All Bits in $ax + b \bmod p$ are Hard
(Extended Abstract)

Mats Näslund

Royal Institute of Technology,
Dept. of Numerical Analysis and Computing Science,
S–100 44 Stockholm, Sweden

E-mail address: matsn@nada.kth.se

Abstract. In this paper we show that for any one-way function f, being able to determine any single bit in $ax + b \bmod p$ for a random $\Omega(|x|)$-bit prime p and random a, b with probability only slightly better than 50% is equivalent to inverting $f(x)$.

1. Introduction

One of the most important questions in cryptography is the existence of so called *one-way functions* — functions easy to compute but hard to invert. It is natural to construct an encryption scheme from this primitive. Intuitively, finding the encrypted message would mean inverting the one-way function which in turn is presumed hard. One must be careful though, since even if the function is hard to invert, it may reveal partial information like: "The encrypted message starts with the text 'Fort Meade"', or less dramatic, "The message is an odd integer". This problem was noticed by Goldwasser and Micali, [5], who also showed how to circumvent it based on certain assumptions.

However, we would like the assumptions to be as weak as possible, preferable we would like to use *any* one-way function f. Although we can never hope that $f(x)$ itself will not leak any information about any bit in x, one could at least hope that some specific bits in x or in some simple function $h(x)$ would remain hidden, given $f(x)$ (or $f(x)$ and the description of h in the second case). We want bits that given $f(x)$ (and h) are unpredictable or "random looking" to any resource bounded adversary, resource bounded refering to computing time. Such bits/functions are called a *hard core* for f. This concept was studied by Blum and Micali in [3] and by Yao in [12]. For instance it was shown how to construct a *pseudo-random generator*, another important cryptographic primitive: Suppose the one-way function f is a permutation and that H is a set of boolean functions such that for random $h \in H$, $h(x)$ is unpredictable given $f(x), h$. Choose a random h, a random x_0 and output $h(x_0)$. Update: $x_{i+1} = f(x_i)$ and output $h(x_i)$, $i = 0, 1, \ldots, m$. Finally, output h. By the properties of f and H just discussed, one can hope that the sequence $h(x_0), h(x_1), \ldots, h(x_m), h$ will look random.

N. Koblitz (Ed.): Advances in Cryptology - CRYPTO '96, LNCS 1109, pp. 114-128, 1996.

Where should we look for hard core functions? For efficiency reasons it should be as easy as possible to compute $h(x)$. What first comes to mind is functions of the form: $h(x)$ = "some bit in x", for instance $h(x) = \mathrm{lsb}(x)$. There are examples of certain f for which $\mathrm{lsb}(x)$ looks random given $f(x)$. RSA-type functions has this property as shown by Alexi et al. in [**1**], Vazirani and Vazirani in [**11**] and Blum et al. in [**2**]. Similar results for discrete exponentiation in different settings appeared in [**3**] by Blum and Micali, in [**8**] by Long and Wigderson and in [**6**], Håstad et al. However, single bits in x may in general *not* be unpredictable. The simplest counterexample being $\mathrm{lsb}(x)$ with respect to $f(x) = g^x \bmod p$, discrete exponentiation modulo a prime (a conjectured one-way function). Here, computing $\mathrm{lsb}(x)$ is merely a matter of determining quadratic residuosity for $f(x)$. Since the most attractive situation is when our hard core bits are hard core for *any* one-way function f, we need more complex functions than single bits in x but which are still easy enough to compute. One soon also realizes that we can not hope that some fixed function will work for any f. However, we can hope that choosing a random function h in some fixed set of functions, H, will do the job. Other questions raised are: If $h \in H$ maps n bits to m bits, are all m bits in $h(x)$ individually hard core? Can certain subsets of the bits be used simultaneously?

The first general and efficient construction of a hard core function was done by Goldreich and Levin in [**4**]. They showed that any one-way function has a hard core bit obtained as the inner product modulo 2 of x and a random string r. Two more types of functions, (i) affine functions on a finite field of characteristic 2 and (ii) the analogue on the field of integers modulo a (not too short) prime p, were shown by Näslund in [**10**] to have the same property. In case (i), it was also shown that any single of the n bits in the function value is hard core for any one-way function. Here we extend the results from [**10**] to show that essentially all individual bits in the type-(ii) functions are also unpredictable.

In general, to show that some H is hard core for f, one shows that an algorithm, O, for predicting $h(x)$ for random $h \in H$ can be transformed into an inverting algorithm for f. Roughly speaking, O is used to determine the bits in x (or some simple, invertible function of x) *one by one*. Also, the bits are generally decided in order left to right or vice versa. The main technical novelty in this paper is a new method whereby the bits are determined *two by two*. The bits are not even adjacent, one of the two is the bit to the left of that predicted by O and the other is, at least in principle, the least significant bit. There are some cases where we during a short initial phase are only able to determine the first of the two bits. However, during this phase the latter bit has no importance and after a short delay we will be able to determine both the bits. We hope that the ideas behind this method will get further use or inspire others to develop methods applicable to show similar results for other types of functions.

The paper is organized as follows: After giving some notation in Section 2 and reviewing some previous work in Section 3, we give a general proof outline in Section 4. We then discuss the hardness of internal, but non leftmost bits in Section 5 and Section 6 extends the techniques to the leftmost bits. Due to space constraints, most proofs can only be sketched here.

2. Preliminaries

The model of computation used is that of probabilistic Turing machines running in time poly(n) where n is the length of the input, pptm's for short. In general, $|y|$ denotes the length of the binary string y. If S is a set, $|S|$ is the cardinality of S and by $x \in_U S$ we mean an x chosen uniformly at random from S.

We call a function $g(n)$ *negligible* if for every constant $c > 0$ and for every sufficiently large n, $g(n) < n^{-c}$. A *one-way function* is a poly-time computable function f such that for every pptm, M, the probability that $M(f(x)) \in f^{-1}(x)$ is negligible. The probability is taken over $x \in_U \{0,1\}^n$ and M's random coin flips. For simplicity all one-way functions in this paper are assumed to be *length-preserving*, i.e. $|f(x)| = |x|$. By simple padding arguments, it can be seen that this is no serious restriction.

Let H be an efficiently sampleable family of functions where each $h \in H$ is computable in deterministic polynomial time and maps $\{0,1\}^n \mapsto \{0,1\}^m$, $m \leq n$ and let f be a one-way function. An $\varepsilon(n)$-*oracle* for H is a pptm O such that $\Pr[O(f(x),h) = h(x)] \geq 2^{-m} + \varepsilon(n)$, the probability taken over $x \in_U \{0,1\}^n$, $h \in_U H$ and O's random choices. We call H a *hard core function for* f if no $\varepsilon(n)$-oracle exists for non-negligible $\varepsilon(n)$. (When $m = 1$ we have a hard core *predicate*.)

For $p \in \mathbb{Z}$ and $z \in \mathbb{Z}_p$, $\mathrm{bit}_i(z)$ denotes the ith bit of z. In particular $\mathrm{bit}_0(z) = \mathrm{lsb}(z)$. For $0 \leq i \leq j < |p|$, let $[z]_i^j$ denote bits $i, i+1, \ldots, j$ in the binary representation of z. Note: $[z]_i^i = \mathrm{bit}_i(z)$. If H is a family of functions, we will write $\mathrm{bit}_i(H)$ when referring to the family $\{\mathrm{bit}_i(h(x)) \mid h \in H\}$.

For $k > 0$, let $\mathcal{P}_k$ denote the set of primes of length n/k. Here, $n = |x|$, the security parameter of some one-way function f. The set of functions we study is

$$H_2^k = \{h(x) = ax + b \bmod p \mid p \in \mathcal{P}_k, a, b \in \mathbb{Z}_p\}.$$

Note that the probability space when choosing $h \in_U H_2^k$ is that of all *triples* (p,a,b) with $p \in_U \mathcal{P}_k$ and $a, b \in_U \mathbb{Z}_p$. However, assuming the existence of an $\varepsilon(n)$-oracle for $\mathrm{bit}_i(H_2^k)$, simple counting arguments allow us to study a set of ps of density $\varepsilon(n)/2$ in $\mathcal{P}_k$ for which our oracle will be successful with probability $1/2 + \varepsilon(n)/2$ taken over *pairs* (a,b). When we have such a p and only vary a, b, we indicate this by denoting the oracle an $(\varepsilon(n)/2, p)$-oracle.

For a given p, the bits in $ax + b \bmod p$ are not uniformly distributed. By the *bias* of the ith bit we mean the value β_i such that $\Pr_{a,b}[\mathrm{bit}_i(ax + b \bmod p) = 0] = \frac{1}{2} + \beta_i$ when a, b are chosen uniformly at random from $\mathbb{Z}_p$. It is easy to see that $\beta_0 = \frac{1}{2p}$ and that only the msb can have bias greater than $1/6$. The bias is significant only for the $O(\log n)$ most significant (leftmost) bits so we divide the paper into two main parts: The internal (non leftmost) and the leftmost bits.

3. Previous Work

In [10] it was shown that an n^{-c}-oracle for $\mathrm{lsb}(H_2^k)$ could be used to retrieve $x \bmod p$ with good probability. Repeating this for different p, Chinese remaindering was applied to find x.

To find $x \bmod p$ basically the following method was used: Assume the existence of a "very good" oracle for the lsb. Having determined the lsb of $ax + b \bmod p$ we zero

it by subtracting: $b' = b - \text{lsb}$. Computing $2^{-1}(ax+b') \bmod p$ will then give us the rest of the bits in $ax+b \bmod p$ shifted one step to the right. Continue this process to extract all the bits in $ax+b \bmod p$. With a, b, p known, x is easily found.

The very good oracle in turn was constructed using the two point based sampling technique from [1] and a majority decision. This could be done if $ax+b \bmod p$ was "small" (zeros in the msbs) to avoid wraparound modulo p (i.e. reduction by p) when adding sample points. Actually a polynomial number of oracles had to be constructed with at least one of them being good. All the oracles were then tried. It was also noted that the methods used applies for $\text{bit}_i(H_2^k)$ with $i \in O(\log n)$, since the bits to the right of the oracle could be put to zero with non-negligible probability.

4. Proof Outline

The method summarized above does not work with $\Omega(n)$ bits to the right initially undetermined. Suppose we have an oracle for $\text{bit}_i(H_2^k)$, $i \in \Omega(n)$. Here we decide the bits two by two; bit $i+1$ and another bit. In principle, we would like the latter bit to be the lsb, but as will be seen this is sometimes too optimistic. There will sometimes be a short initial phase during which we can only determine bit $i+1$, but on the other hand, the lsb can be shown to be unimportant during this phase. After this phase, the lsb (actually what *was* the lsb some iterations ago) will start to matter but this will also imply that we can start to determine it. To begin with, let us for simplicity ignore any possible problem and for a second let us assume that we *can* determine bit $i+1$ and the lsb. Having determined them, they are set to zero by subtracting: $b' = b - 2^{i+1}\text{bit}_{i+1} - \text{lsb}$. Next, $ax+b$ can be right-shifted: $2^{-1}(ax+b') \bmod p$. No non-zero bit will pass position i and since the lsb is always zero before the shift, no wraparound is caused. We can now proceed with the next two bits.

Finding bit $i+1$, lsb is now done as follows: Assume that we can make sure that for some $m', d \in O(\log n)$, we have $[ax+b]_{i-d}^{i} = 0$ and $[ax+b]_{n/k-m'}^{n/k-1} = 0$ i.e. $ax+b \bmod p$ has zeros to the right of bit i and in the msbs. Furthermore, suppose we have a random value of the form $rx+s \bmod p$ for which $[rx+s]_{i-d}^{i} = z$ and $[rx+s]_{n/k-m'}^{n/k-1} = w$ are both known. Then, with high probability (if no wraparound occurs), $[ax+b+rx+s]_{i-d}^{i} \approx z$ too. (More precisely, it will be z or $z+1$ depending on if a carry propagates from less significant bits into the bit-window we are studying.) What would happen if we attempt to shift $ax+b$ to the right mod p, i.e. ask the oracle about $2^{-1}(ax+b)+rx+s \bmod p$?

Case 1, $\text{lsb}(ax+b \bmod p) = 0$**:** We indeed shift $ax+b$ to the right, so the bits to the right of bit i in $2^{-1}(ax+b) \bmod p$ will still be all zeros. The important thing that happens is that $\text{bit}_{i+1}(ax+b \bmod p)$ is shifted into position i. This bit has weight 2^d among the bits $i, i-1, \ldots, i-d$.

Case 2, $\text{lsb}(ax+b \bmod p) = 1$**:** Bit $i+1$ is still moved into position i, but in addition we get wraparound. More precisely we will get

$$2^{-1}(ax+b) \equiv 2^{-1}(ax+b-1) + 2^{-1} \equiv 2^{-1}\underbrace{(ax+b-1)}_{\text{lsb}=0} + \frac{p+1}{2} \pmod{p}. \qquad (1)$$

Now, assume the prime p looks like $p = p_1 2^{i+2} + \frac{p_2}{2^{d+1}} 2^{i+2} + p_3$ where $p_1, p_2, p_3 \in \mathbb{Z}$, $p_2 < 2^{d+1}$ and p_3 is "small". We get

$$\frac{p+1}{2} = \frac{p_1 2^{i+2} + \frac{p_2}{2^{d+1}} 2^{i+2} + p_3 + 1}{2} = p_1 2^{i+1} + p_2 2^{i-d} + \frac{p_3+1}{2}. \tag{2}$$

Since p_3 is small, the term $(p_3+1)/2$ will only affect the rightmost bits (with high probability). The term $p_1 2^{i+1}$ will only affect bits to the left of the oracle, assuming no extra wraparound occurs when the msbs in p are added to the msbs in $rx+s$. Now, since we know the value w above, we know if wraparound occurs in which case we subtract $(p-1)/2$ in a similar way. So it would seem that wraparound is no serious concern but in fact this is what sometimes prevents us from determining the lsb. Nevertheless, let us again ignore this and push the problems ahead somewhat.

Thus the important difference in this case is that we by (1) and (2) also add p_2 to the value in bits $i, i-1, \ldots, i-d$. Cases 1, 2 can hence be summarized as: If bits $[ax+b]_{i-d}^{i} = 0$, $[ax+b]_{n/k-m'}^{n/k-1} = 0$ and $[rx+s]_{i-d}^{i} = z$, then with high probability, i.e. unless unforseen wraparound occurs, we have

$$\begin{aligned} z' &= [2^{-1}(ax+b) + rx + s]_{i-d}^{i} \approx [2^{-1}(ax+b)]_{i-d}^{i} + [rx+s]_{i-d}^{i} \\ &= z + 2^d \,\mathrm{bit}_{i+1}(ax+b \bmod p) + p_2 \,\mathrm{lsb}(ax+b \bmod p). \end{aligned} \tag{3}$$

Knowing z and determining z' thus gives some information about the two unknown bits in $ax+b$. We immediately see that we can not allow $|p_2 - 2^d|$ to be small since the effect of $\mathrm{bit}_{i+1}(ax+b \bmod p) = 1$ would be the same as if $\mathrm{lsb}(ax+b \bmod p) = 1$. We will later make precise what should be demanded from p_2 (or actually from $p_2/2^{d+1}$).

Section 5.2 gives a generalization of the two point based sampling from [**1**] to get values of the type $rx+s \bmod p$ with known z-values. How do we determine the z'-value *after* the shift?

Observation 4.1. Let J, J' be equal-length subintervals to $\{0, 1, \ldots, 2^{d+1}-1\}$, at distance $D = 2^d \,\mathrm{bit}_{i+1}(ax+b \bmod p) + p_2 \,\mathrm{lsb}(ax+b \bmod p)$ (i.e. their left endpoints are D apart, modulo 2^{d+1} and we write $J' = J + D$). Then by (3), with high probability:

$$[rx+s]_{i-d}^{i} \in J \Leftrightarrow [2^{-1}(ax+b) + rx + s]_{i-d}^{i} \in J + D = J'.$$

If the fraction of 1-answers the oracle gives on J differs non-negligible from that on J', we can distinguish this by sampling. Thus, we need four interval pairs: $(J_v, J_v + D_v)$, $v = 1,2,3$, for the three D_v-values $p_2, 2^d$ and $2^d + p_2$ and one, $(J_4, J_4 + D_4)$, for the exclusive or of bit_{i+1} and the lsb: $D_4 = 2^d - p_2$. (We will later need "two-dimensional" intervals to take care of the wraparound problem. Nevertheless the intuition is the same and the above notation simplifies the layout of the ideas.)

We will in Section 5.1, 5.3 discuss the properties needed from the prime p and use this in Section 5.4 to show existence of, and how to find the intervals.

With the general idea described, let us return to the wraparound problem. A close look will show that depending on whether $rx+s \bmod p \geq p/2$ or not and whether $\mathrm{lsb}(ax+b)$ is zero or one, we always have one of two possibilities for the value we query the oracle on: (1) It is smaller than $p/2$ and lies in some interval J, or: (2) It is greater than $p/2$ and lies in the interval $J + p_2$. Hence, we can not argue that we have

random points in J and $J + p_2$. For instance the oracle might be correct only on $y \in \mathbb{Z}_p$ such that $\mathrm{bit}_i(y) = \mathrm{bit}_i(y + \lfloor p/2 \rfloor)$ and otherwise behave randomly.

All is not lost though, since this means that the lsb-value has virtually no effect on the oracle's behaviour and the information we get is precisely the value of the $(i+1)$th bit. We therefore simply neglect the lsb, determine bit $i+1$ and zero it and shift $ax+b$ to the right. We don't know the lsb that went out to the right, but by assumption, the oracle behaves independently of that bit. What happens on the next attempted shift? We now add roughly $2^d \mathrm{bit}_{i+2}(ax+b) + p_2 \mathrm{bit}_1(ax+b) + [\lfloor p/4 \rfloor]_{i-d}^{i} \mathrm{lsb}(ax+b)$ to the value in the sample point in Equation (3). (To be correct, $\lfloor p/4 \rfloor$ should be replaced by $2^{-2} \bmod p$, but these numbers are basically the same.) We can ignore the term $p_2 \mathrm{bit}_1(ax+b)$ since it (by assumption) has no effect on the oracle. Now, either the term $[\lfloor p/4 \rfloor]_{i-d}^{i} \mathrm{lsb}(ax+b)$ influences the oracle or it doesn't. If it does, we can now determine the lsb delayed by one step and in the future, all bits to the left of it delayed one step too. On the other hand, if not even $[\lfloor p/4 \rfloor]_{i-d}^{i} \mathrm{lsb}(ax+b)$ has any effect, we can ignore this term too and just determine bit $i+2$.

If this should go on, we will get more and more "spooky" bits that moves out to the right but as long as their effect is negligible, we can determine the bit to the left of the oracle in each step. But this can not go on for ever! We claim that after a small $O(\log n)$ delay during which the spooky bits are of no importance, we can start to determine them: If we after τ shifts still have no non-negligible advantage in determining the lsb, this means that most of our oracle's advantage is for $y \in \mathbb{Z}_p$ such that most of

$$\mathrm{bit}_i(y), \mathrm{bit}_i(y + \lfloor p/2^\tau \rfloor), \mathrm{bit}_i(y + \lfloor 2p/2^\tau \rfloor), \cdots, \mathrm{bit}_i(y + \lfloor (2^\tau - 1)p/2^\tau \rfloor)$$

are equal. (Actually, the values $\{\lfloor jp/2^\tau \rfloor \mid j = 1, \ldots, 2^\tau - 1\}$ is a permutation of values close to $\{j2^{-\tau} \bmod p \mid j = 1, \ldots, 2^\tau - 1\}$, which is what we actually want to study.) Intuitively, if no small integer multiple of $\lfloor p/2^\tau \rfloor$ is close to any multiple 2^i, this set of ys must be very small, contradicting our hypothesis on the oracle! The value τ above will be called a *good shift*.

5. Security of Non Leftmost Bits

Assume we have an n^{-c}-oracle for $\mathrm{bit}_i(H_2^k)$ where $(26c+26)\lceil \log n \rceil \le i \le n/k - c_c' \log n$ with c_c' the maximum of $7c+8$ and the smallest constant such that the bias of bit $n/k - c_c' \log n$ is at most $n^{-c}/4$. This choice of c_c' will be explained later. The case $0 \le i < (26c+26)\lceil \log n \rceil$ is covered by [**10**].

5.1. Some Results on Uniform Distribution of Sequences. For $\alpha \in \mathbb{Q}$, $\{\alpha\}$ denotes the fractional part, $\alpha \pmod 1$. By a *rational sequence* we mean a sequence of the form $\{\alpha\}, \{2\alpha\}, \ldots, \{N\alpha\}$ where $\alpha \in \mathbb{Q}$, $N \in \mathbb{N}$. We denote such a sequence by $(\alpha)_N$.

Definition 5.1. The rational number α, $0 \le \alpha < 1$ is said to be of (Q, ψ)-*type* if for all integers t, q, $0 \le t < q \le Q$:

$$\left| \alpha - \frac{t}{q} \right| > \frac{1}{q^2 \psi}.$$

(That is, the distance from $q\alpha$ to the nearest integer is at least $\frac{1}{q\psi}$.)

Lemma 5.1. Let $s \in \mathbb{Z}$ be given and let $s \geq Q^2\psi$. Then

$$\Pr_{r \in_U \mathbb{Z}_s}\left[\frac{r}{s} \text{ is of } (Q,\psi)-\text{type}\right] \geq 1 - \frac{9\log^2 Q}{\psi}.$$

Proof: Given s, look at a particular pair of integers t,q, $t<q$. The values of r that are bad for this t,q are the ones satisfying $|r/s - t/q| \leq 1/(q^2\psi)$ which is equivalent to r being chosen in the interval $\left[\frac{s}{q}\left(t-\frac{1}{q\psi}\right), \frac{s}{q}\left(t+\frac{1}{q\psi}\right)\right]$. This has probability at most $2(q^2\psi)^{-1} + s^{-1}$ since the interval contains at most $2s(q^2\psi)^{-1}+1$ integers. Hence

$$\begin{aligned}\Pr[r \text{ is bad}] &\leq \sum_{0\leq t<q\leq Q}\left(\frac{2}{q^2\psi}+\frac{1}{s}\right) \leq \frac{Q^2}{s}+\frac{2}{\psi}\sum_{1\leq t,q\leq Q}\frac{1}{tq} \\ &= \frac{Q^2}{s}+\frac{2}{\psi}\left(\sum_{1\leq q\leq Q}\frac{1}{q}\right)^2 \leq \frac{Q^2}{s}+\frac{8}{\psi}\log^2 Q \leq \frac{9}{\psi}\log^2 Q.\end{aligned}$$

□

Definition 5.2. Let $(\alpha)_N$ be a rational sequence and let $J=[a,b)$ be a subinterval of $[0,1)$ (allowing wrap around 1 by $1 \equiv 0 \pmod 1$). Define

$$A((\alpha)_N, J) = |(\alpha)_N \cap J|,$$

i.e. the number of points in the sequence falling into J.

The *discrepancy* (deviation from uniform distribution) of $(\alpha)_N$ is defined by

$$\mathcal{D}((\alpha)_N) = \sup_{0\leq a<b\leq 1}\left|\frac{A((\alpha)_N,[a,b))}{N} - (b-a)\right|.$$

We will of course not actually deal with rational numbers, but rather integers. However, it is easy to see that the sequence $ir \bmod s$, $i=1,\ldots,N$, will be as well distributed in $\mathbb{Z}_s$ as the sequence $(r/s)_N$ is in $[0,1)$. Well known results state that if α is irrational then $\mathcal{D}((\alpha)_N) \to 0$ as $N \to \infty$, see [7]. We will later need a *quantitative* result for *rational* α.

5.2. Generalizing the Two-point Based Sampling. As mentioned in the proof outline, we will need a set, P, of random sample points of the form $rx+s \bmod p$ with bits $i, i-1, \ldots, i-O(\log n)$ "known" beforehand, i.e. listed in a set L. To control wraparound, their $O(\log n)$ most significant bits must also be "known" as the values in a list M. To this end we will use a generalization from [**10**] of the two point based sampling technique introduced in [**1**]. We will get a polynomial number of candidates, $\{(L,M)\}$ one (L,M)-pair being correct with high probability.

Lemma 5.2. Let $d_1, d_2, t > 0$ be constants, let $m \in \text{poly}(n)$ and let $r_1, s_1, r_2, s_2 \in_U \mathbb{Z}_p$ be given. It is then in deterministic polynomial time possible to generate a list $P^{(r_1,s_1,r_2,s_2)}$, of tm uniformly distributed, pairwise independent points of the form $rx+s \bmod p$ and at most $t^2m^5 2^{2(d_1+d_2)\lceil\log n\rceil}$ possibilities for a pair of lists $\{(L,M)\}$, each L consisting of tm values in $\{0,1\}^{(d_1+d_2)\lceil\log n\rceil}$ and each M of tm values in $\{0,1\}^{3/2\log m}$.

For random r_1, s_1, r_2, s_2, for at least one $(L^0, M^0) \in \{(L, M)\}$, for every $j = 1, \ldots, tm$ the following two conditions hold, each with probability at least $1 - \frac{2}{tm^{3/2}}$:

$$[P_j^{(r_1,s_1,r_2,s_2)}]_{i-(d_1+d_2)\lceil \log n \rceil}^{i} = L_j^0 \quad \text{and} \quad [P_j^{(r_1,s_1,r_2,s_2)}]_{n/k-3/2\log m}^{n/k-1} = M_j^0$$

Proof: Straight forward generalization of Lemma 10 in [**10**]. □

5.3. Good Primes. Using the results in the previous subsections, now define the set of primes to study from now on.

Definition 5.3. Let d_1, d_2 be constants[1] satisfying $d_2 > 23c + 23$ and $c + 4 < d_1 \leq 3c + 3$. Let $l(n) = 2^{(d_1+d_2)\lceil \log n \rceil + 1}$, $\psi(n) = 2^{d_1 \lceil \log n \rceil}$ and $N(n) = 5n^{7c+7}$ (Note: $l(n) \geq 4N(n)^2\psi(n)$ for sufficiently large n). Finally, let $\tau(n) = \lceil \log N(n) \rceil$. Assume $p \in \mathcal{P}_k$, $k > 0$ and that p is of the form

$$p = p_1 2^{i+\tau(n)+2} + \frac{p_2}{l(n)} 2^{i+2} + p_3$$

where $p_1, p_2, p_3 \in \mathbb{Z}$, $p_1 \geq 0$, $1 \leq p_3 < \frac{2^{i+2}}{l(n)}$, $0 \leq p_2 < 2^{\tau(n)} l(n)$ and where

$$p_2^{(j)} = [p_2]_{j-1}^{j-1+(d_1+d_2)\lceil \log n \rceil + 1}$$

is such that $\frac{p_2^{(j)}}{l(n)}$ is of $(2N(n), \psi(n))$-type for $j = 1, 2, \ldots, \tau(n)$.

Then p is called a *good prime*. Call this set of good primes $\mathcal{P}_k'$ and let H_3^k be as the set H_2^k, but with p restricted to $\mathcal{P}_k'$.

Note that $p_2^{(j)}$ corresponds to bits $i, i-1, \ldots, i-(d_1+d_2)\lceil \log n \rceil$ in $\lfloor p/2^j \rfloor$ which is $\approx 2^{-j}$ mod p and also that by the choice of c_c' earlier, we always have $\tau(n)$ bit to the left of bit i. By only studying these bits, we will introduce a truncation error, but by choosing d_2 sufficiently large, this error can be controlled.

Lemma 5.3. The probability that a randomly chosen $p \in \mathcal{P}_k$ is good is $1 - O(n^{-(d_1-3)})$.

Proof: By Definition 5.3 and Lemma 5.1, the probability that an n/k bit number is bad (regardless of primality) is at most $9\tau(n)\log^2(2N(n))/\psi(n) \in O(2^{-(d_1-2)\lceil \log n \rceil})$ (Sum up the probabilities that each $p_2^{(j)}$s is not of $(2N(n)^2, \psi(n))$-type.) It is well known that the number of primes of length n/k is $\theta\left(\frac{k}{n} 2^{n/k}\right)$ and so

$$\Pr[p \text{ is a bad prime }] \leq \Pr[p \text{ is bad }] / \Pr[p \text{ prime }] \in O(n^{-(d_1-3)}).$$

□

5.4. Finding Good Intervals and Shifts. Let I be the set of all possible values for the bits $i, i-1, \ldots, i-(d_1+d_2)\lceil \log n \rceil$ in a value modulo p. That is, $I = \{0, 1, \ldots, l(n)-1\}$. Let $Y = \{\lfloor z/(p/2^{\tau(n)}) \rfloor \mid z \in \mathbb{Z}_p\}$, i.e. a partition of $\mathbb{Z}_p$ into (roughly) equal sized sets.

Define projections from $\mathbb{Z}_p$ onto I and Y respectively: $\pi_I(z) = [z]_{i-(d_1+d_2)\lceil \log n \rceil}^{i}$ and $\pi_Y(z) = \lfloor z/(p/2^{\tau(n)}) \rfloor$ and let $\pi(z)$ be the pair $(\pi_I(z), \pi_Y(z))$. Observe that $\pi(2^i) = (l(n)/2, 0)$ and $\pi(\lfloor p/2^j \rfloor) = (p_2^{(j)}, 2^{\tau(n)-j})$ for $1 \leq j \leq \tau(n)$.

[1] Please note that this is a preliminary version and that some of the constants may possibly be improved and others may have to be increased.

We will by the word *interval* refer to a pair $J = (I', y)$ where $y \in Y$ and I' is a subset of consecutive values (points) in I, allowing wrap around the endpoints of I. The length of J, $|J|$, is the length of I'.

For any interval $J = (I', y)$, $J + \pi(z)$ refers to the interval $(I' + \pi_I(z), y + \pi_Y(z))$ with the addition carried out componentwise modulo $|I|$ and $|Y|$ respectively. By $P_1(J)$ we mean the average fraction of 1-answers the oracle gives on J. For two intervals J_1, J_2, denote by $\Delta(J_1, J_2)$ the value $|P_1(J_1) - P_1(J_2)|$.

Definition 5.4. Let $\tau_0 \in O(\log n)$ be the smallest integer (if it exists) such that there is an interval J of length at least $2^{d_2\lceil \log n \rceil - 2}$ and with $\Delta(J, J + \pi(\lfloor p/2^{\tau_0} \rfloor))$ non-negligible. Then τ_0 is called a *good shift*.

The good shift is the number of steps we may have to wait before we can start to determine (what was) the lsb. It is important that we choose the smallest possible τ_0 since we must be certain that the information we get for the $\tau_0 - 1$ first shifts is precisely the $(i+1)$th bit. This holds since the effect of the spooky bits is negligible for these first shifts. It is perhaps not obvious that a good shift exists or if it does, that we can keep on determining the $(i+1)$th bit from the τ_0th shift and on but we will shortly prove that this is the case.

To begin with, we have the following immediate consequence to our assumptions:

Lemma 5.4. If O is an $(n^{-c}/2, p)$-oracle for $\text{bit}_i(H_3^k)$ and the bias of bit_i is β_i, then for $h(x) = ax + b \bmod p$, $a, b \in_U \mathbb{Z}_p$:

$$\Pr[O(h, f(x)) = 1 \,|\, \text{bit}_i(h(x)) = 1] - \Pr[O(h, f(x)) = 1 \,|\, \text{bit}_i(h(x)) = 0] \geq 2\left(\frac{n^{-c}}{2} - \beta_i\right).$$

Put differently, and using our assumptions on the bias induced by the choice of the value c_c' earlier: There is an interval J of length $l(n)/2$ satisfying $\Delta(J, J + \pi(2^i)) \geq n^{-c}/2$.

We omit the (simple) proof. The next lemma will show the existence of other, non-trivial, subintervals for which the fraction of 1-answers from the oracle also differ non-negligible.

Lemma 5.5. Let $p \in \mathcal{P}_k'$. There is an interval J in $I \times Y$ of length at least $2^{d_2\lceil \log n \rceil - 2}$ with

$$\Delta(J, J + \pi(\lfloor p/2^{\tau(n)} \rfloor)) \geq \gamma(n)$$

where $\gamma(n)$ is non-negligible.

Proof: See the appendix for a proof sketch. □

It is not too hard to see that adding 2^i does not affect numbers of $(2N(n), \psi(n))$-type. So, together with Lemma 5.4, we have the following corollary:

Corollary 5.6. There are intervals J_1, J_2, J_3, J_4 of length at least $2^{d_2\lceil \log n \rceil - 2}$ and a value $\tau_0 \leq \tau(n)$ such that

$$\Delta(J_1, J_1 + \pi(\lfloor p/2^{\tau_0} \rfloor)), \Delta(J_2, J_2 + \pi(2^i)), \Delta(J_3, J_3 + \pi(2^i + \lfloor p/2^{\tau_0} \rfloor)),$$
$$\Delta(J_4, J_4 + \pi(2^i - \lfloor p/2^{\tau_0} \rfloor))$$

are all at least $\gamma(n)$. In particular, every $(n^{-c}/2, p)$-oracle for $\text{bit}_i(H_3^k)$ has a good shift.

Since none of the $p_2^{(j)}$s can be close to $\pi(2^i)$, there will be no trouble retrieving bit_{i+1} for the $\tau_0 - 1$ first shifts.

Now that good intervals/shifts do exist, we will need to be able to find them.

Lemma 5.7. Given is $T \in \mathrm{poly}(n)$ subintervals to $I \times Y$: $J_1, J_2, \ldots, J_T$, each of length $\lambda(n) \geq 2^{d_2\lceil \log n \rceil - 2}$. Then for any constants $s, t > 0$, it is possible to find a polynomial number of lists, $\{R^{(j)} \mid j = 1,2,\ldots\}$, each $R^{(j)}$ indexed by the J_ls such that for at least one j, for each $l = 1,2,\ldots,T$:

$$\Pr[|R_{J_l}^{(j)} - P_1(J_l)| \leq \gamma(n)/t] \geq 1 - n^{-s}.$$

Proof sketch: Use Lemma 5.2 to generate random points in $\mathbb{Z}_p$ for which we "know" the intervals they belong to and query the oracle on these. Roughly $\gamma(n)^{-2} n^{2s} l(n) 2^{\tau(n)}/\lambda(n)$ sample points are needed. □

For the remainder of the paper, we will assume that the good shift is equal to one, i.e. that we can distinguish between values of the form z and $z + \lfloor p/2 \rfloor$. Equivalently, we can assume that we have already shifted $ax + b$ by $\tau_0 - 1$ steps to the right, determining and zeroed bit $i+1$ at each step, and from now on want to determine both bit $i+1$ and the lsb (or rather, what was the lsb $\tau_0 - 1$ iterations ago).

5.5. Constructing a Good $(\mathrm{bit}_{i+1}, \mathrm{lsb})$**-oracle.** Suppose the bits $[ax+b]_{i-(d_1+d_2)\lceil \log n \rceil}^{i}$ and $[ax+b]_{n/k-3/2\log m}^{n/k-1}$ all are zero and that we have sample points $P = \{rx + s \bmod p\}$ from Lemma 5.2 with the value in the same bitpositions "known". By substituting $d = (d_1 + d_2)\lceil \log n \rceil$ (so that $2^d = l(n)/2$) in Observation 4.1 we have

Observation 5.1. Let J_1, J_2, J_3, J_4 be the intervals from Corollary 5.6, let b_{i+1}, b_0 be the two (unknown) bits $\mathrm{bit}_{i+1}(ax + b \bmod p), \mathrm{lsb}(ax + b \bmod p)$ respectively and let $\tilde{b}_{i+1}, \tilde{b}_0$ be a current hypothesis for b_{i+1}, b_0. Then, if $\pi(rx + s \bmod p) = (z, y)$, we have (assuming no wraparound from the msbs):

$$\begin{aligned}(z', y') &= \pi(2^{-1}(ax+b) + (rx+s) \bmod p) \approx (z,y) + \pi(2^i b_{i+1} + \lfloor p/2 \rfloor b_0) \\ &= (z + \frac{l(n)}{2} b_{i+1} + p_2^{(1)} b_0, y + b_0 2^{\tau(n)-1})\end{aligned}$$

and therefore, for $v = 1,2,3,4$, with high probability:

$$(z,y) - (\frac{l(n)}{2}\tilde{b}_{i+1} + p_2^{(1)}\tilde{b}_0, 0) \in J_v$$

$$\Updownarrow$$

$$(z', y') \in J_v + (\frac{l(n)}{2}(b_{i+1} - \tilde{b}_{i+1}) + p_2^{(1)}(b_0 - \tilde{b}_0), b_0 2^{\tau(n)-1}).$$

Observe that by substituting suitable values in $\{0,1\}$ for $b_{i+1}, b_0, \tilde{b}_{i+1}, \tilde{b}_0$, we recognize the four good interval pairs $J_v, J_v + \pi(D_v)$ from Corollary 5.6.

For two equal-sized sets (intervals) $J' = (I', y'), J'' = (I'', y'')$ define $\#(J', J'') = 2(|I'| - |I' \cap I''|)$ if $y' = y''$ and $|I'|$ otherwise. (The size of the symmetric difference.) The next lemma tells us how to distinguish between the two intervals in each pair, or rather, how to *exclude* one of the two as a possibility.

Lemma 5.8. Let $J_v, J_v + \pi(D_v)$ be any of the interval pairs from Corollary 5.6 and let $R = \{R_J\}$ be approximations to the respective $P_1(J)$ within $\gamma(n)/4$. Let S be a set of $16tn\gamma(n)^{-2}$ uniformly distributed, pairwise independent samples from *any* interval J' of the same length as J_v. Define boolean variable $B(S) = 0$ if $\#(J_v, J') \in O(1)$ and $B(S) = 1$ if $\#(J_v + \pi(D_v), J') \in O(1)$. Let $B(S)$ be undefined otherwise. Finally, define $B_v^R(S) = 0$ if the number of 1-answers the oracle gives on J' is closer to R_{J_v} than to $R_{J_v+\pi(D_v)}$ and 1 otherwise. Then

$$\Pr[B_v^R(S) \neq B(S) \mid B(S) \text{ is defined }] \leq \frac{1}{tn}.$$

The proof is a fairly simple application of Chebyshev's inequality and is therefore left out. The idea is that if we conclude that $B_v^R(S) = 1$ so that we believe the samples to be from $J_v + \pi(D_v)$, then with high probability, the hypothesis $\tilde{b}_{i+1}, \tilde{b}_0$ as in Observation 5.1 must be wrong — we would otherwise have sampled in J_v and should have $B_v^R(S) = 0$. Conversely, if $B_v^R(S) = 0$, then $(\text{bit}_{i+1}, \text{lsb})$ is very unlikely to have a value (b_{i+1}, b_0) such that $(\frac{l(n)}{2}(b_{i+1} - \tilde{b}_{i+1}) + p_2^{(1)}(b_0 - \tilde{b}_0), b_0 2^{\tau(n)-1}) = \pi(D_v)$, since if this would be the case, we would sample in $J_v + \pi(D_v)$ and should have $B_v^R(S) = 1$.

Lemma 5.9. Given an $(\frac{n^{-c}}{2}, p)$-oracle for $\text{bit}_i(H_3^k)$ with $(26c + 26)\lceil \log n \rceil \leq i \leq n/k - c_c' \log n$, and assuming that $[ax + b]_{i-(d_1+d_2)\lceil \log n \rceil}^{i}$ and $[ax + b]_{n/k-3/2\log m}^{n/k}$ are all zero, then for any constant $t > 0$, it is possible to construct a polynomial number of oracles, $O_1, O_2, \ldots$, such that with high probability, at least one O_j is a $(\frac{3}{4} - \frac{1}{tn}, p)$-oracle for bit_{i+1} *and* the lsb in H_3^k. The construction can be made in polynomial time.

Proof: Suppose we have: (1) the intervals J_1, J_2, J_3, J_4 from Corollary 5.6, (2) the approximations, R, to all $P_1(J_v), P_1(J_v + \pi(D_v))$ within $\gamma(n)/4$ from Lemma 5.7 and (3) a set P of $3m = 240tn\gamma(n)^{-2}$ sample points from Lemma 5.2 with correct values for the "guessed" bits. We actually have a polynomial number of suggestions for (1),(2),(3), but trying all, making one oracle for each, we can assume we have the correct one. In each of three runs we will compute a particular B_v^R as described in Lemma 5.8, each run will attempt to rule out one of the remaining possibilities for the pair $\text{bit}_{i+1}, \text{lsb}$ in $ax + b \bmod p$.

Consider one of the three runs, trying to compute $B_v^R(S)$ with $J_v = (J', y)$ for some $v \in \{1, 2, 3, 4\}$ and having $\tilde{b} = (\tilde{b}_{i+1}, \tilde{b}_0)$ as a current hypothesis for $(\text{bit}_{i+1}, \text{lsb})$. Use the following sample points for this particular $v, \tilde{b}$: Let $\{P_j = r_j x + s_j \bmod p\}$ be the first $m = 80tn\gamma(n)^{-2}$ sample points in P and define $r'(\tilde{b}) = \frac{l(n)}{2}\tilde{b}_{i+1} + p_2^{(1)}\tilde{b}_0$. Let $r_1^v, r_2^v, \ldots r_m^v$ be points chosen independently and uniformly in $\{0,1\}^{(d_1+d_2)\lceil \log n \rceil}$ to make $r_j^v - r'(\tilde{b}) + [P_j]_{i-(d_1+d_2)\lceil \log n \rceil}^{i}$ uniformly distributed in J'. This can be done since $[P_j]_{i-(d_1+d_2)\lceil \log n \rceil}^{i}$ and $\tilde{b}$ are known. Now use the following sample points:

$$S = S_v(\tilde{b}) = \{2^{-1}(ax + b) + P_j + (r_j^v - r'(\tilde{b}))2^{i-(d_1+d_2)\lceil \log n \rceil} \bmod p \mid j = 1, \ldots, m\}.$$

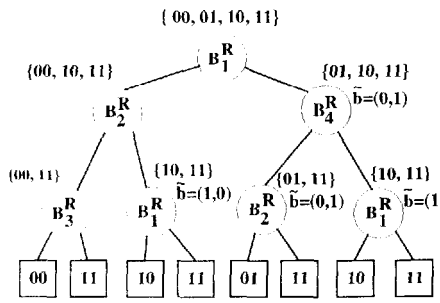
We can describe the three runs with the decision tree in the figure below. (We assume no wraparound from the msbs occur.)

Internal vertices are labeled with the variable B_v^R on which we base our decision at that vertex. We use $\tilde{b} = (0,0)$ unless indicated. We follow a left edge if $B_v^R(S_v(\tilde{b})) = 0$

and a right if $B_v^R(S_v(\tilde{b})) = 1$. The sets in braces indicate the remaining possible values for the pair $(\mathrm{bit}_{i+1}, \mathrm{lsb})$ and the leaves are the concluded value for $(\mathrm{bit}_{i+1}, \mathrm{lsb})$.

By Observation 5.1 and assuming that Lemma 5.8 can be applied, it is easy to see that this *will* give us $(\mathrm{bit}_{i+1}, \mathrm{lsb})$ if none of the three runs make an error in deciding the corresponding $B_v^R(S)$. However, to apply Lemma 5.8 we must check that the conditions there are satisfied.

First we see that from Lemma 5.7, the probability of not having good approximations in R can be made as small as we wish. Next we must check that if $(\tilde{b}_{i+1}, \tilde{b}_0)$ is a correct hypothesis then all but $O(1)$ of the samples will be taken in the J_v we are aiming for. From this will also follow that if $(\tilde{b}_{i+1}, \tilde{b}_0)$ satisfies the relation: $((\mathrm{bit}_{i+1} - \tilde{b}_{i+1})l(n)/2 + (\mathrm{lsb} - \tilde{b}_0)p_2^{(1)}, b_0 2^{\tau(n)-1}) = \pi(D_v)$, then all but $O(1)$ of the samples will be from $J_v + \pi(D_v)$. In one single sample there are three sources of errors that could sabotage this — (i): Bits $[P_j]_{i-(d_1+d_2)\lceil \log n \rceil}^{i}$ are wrong, (ii): Uncontrolled wraparound and (iii): A carry from the least significant bits propagates into bits $i, i-1, \ldots, i-(d_1+d_2)\lceil \log n \rceil$ of the sample point and we are already at the end of the interval. Using Lemma 5.2 it is easy to show that the total error probability of (i), (ii), (iii) is less than $4m^{-3/2}$. Therefore, the probability that more than $O(1)$ points end up outside is small. A calculation will show that all points are where we want them to be with probability at least $1 - \frac{1}{10tn}$ and so Lemma 5.8 can be applied. Hence, using the m points in $S_v(\tilde{b})$ in Lemma 5.8, the probability that $B_v^R(S_v(\tilde{b}))$ is correct is at least $1 - \frac{1}{5tn}$ and we follow the correct edge in the tree. The probability we follow the correct edges on all three runs is at least $1 - 3\left(\frac{1}{10tn} + \frac{1}{5tn}\right) > 1 - \frac{1}{tn}$. □



Theorem 5.10. Given $f(x)$ and an n^{-c}-oracle for $\mathrm{bit}_i(H_3^k)$, $(26c+26)\lceil \log n \rceil \le i \le n/k - c_c' \log n$ then for $p \in_U \mathcal{P}_k'$ it is possible to find a polynomial number of values, $\{x' \in \mathbb{Z}_p\}$ at least one satisfying $x' \equiv x \pmod{p}$ with probability at least $3n^{-c}/8$.

Proof sketch: Assume for simplicity that $i = n/(2k)$. Choose $p \in_U \mathcal{P}_k'$. With probability $n^{-c}/2$ this gives a $(n^{-c}/2, p)$-oracle. Next choose random $a, b \in \mathbb{Z}_p$. Based on all possible values, u, of $[ax+b]_{n/k-3/2\log m}^{n/k}$, $[ax+b]_{i-(d_1+d_2)\lceil \log n \rceil}^{i}$, set these bits to zero by modifying b. For all candidate $(\frac{3}{4} - \frac{k}{2n}, p)$-oracles found in Lemma 5.9: Find $\mathrm{bit}_{i+1}, \mathrm{lsb}$ in $ax+b \bmod p$ and set them to zero by: $b' = b - \mathrm{lsb} - 2^{i+1}\mathrm{bit}_{i+1}$. Right-shift by $2^{-1}(ax+b') \bmod p$ and continue with the next two bits.

Let $\{x'\}$ be the corresponding values for $x \bmod p$ found for each of the above possibilities. For the correct choice of u, the "good" oracle O will determine $\mathrm{bit}_{i+1}, \mathrm{lsb}$ with probability at least $\frac{1}{4} + \frac{3}{4} - \frac{k}{2n} = 1 - \frac{k}{2n}$ so the probability that O is correct on all $\frac{n}{2k}$ calls is at least $1 - \frac{n}{2k}\frac{k}{2n} = 3/4$. For the running time, note that each of the polynomially many choices can be tried in polynomial time. □

The situation is now the same as in [**10**]: Apply Theorem 5.10 for many, say $10kn^c$ ps. For all $O(n^{ck})$ k-subsets of the the ps and for each (of the polynomially many) modular equations obtained for each p, use Chinese remaindering to get a suggestion for x. We get a polynomial number of suggestions and with high probability at least one of them will be correct (verify against $f(x)$). Hence, as almost every p in $\mathcal{P}_k$ also belongs to $\mathcal{P}'_k$:

Theorem 5.11. For any $k > 0$ and any i, $(26c+26)\lceil \log n \rceil \leq i \leq n/k - c'_c \log n$, $\mathrm{bit}_i(H_2^k)$ is a hard core predicate for any one-way function.

6. Security of the Leftmost Bits

We will now study the $O(\log n)$ most significant bits. If the bias of the ith bit now still is smaller than $n^{-c}/4$, we use the methods from the previous section. Otherwise the bit might *not* be hard core according to previous notation. So we use definitions introduced in [**9**]:

Definition 6.1. Let H be a family of boolean functions having bias $\beta < 1/2$ and let O be a pptm. The quantity

$$\frac{\Pr[O(f(x),h) = h(x) \mid O(f(x),h) = 1]}{1/2 - \beta} + \frac{\Pr[O(f(x),h) = h(x) \mid O(f(x),h) = 0]}{1/2 + \beta},$$

the probability taken over $h \in_U H$, $x \in_U \{0,1\}^n$ and the random choices of O, is called the *weighted success ratio* of O and is denoted by $\mathrm{ws}_O(H,f)$.

Let f be a function. Call H a hard core predicate for f if no O exists with $\mathrm{ws}_O(H,f) \geq 2 + g(n)$ for non-negligible g.

A well known lemma, also from [**9**], states that for an oracle with a non-negligible $g(n)$ as above, the fraction of correct 1-answers must differ non-negligible from the fraction of erroneous 1-answers. Thus we can replace Lemma 5.4 by this generalization and all is set to prove Lemma 5.5 in the new setting. In fact, now only one pair of intervals is needed since only one bit, the lsb, is unknown at each instant. One small detail must be taken care of: According to the definition of the set $\mathcal{P}'_k$, we need $\log N(n)$ bits to the left of bit i to make sure there is a good shift. On the other hand, it is easy to see that the good shift can not be greater than roughly $|p| - i$ and we can thus change the definition of $\mathcal{P}'_k$ accordingly.

Theorem 6.1. Each of the $O(\log n)$ most significant bits in H_2^k is a hard core predicate for any one-way function.

7. Discussion and Open Problems

The simultaneous hardness of the internal bits remain open. The fact that we could show hardness for each individual bit was due to the new technique of determining the bits two by two. However, it seems that the fact that two bits have influence over what the oracle gets as input also makes it impossible to use the method to show simultaneous security in the general case. The natural approach, reducing to a next-bit test (see [**12**]), doesn't seem to work as we do not know the bits close to the ith.

The reduction from the inversion of $f(x)$ is polynomial time but of rather large degree. Is there a simpler, more security preserving reduction (a simpler proof)?

Finally, we ask if it would be possible to improve the results to allow for primes significantly shorter than $\Omega(n)$, say $|p| = n^{1-\varepsilon}$.

8. Acknowledgment

First I would like to thank my supervisor, Johan Håstad for invaluable discussions and providing me with a great number of ideas. I must also express my gratitude towards Mikael Goldmann; his engagement was of great help during last minute changes to this paper.

I am also grateful for discussions and LaTeX help from my colleagues Christer Berg and Staffan Ulfberg. Roger Fischlin at U. Frankfurt was a very ambitious reader of [**10**], and his comments have influenced this paper. Finally, I thank Andrew Odlyzko for pointing out the work in [**7**].

References

[1] W. Alexi, B. Chor, O. Goldreich and 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] L. Blum, M. Blum and M. Shub: *A simple Unpredictable Pseudo-random Number Generator.* SIAM J. on Computing vol 15, no 2 1986, pp. 364–383.

[3] M. Blum and S. Micali: *How to Generate Cryptographically Strong Sequences of Pseudo-random Bits.* SIAM J. on Computing vol 13, no 4 1986 pp. 850–864.

[4] O. Goldreich and L. A. Levin: *A Hard Core Predicate for any One Way Function.* STOC 1989, pp. 25–32.

[5] S. Goldwasser and S. Micali: *Probabilistic Encryption.* JCSS vol 28, no 2, 1984, pp. 270–299.

[6] J. Håstad, A. W. Schrift and A. Shamir: *The Discrete Logarithm Modulo a Composite Hides $O(n)$ Bits.* JCSS 47 1993, pp. 376–403.

[7] L. Kuipers and H. Niederreiter: *Uniform Distribution of Sequences.* John Wiley & Sons 1974, ISBN 0-471-51045-9.

[8] D. L. Long and A. Wigderson: *The Discrete Log hides $O(\log n)$ bits.* SIAM J. on Computing vol 17, no 2 1988 pp. 413–420.

[9] A. W. Schrift and A. Shamir: *On the Universality of the Next Bit Test.* Proceedings Crypto 1990, LNCS 537, pp. 394–408, Springer Verlag.

[10] M. Näslund: *Universal Hash Functions & Hard Core Bits.* Proceedings Eurocrypt 1995, LNCS 921, pp. 356–366, Springer Verlag.

[11] U. V. Vazirani and V. V. Vazirani: *Efficient and Secure Pseudo-Random Number Generation.* Proceedings FOCS 1984, pp. 458–463.

[12] A. C. Yao: *Theory and Applications of Trapdoor Functions.* Proceedings FOCS 1982, pp. 80–91.

Appendix A. Proof Sketch to Lemma 5.5

We first need the following results:

Lemma A.1. If α is of (Q, ψ)-type and $N \leq Q/2$, then there is an absolute constant B such that

$$\mathcal{D}((\alpha)_N) \leq \frac{B\psi \log^2 N}{N}.$$

Proof sketch: We use the Erdös-Turán Theorem (Theorem 1.5 in [**7**]) and adaptions of lemmas similar to Lemma 3.2, 3.3 of [**7**]. Space constraints forces us to omit further details. □

Let $\mathcal{U}_s$ be the uniform distribution on $\mathbb{Z}_s$.

Lemma A.2. Let $s,m,N \in \mathbb{Z}$ with $m,N < s$. Suppose X,W are independent random variables with $X \in \mathcal{U}_m$, $W \in \mathcal{U}_N$. Let $v \in \mathbb{Z}_s$ and define the rational number $\alpha = v/s$. Finally, let $\mathcal{V}$ be the distribution of $Z = X + vW \bmod s$. Then Z is within $s\mathcal{D}((\alpha)_N)/m$ of $\mathcal{U}_s$.

Proof: For any $j \in \mathbb{Z}_s$ we have

$$\begin{aligned}
\Pr_{X,W}[X + vW \bmod s = j] &= \Pr_W[vW \bmod s = t \in (j - m \bmod s, j]] \Pr_X[X = j - t] \\
&= \Pr_W\left[\alpha W \bmod 1 = \frac{t}{s} \in \left(\frac{j-m}{s} \bmod 1, \frac{j}{s}\right]\right] \Pr_X[X = j - t] \\
&= \frac{A((\alpha)_N, ((j-m)/s \bmod 1, j/s])}{N} \frac{1}{m} \\
&\in \left(\frac{m}{s} \pm \mathcal{D}((\alpha)_N)\right)\frac{1}{m} = \frac{1}{s}\left(1 \pm \frac{s\mathcal{D}((\alpha)_N)}{m}\right).
\end{aligned}$$

□

Within the space limits of this paper, we can now briefly sketch the idea behind the proof of Lemma 5.5. Recall the definition of the two-dimensional space $I \times Y$. Let I^0 be the "left" half of I, $\{0,1,\ldots,l(n)/2-1\}$, (All $j \in I^0$ have their msb, which is bit_i in the representation mod p, equal to 0) and I^1 the right half, $\{l(n)/2,\ldots,l(n)-1\}$ (where $\text{bit}_i = 1$). Let $D = \pi_I(\lfloor p/2^{\tau(n)} \rfloor)$. Supposedly, there must be a $y \in Y$ such that

$$\Delta((I_1,y),(I_0,y)) \geq n^{-c}/2,$$

this follows from Lemma 5.4. W.l.o.g, assume that $y = 0$. Divide I^0 into sections of the form

$$I_j^0 = [\frac{l(n)}{2} - jD, \frac{l(n)}{2} - (j-1)D - 1], j = 1,\ldots,K-1$$

and $I_K^0 = [0, \frac{l(n)}{2} - (K-1)D - 1]$. Likewise, make a similar construction in I^1, producing I_j^1, $j = 1,2,\ldots,K$. Now consider moving all these intervals in unit steps up the y-axis, and in steps by D along the I-axis. For instance, for some I_j^s, $s = 0$ or 1, we move over

$$(I_j^s,0),(I_j^s+D,1),(I_j^s+2D,2),\ldots,(I_j^s+(2^{\tau(n)}-1)D,2^{\tau(n)}-1)$$

Now, if either for $s = 0$ or 1 there is a j and a t such that $\Delta((I_j^s+(t-1)D,t-1),(I_j^s+tD,t))$ is non-negligible we are done since this is precisely what we want. Can we be sure that there is such s,j,t?

By assumption, $D/l(n)$ is of $(2N(n),\psi(n))$-type. Therefore, by Lemma A.2, choosing a random t, we see that tD is close to the uniform distribution on $\{0,1,\ldots,l(n)-1\} = I$. Now, we know that the average fraction of 1-answers the oracle gives on I^0 differs non-negligible from that on I^1. On the other hand, the discussion above implies that each term of the form $P_1((I_j^1+tD,t))$ contributes (almost) as much to $P_1(I^1)$ as it does to $P_1(I^0)$. So if they are all the same, this would mean that $P_1(I^0)$ is close to $P_1(I^1)$!

This also motivates why we can not hope that the first shift, $\tau = 1$, will work. We then have just two copies of each I_j^s. There is no way we can argue that one of them is in I^1 and one in I^0, they may very well both be in the same half.