polynomial in $n, \epsilon(n)^{-1}$, see [1], [3] for details. □

We can now prove the main theorem of this section:

Proof. (Of Theorem 8.) Assume that for some constant $c > 0$ we have a n^{-c}-oracle, O_l, for $\mathrm{lsb}(ax + b \pmod p)$ for randomly chosen p, a, b, x as in the formulation of the theorem. Choose uniformly at random a set of $4n^{2c+r}$ primes from $\mathcal{P}_k$: $p_1, p_2, \ldots, p_{4n^{2c+r}}$.

For each p_i, use Lemma 11 to convert the lsb-oracle into a set of $4m^3$ parity-oracles, $O_{i,1}, O_{i,2}, \ldots, O_{i,4m^3}$, one of them being a $(\frac{1}{2} - \frac{k_i}{2n})$-oracle for parity.

Next, using Lemma 9, for each p_i, use each $O_{i,j}$ to get a list of suggestions for $x \pmod{p_i}$: $z_{i,1}, z_{i,2}, \ldots, z_{i,4m^3}$.

Any set of k *correct* congruences $x \equiv z_{i,j} \pmod{p_i}$ for distinct p_i's will by the Chinese remainder theorem determine $x \pmod{2^n}$. Let X be the random variable that counts the number of i's for which the corresponding list $z_{i,1}, z_{i,2}, \ldots, z_{i,4m^3}$ *does* contain such a correct modular equation. By lemmas 9 and 11, we see that X is binomially distributed with expectance at least n^{c+r} and variance at most n^{2c+r}. By Chebyshevs inequality, we can bound the probability that $X < k$ from above by $O(n^{-r})$. Thus, the probability of retrieving the n-bit string x is at least $1 - n^{-r}$. The result can of course be checked by evaluating f and comparing to $f(x)$.

We do not know which of the parity oracles that is the good one for each p_i and hence neither which of the corresponding $z_{i,j}$'s to use. However, there are $O(n^{k(2c+r)})$ k-subsets in all and for each such there are $4^k m^{3k}$ ways to choose these $z_{i,j}$'s. We have a total of $O(n^{k(2c+r)} m^{3k}) \in \mathit{poly}(n)$ possibilities and each of them can be computed in polynomial time. The entire algorithm is therefore polynomial time. □

4.3 Security of other bits

The same reasoning as in [1], [3] also gives

Theorem 12. *Let a, b, p be chosen as in Theorem 8. Let c be any positive constant and let $j \leq c \log |p|$. Then: 1. The j:th least significant bit of $ax + b \pmod p$ is n^{-c}-secure. 2. The j least significant bits are simultaneously secure.*

To see 1, note that in the proof of Lemma 11 we can afford to try all possibilities for the j least significant bits. Also, in Lemma 9 we can "guess" the $j - 1$ least significant bits (or assume that they are $000 \cdots 0$) and start the algorithm from the j:th bit.

The simultaneous security follows from 1 and from Yao's unpredictability criterion, [7]: Predicting the j:th bit given the $j - 1$ first bits is equivalent to distinguishing the j least significant bits from random bits.

What about other bits? We first note that it is easy to see that the internal bits are secure with respect to a perfect oracle. Furthermore, we have recently been able to show the following result.

Theorem 13. *For "certain" primes $p \in \mathcal{P}_k$, for any $i = \alpha n/k$, $0 < \alpha < 1$, the i:th bit of $ax + b \pmod p$ for $a, b \in_U \mathbb{Z}_p$ is a family of hard-core predicates for any one-way function.*

The intuitive definition for "certain primes" is primes "sufficiently far from" a multiple of 2^i. At the present time some refinements will have to be made to make this set sufficiently dense among the set of all primes in $\mathcal{P}_k$. We will explore this further in the full paper.

5 Open problems

The proof of Theorem 8 does not hold if we choose $|p|$ too small. However, we ask if it would be possible to improve the result to allow for primes significantly shorter than $\Omega(n)$. For instance, is the theorem still true if $|p| \approx \sqrt{n}$?

The three common examples of UHF's often given in the literature are:

1. Multiplication by a randomly chosen boolean matrix.
2. Linear functions on GF[2^n].
3. Linear functions on $\mathbb{Z}_p$. (Almost universal.)

The results in [4] together with the above results show that all these give a logarithmic number of hard-core bits. The natural conjecture must be that *all* UHF's give hard-core bits. We note that both the proof in [4] as well as the proofs given here rely heavily on the explicit construction of the hash function. Some new technique seems called for to approach the general case.

6 Acknowledgements

I would like to thank my supervisor, Johan Håstad, for many enlightening discussions and for pointing out several errors in the early versions of this paper. Thanks to Christer Berg for further proofreading help.

References

1. W. Alexi, B. Chor, O. Goldreich & C. P. Schnorr: *RSA and Rabin Functions: Certain Parts Are as Hard As the Whole.* SIAM J. on Computing vol 17, no 2 1988, pp. 194–209.
2. J. L. Carter & M. N. Wegman: *Universal Classes of Hash Functions.* JCSS 18 1979, pp. 265–278.
3. B. Chor: *Two Issues in Public Key Cryptography.* An ACM distinguished Dissertation. MIT Press 1985.

4. O. Goldreich & L. A. Levin: *A Hard Core Predicate for any One Way Function.* STOC 1989, pp. 25–32.
5. J. Håstad, R. Impagliazzo, L. A. Levin & M. Luby: *Pseudo Random Number Generators from any One Way Function.* Manuscript 1993. Earlier versions appeared in STOC 1989, 1990.
6. U. V. Vazirani & V. V. Vazirani: *Efficient and Secure Pseudo-Random Number Generation.* FOCS 1984, pp. 458–463.
7. A. C. Yao: *Theory and Applications of Trapdoor Functions.* FOCS 1982, pp. 80–91.