Nearly One-Sided Tests and the Goldreich-Levin Predicate

Gustav Hast

Department of Numerical Analysis and Computer Science
Royal Institute of Technology, 100 44 Stockholm, Sweden
ghast@nada.kth.se

Abstract. We study statistical tests with binary output that rarely outputs one, which we call nearly one-sided statistical tests. We provide an efficient reduction establishing improved security for the Goldreich-Levin hard-core bit against nearly one-sided tests. The analysis is extended to prove the security of the Blum-Micali pseudo-random generator combined with the Goldreich-Levin bit.

Furthermore, applications where nearly one-sided tests naturally occur are discussed. This includes cryptographic constructions that replace real randomness with pseudo-randomness and where the adversary's success easily can be verified. In particular, this applies to signature schemes that utilize a pseudo-random generator as a provider of randomness.

Keywords: Nearly one-sided statistical test; Goldreich-Levin predicate; Pseudorandom generator; Provable security; List decoding.

1 Introduction

Many algorithms are probabilistic and therefore require a source of randomness to be implemented correctly. This is true in particular for most cryptographic algorithms. Obtaining random material is often a hard and time consuming process and therefore it is convenient to use a pseudo-random generator to generate much random looking material from a short truly random seed. One would of course like to have a guarantee that by exchanging random material for the output of a generator, the performance of the algorithms are not changed in a harmful way.

The pioneering works of Blum and Micali [6] and Yao [19] laid the foundation of the theory of pseudo-randomness. Blum and Micali showed how to construct a pseudo-random bit generator (PRBG) whose security is based on the hardness of solving the discrete logarithm problem. More specifically, they proved and used the fact that the most significant bit is a hard-core predicate for exponentiation. A predicate b is a hard-core predicate for a function g if it is not feasible to efficiently determine the boolean value of $b(x)$ given the value of $g(x)$. Goldreich and Levin [11] showed how to construct such a hard-core predicate from any one-way function. This construction can be applied on the above mentioned PRBG so that the security can be based on the one-wayness of an arbitrary permutation f. The proof of security provided by [6] was a polynomial reduction from solving the discrete logarithm problem (or if we use the result from [11] inverting f), to breaching the security of the bit generator.

In this work we analyze the security of the well-known pseudo-random generator obtained by combining the works of Blum and Micali [6] and Goldreich and Levin [11]. We refer to this generator as $BMGL$. As noted in previous works (e.g., [10], [11] and [16]) the exact efficiency of a reduction between two different cryptographic primitives is of vital interest when determining the practical security consequences of the reduction. Examples of more recent works that deal with the issue to bridge the gap between theoretical complexity based cryptography and practical cryptography by improved reductions and analysis are [3] and [7]. For a more extensive list see [2]. In the case of $BMGL$, the reduction relates the one-wayness of a permutation to the pseudo-randomness of the output from $BMGL$. A distribution is considered to be pseudo-random if there is no statistical test, from a specific set of admissible tests, that more than negligibly can distinguish between elements from that distribution and from the uniform distribution. (A distribution is in fact not considered to be pseudo-random, but instead an ensemble of distributions. For simplicity reasons we do not make this distinction throughout the introduction.) Normally, the set of admissible tests is specified by a maximum running time. Improvements to the security reduction of $BMGL$ and its analysis have earlier been made by Rackoff (explained in [8]), Levin [17] and Håstad and Näslund [15]. Apart from $BMGL$, there have been numerous other constructions of pseudo-random generators based on the Blum-Micali paradigm, for example constructions exploiting the hardness of the factoring problem starting with the work of Blum et al. [4].

Earlier analyses of reductions have characterized the efficiency of a statistical test D, distinguishing the distributions X and Y, by using a measure δ such that

$$\left| \Pr_{x \in X}[D(x) = 1] - \Pr_{y \in Y}[D(y) = 1] \right| \geq \delta \ . \tag{1}$$

In this paper we consider nearly one-sided statistical tests (this concept has been investigated earlier by Blum and Goldreich [5]) which are tests that, on truly random input, almost always output zero and rarely output one. The measure in (1) does not capture whether or not a test is nearly one-sided and therefore we introduce the notion of a parameterized distinguisher, which for a test and a certain pair of input distributions impose two thresholds, separating the corresponding output distributions of the test. We say that a test D (δ_1, δ_2)-distinguishes X and Y if

$$\Pr_{x \in X}[D(x) = 1] \leq \delta_1 < \delta_2 \leq \Pr_{y \in Y}[D(y) = 1] \ .$$

Thus, if δ_1 is small and X is the uniform distribution, the test is considered to be nearly one-sided. (We do not formally define "small" but instead use the parameterized distinguisher to express formal results in this paper.) The use of this characterization of a distinguisher enables a more careful analysis of the reduction from inverting a permutation to distinguishing between the output of $BMGL$ and the uniform distribution. The analysis shows that the success probability of inverting the permutation is proportional to $(\delta_2 - \delta_1)^2/\delta_2$, to be compared to previous results obtaining $(\delta_2 - \delta_1)^2$. If δ_1 is small compared to δ_2 (as is the case with nearly one-sided tests) the increase in reduction quality is significant.

The heart of the improved analysis is in the analysis of the Goldreich-Levin hard-core bit. Essentially, the classical proof shows that if the adversary can predict the hard-core bit with advantage ε, then one can invert the one-way function with probability proportional to ε^2. This was shown by Adcock and Cleve [1] to be optimal in the general case. In the case of nearly one-sided tests, the distinguisher can be transformed to a predictor that most of the time has almost no advantage against a random guess, but for a small fraction of the inputs it has a significant advantage. In the classical proof this predictor had to make a guess even though its confidence was low. This caused the produced high quality predictions to be concealed by the big amounts of predictions of low quality. In our reduction we therefore allow the predictor also to output $\perp$, which means that the confidence is too low to make a prediction. This enables us to invert the one-way function with probability proportional to ε^2/p, where p is the probability that the predictor makes a prediction. (The seemingly contradiction that the success probability increases when the probability of making a prediction decreases is explained by the fact that this also implies that the confidence in the prediction increases.)

The reduction that was used to show that the Goldreich-Levin bit is hard to predict is essentially a list decoding algorithm of the Hadamard code. Using this viewpoint, the possibility for the predictor to output $\perp$ will correspond to an erasure in the Hadamard code. After the work of Goldreich and Levin [11] subsequent works in list decoding includes a generalization of the algorithm to the non-binary and non-linear case [12]. Sudan et al. [18] showed how to transform predicates, that are hard in the worst case, to become predicates that can be predicted only negligibly better than by a random guess. The transformation and proof was based on error-correcting codes and list decoding.

In cryptography the major application can be found in different types of authentication schemes as nearly one-sided tests often occur there naturally. For example, consider a signature scheme that is secure if given a source of random bits. Suppose that one instead feeds this signature scheme with bits from a bit generator (e.g. $BMGL$) and that it then no longer is secure. The attacker of this schemes now serves as a nearly one-sided distinguisher between the two different schemes because it has a negligible respectively non-negligible probability to produce a valid signature when attacking the two different schemes. Thus, this attacker can be combined with the signature scheme to build a nearly one-sided test that distinguishes between the output distribution of the bit generator and the uniform distribution. In Sect. 8, a more extensive discussion is made about possible applications.

The outline of this paper is as follows: In Sect. 3 the Goldreich-Levin hard-core bit is explained and we discuss why low rate predictors are more powerful than ordinary predictors when list decoding the Hadamard code. In the next section we prove a theorem about list decoding Hadamard codes with both erasures and errors and in Sect. 5 this theorem is used to establish the reduction from inverting a function to predicting the Goldreich-Levin bit. Section 6 discusses statistical tests and their connection with predictors and in Sect. 7 the security of the $BMGL$ is shown. The paper is concluded with some applications and open questions.

2 Notation

In this work we use the following notation:

1. By $[m]$ we mean the set $\{1, \ldots, m\}$ and $2^{[m]}$ is the set of all subsets of $[m]$.
2. The xor operation is denoted by $\oplus$.
3. The function $b(r, x)$ is the inner product between r and x modulo 2.
4. The i'th unit vector e^i is a bit string containing only zeros, except for the i'th bit (which is 1). The dimension of e^i is implicitly given by its use.
5. We let $\langle J, L\rangle$, where J and L are sets, denote the size of $J \cap L$ modulo 2.
6. If x is a bit string, the length of x is denoted by $|x|$.
7. When the logarithmic function log is used without the base having been specified, it is implicit base 2.
8. The uniform distribution of bit strings of length n is denoted by U_n.

3 The Goldreich-Levin Bit and List Decoding of Hadamard Code

Goldreich and Levin [11] showed how to modify an arbitrary one-way function to make it have a hard-core predicate: if f is a one-way function, then $b(r, x)$ (the inner product of r and x modulo 2) is a hard-core predicate for the one-way function $f'(r, x) = (r, f(x))$. This means that there is no efficient algorithm that given $(r, f(x))$ as input (where r and x are drawn from the uniform distribution) can guess the value of $b(r, x)$ significantly better than a random guess. We do not formally define hard-core predicate as the results in this paper are instead expressed in terms of the advantage and rate of a predictor (see Definition 2).

The above result is shown using a reduction from inverting f to predicting the value of $b(r, x)$. The efficiency of the reduction depends on how well the bit $b(r, x)$ is guessed, which usually is measured by the advantage $\varepsilon(n)$ of the guessing algorithm P, often called the predictor:

$$\varepsilon(n) = \Pr_{r,x \in U_n} [P(r, f(x)) = b(r, x)] - \frac{1}{2} \ .$$

The main part of the reduction consists of a list decoding algorithm for the binary Hadamard code.

Definition 1. *The (binary) Hadamard code of a bit string x of length n is $\langle b(r,x)\rangle_{r\in\{0,1\}^n}$.*

The i'th bit of the Hadamard code of x is thus exactly $b(i,x)$, where i is interpreted in the natural way as a bit string of the same length as x. The task for a list decoding algorithm is to produce a list of possible x, having oracle access to a Hadamard code with a certain fraction of errors. The algorithm should come with a lower bound on the probability that x appears in the list output and an upper bound on the number of oracle queries made. Given the value of $f(x)$, the predictor P corresponds in a natural way to the oracle of the Hadamard code, and the advantage of P (over a fix x) is closely related to the number of errors of the oracle.

Now suppose that we have a predictor P that on some input answers with high confidence and on other inputs just flips a coin. The informative answers from P (when it does not just flip a coin) would then be somewhat clouded by the random noise provided by the other answers. Let us therefore give the predictor more freedom by also letting it output $\perp$ in those cases where the confidence in the prediction is low. Instead of characterizing this type of predictor with only its advantage in the traditional sense, we also use its rate, which is how often it outputs a prediction. The advantage is generalized in a natural way for this different type of predictor.

Definition 2. *A predictor $P : \{0,1\}^* \times \{0,1\}^* \to \{0,1,\perp\}$ is said to have rate $\delta(n)$ and advantage $\varepsilon(n)$ in predicting $b(x,r)$ from $f(x)$ and r, where*

$$\delta(n) = \Pr_{x,r\in U_n}[P(f(x),r)\neq\perp]$$

and

$$\varepsilon(n) = \Pr_{x,r\in U_n}[P(f(x),r)=b(x,r)] - \frac{1}{2}\Pr_{x,r\in U_n}[P(f(x),r)\neq\perp] \ .$$

Going back to the list decoding algorithm for the Hadamard code, this new type of predictor corresponds to a Hadamard code oracle with both errors and erasures, where the fraction of erasures is $1-\delta$ (where δ is the rate of the predictor). The heart of the improved reduction is in the analysis of the list decoding algorithm with an oracle that has a relatively large part of erasures (or in other words a predictor with low rate). Let us briefly discuss why a predictor with low rate is more powerful than one with a higher rate and the same advantage. With more powerful we here mean that the predictor does not have to be called as many times in the list decoding algorithm. Later we show how a nearly one-sided test for the Goldreich-Levin bit easily can be turned into a predictor with low rate.

Assume that P is a predictor with rate δ and advantage ε. Earlier analyses, that only made use of the advantage, have shown (see [8], Sect. 2.5.2) that the number of needed calls to P should be at least proportional to ε^{-2} for the list decoding algorithm to succeed with probability one half. The probability that

P makes a correct prediction is $\frac{1}{2}\delta + \varepsilon$. Let us now ignore all the calls that received $\perp$-answers from P. The probability that P, on the remaining calls, guesses correctly is then $(\frac{1}{2}\delta + \varepsilon)\delta^{-1} = \frac{1}{2} + \delta^{-1}\varepsilon$. In some sense this gives us a not fully working predictor with advantage $\delta^{-1}\varepsilon$, the problem being that it does not make predictions for all inputs and that it on the average only makes one prediction per δ^{-1} calls. If the first problem mentioned can be dealt with in the list decoding algorithm we can expect that the number of calls to P is proportional to the inverted advantage squared times the extra factor of δ^{-1}: $O((\delta^{-1}\varepsilon)^{-2}\delta^{-1}) = O(\delta\varepsilon^{-2})$. Note that if the advantage of P is at least a constant fraction of the rate, we have that $\delta = O(\varepsilon)$ and thus the number of calls needed would only be $O(\varepsilon^{-1})$ compared to $O(\varepsilon^{-2})$ before. In the proofs of Theorem 1 and 2 we show that this intuition really has merit.

4 List Decoding of Hadamard Code with Errors and Erasures

The main part of the proof that the Goldreich-Levin bit is a hard-core predicate consists of a list decoding algorithm of a binary Hadamard code with errors. To ensure that the power of the low rate of the predictor does not vanish, we repeat the analysis of the list decoding algorithm (not the original one [11], but one due to Rackoff explained in [8]) while letting the Hadamard code also contain erasures. As far as the author is aware of, no previous work has been done on list decoding the Hadamard code in the presence of errors and erasures.

Theorem 1. *There is an algorithm* **LD** *that, on input l and n and with oracle access to a Hadamard code of x (where $|x| = n$) with an e-fraction of errors and an s-fraction of erasures, can output a list of 2^l elements in time $O(nl2^l)$ asking $n2^l$ oracle queries such that the probability that x is contained in the list is at least one half if $l \geq \log_2\left(\frac{8n(e+c)}{(c-e)^2} + 1\right)$, where $c \overset{def}{=} 1 - s - e$ (the fraction of correct answers from the oracle).*

Proof. Let C be an oracle that represents a Hadamard code of a fixed x with an e-fraction of errors and s-fraction of erasures. In Fig. 1 the list decoding algorithm $\mathbf{LD}^C$ is defined. First we prove its correctness in respect to the claim made in Theorem 1 that it outputs x with at least probability one half. We then analyze its time complexity.

Correctness of $\mathbf{LD}^C$: We start by proving the following claim about the value of $C'(e^i \oplus s^J)$ which is a principal component in the calculations made in step 3 of our list decoder $\mathbf{LD}^C$.

Claim 1 *Let s^J and C' be defined as in the description of $\mathbf{LD}^C$. Then for a nonempty $J \subseteq [l]$ and $L_0 = \{i \,|\, i \in [l],\, b(s^i, x) = 1\}$ the following equalities hold:*

$$\Pr[(-1)^{\langle J,L_0\rangle}C'(e^i \oplus s^J) = 0] = s$$
$$\Pr[(-1)^{\langle J,L_0\rangle}C'(e^i \oplus s^J) = (-1)^{b(e^i,x)}] = c$$
$$\Pr[(-1)^{\langle J,L_0\rangle}C'(e^i \oplus s^J) = -(-1)^{b(e^i,x)}] = e,$$

Implementation of list-decoder $\mathbf{LD}^C$: Let $\mathbf{LD}$ have oracle access to $C : \{0,1\}^n \to \{0,1,\perp\}$. On input l and n, $\mathbf{LD}^C$ proceeds as follows:

1. Choose s^1,…, s^l independently from U_n.
2. Define a predictor C' that uses C so that
$$C'(r) = \begin{cases} 1 & \text{if } C(r) = 0 \\ -1 & \text{if } C(r) = 1 \\ 0 & \text{if } C(r) = \perp \end{cases}$$
3. Calculate $d^i_L = \sum_J (-1)^{\langle J,L\rangle} C'(e^i \oplus s^J)$ for all $L \subseteq [l]$ and $i \in [n]$, where $s^J \overset{\text{def}}{=} \oplus_{i \in J}\, s^i$ for all nonempty $J \subseteq [l]$.
4. Output the list $\{z^L\}_{L \subseteq [l]}$ where the i'th bit of z^L is defined as $\frac{1-\text{sgn}(d^i_L)}{2}$.

Fig. 1. The List-Decoder $\mathbf{LD}^C$

where the probabilities are taken over the choices of s^i in step 1 of $\mathbf{LD}^C$*.*

Proof. We observe that

$$b(s^J, x) = \oplus_{j \in J} b(s^j, x) = \oplus_{j \in J \cap L_0} b(s^j, x) = \langle J, L_0 \rangle \ ,$$

and

$$C'(e^i \oplus s^J) = \begin{cases} (-1)^{b(e^i,x) \oplus b(s^J,x)} & \text{if } C \text{ answers correctly} \\ -(-1)^{b(e^i,x) \oplus b(s^J,x)} & \text{if } C \text{ answers incorrectly} \\ 0 & \text{if } C \text{ answers } \perp \ . \end{cases}$$

As J is non-empty, the value of $e^i \oplus s^J$ will be uniformly distributed and thus the probability that C answers '$\perp$' is s, incorrectly is e and correctly is c. As $b(s^J, x) = \langle J, L_0 \rangle$ the claim follows from

$$(-1)^{b(s^J,x)} C'(e^i \oplus s^J) = \begin{cases} (-1)^{b(e^i,x)} & \text{if } C \text{ answers correctly} \\ -(-1)^{b(e^i,x)} & \text{if } C \text{ answers incorrectly} \\ 0 & \text{if } C \text{ answers } \perp \ . \end{cases}$$

□

As a consequence of their construction, the values of s^J, for nonempty $J \subseteq [l]$, are pairwise independent and uniformly distributed. Let L_0 be defined as $\{i \,|\, i \in [l],\, b(s^i, x) = 1\}$ and study for a fixed i the value of $d^i_{L_0}$ calculated in step 3 of $\mathbf{LD}^C$. It is a sum of expressions on the form that is analyzed in Claim 1. The probability of different results (expressed in terms of the i'th bit of x) of each term in this sum is specified in the claim. Using the value (sign) of the sum we can thereby guess the i'th bit of x and by knowing the different probabilities we can calculate an upper bound for the probability that the guess is incorrect.

For our guess to be correct we would like to have more terms that equal $(-1)^{b(e^i,x)}$ than $-(-1)^{b(e^i,x)}$. As we know the probability for each outcome we can, by using Chebyschev's inequality, give an upper bound on the probability that the guess is incorrect. For every non-empty $J \subseteq [l]$, define ζ_c^J to be the indicator variable for the event that $(-1)^{\langle J,L_0\rangle} C'(e^i \oplus s^J) = (-1)^{b(e^i,x)}$ and let ζ_e^J be the indicator variable for the event that $(-1)^{\langle J,L_0\rangle} C'(e^i \oplus s^J) = -(-1)^{b(e^i,x)}$. We would like to be able to state that $\sum_J \zeta_c^J > \sum_J \zeta_e^J$ with high probability. Chebyschev's inequality states that for any positive t

$$\Pr[|Y - \mu| \geq t\sigma] \leq t^{-2} \enspace ,$$

where σ is the standard deviation and μ is the expectation of the variable Y.

This inequality is to be applied on the number of incorrect answers $Y = \sum_J \zeta_e^J$ which has the expected value of $\mu = Ne$ (where $N \overset{\text{def}}{=} 2^l - 1$ is the number of terms in the sum) and the standard deviation σ is less than $\sqrt{Ne}$ (using the fact of pairwise independency). We set $t = \frac{\sqrt{N}(c-e)}{2\sqrt{e}}$ which gives us

$$\Pr\left[|\sum_J \zeta_e^J - Ne| > \frac{N(c-e)}{2} \right] \leq \frac{4e}{N(c-e)^2} \enspace .$$

Applying the same inequality on the number of correct answers $Y = \sum_J \zeta_c^J$ with $\mu = Nc$, $\sigma < \sqrt{Nc}$ and $t = \frac{\sqrt{N}(c-e)}{2\sqrt{c}}$ gives

$$\Pr\left[|\sum_J \zeta_c^J - Nc| > \frac{N(c-e)}{2} \right] \leq \frac{4c}{N(c-e)^2} \enspace .$$

If none of the sums $\sum_J \zeta_c^J$ and $\sum_J \zeta_e^J$ deviate more than $\frac{N(c-e)}{2}$ from their expected values we can conclude that the number of correct answers outnumbers the number of incorrect answers. Thus, the probability that this is not the case and we thereby are not able to make a correct prediction is at most $\frac{4(e+c)}{N(c-e)^2}$.

For the algorithm to succeed (in the supposed fashion) each of the n different bits of x has to be predicted correctly. In other words $d_{L_0}^i$ has to have the correct sign for each $i \in [n]$. An upper bound for this not occurring is $\frac{4n(e+c)}{N(c-e)^2}$ which is the sum of the upper bounds for each bit prediction failure. If $N \geq \frac{8n(e+c)}{(c-e)^2}$ then this bound is less than one half. As N equals $2^l - 1$ we conclude that if the input l satisfy

$$l \geq \log\left(\frac{8n(e+c)}{(c-e)^2} + 1 \right)$$

then the probability that x appears in the output list is at least one half.

Efficiency of $\mathbf{LD}^C$: The first step of $\mathbf{LD}^C$ takes time $O(nl)$ and the last step takes time $O(n2^l)$. The time consuming step of the algorithm is the calculation of the different values of d_L^i. The naive way to do this would be by calculating each d_L^i-value independently for each L. This would make the algorithm work in time $O(n^2 2^{2l})$ making $O(n2^l)$ calls to C, as there are $n2^l$ different d_L^i-values

Description of inverter Inv: P is a predictor with rate $\delta_P(n)$ and advantage $\varepsilon_P(n)$. On input $y = f(x)$ and n **Inv** proceeds as follows:

1. Select j from $\{-1, \ldots, h-2\}$, where $h = \lfloor h(n) \rfloor = \lfloor \log \frac{\delta_P(n)}{\varepsilon_P(n)^2} \rfloor$, with probability 2^{j-h} and set $l = \lceil \log(n\delta_P(n)/\varepsilon_P(n)^2) \rceil - j + 2$. If no j is chosen, stop and output $\perp$.
2. Call the list-decoder $\mathbf{LD}^{P_y}$ with input l and n, where $P_y(\cdot) \overset{\text{def}}{=} P(y, \cdot)$.
3. Apply f on each element x' of the output from the list-decoder. If $f(x') = y$, then output x' and stop.
4. Output $\perp$.

Fig. 2. The inverter **Inv**

and each value is a sum of $2^l - 1$ terms and each term can be calculated in time $O(n)$.

Using Fourier-analysis of functions $g : 2^{[l]} \to \mathbb{R}$, the expression for d_L^i, $\sum_J (-1)^{\langle J,L \rangle} C'(e^i \oplus s^J)$, can be identified as the L'th Fourier coefficient of the function $g_i(J) = C'(e^i \oplus s^J)$. Using the fast Fourier-transform algorithm all 2^l Fourier coefficients can be calculated in time $O(l2^l)$. Therefore, for each i we can calculate all the values $\{d_L^i\}_{L \subseteq [l]}$ in time $O(l2^l)$ using 2^l oracle queries. There are n different values of i making the total time $O(nl2^l)$ and the total number of oracle queries $n2^l$. □

5 Goldreich-Levin Hard-Core Bit

In this section we make an efficient reduction from inverting a function f to predicting the Goldreich-Levin bit of f. The list decoding algorithm from the previous section is the main component of the algorithm that performs the reduction.

Theorem 2. *Let P be a probabilistic algorithm with running time $t_P : \mathbb{N} \to \mathbb{N}$, and rate $\delta_P : \mathbb{N} \to [0,1]$ and advantage $\varepsilon_P : \mathbb{N} \to [0, \frac{1}{2}]$ in predicting $b(x,r)$ from $f(x)$ and r. Define $h(n)$ to be $\log_2(\delta_P(n)/\varepsilon_P(n)^2)$. Then there exists an algorithm* **Inv** *that runs in expected time $(t_P(n) + h(n)\log_2(n)) \cdot h(n) \cdot O(n^2)$ and satisfies*

$$\Pr_{x \in U_n}\left[f(\mathbf{Inv}(f(x), n)) = f(x)\right] = \Omega\left(\frac{\varepsilon_P(n)^2}{\delta_P(n)}\right) .$$

Theorem 2 states that if there is an algorithm P that predicts the Goldreich-Levin bit of f, then there also exists an algorithm **Inv** (depicted in Fig. 2) that inverts f. If P, for all possible values of x, would have approximately the same advantage (and rate) in predicting $b(r,x)$ from $f(x)$ and r, this can be shown by directly applying the list-decoding algorithm **LD** with the appropriate value of l. But as this is not true in general we need to make an averaging argument. This is done by calling **LD** with small values on l with high probability (to cope with values of x giving P a high advantage), and calling **LD** with big values on l with low probability (to cope with values of x giving P an intermediate advantage). A proof of the theorem can be found in the journal version of this work [14].

Compared to previous analyses the inverting probability improves with approximately a factor of δ_P^{-1} (see Proposition 2.5.4 in [8]). In some applications we know that the advantage $\varepsilon_P(n)$ will be a constant fraction of the rate $\delta_P(n)$ and then the improvement will be considerable as the inverting probability will increase from $\Omega(\varepsilon_P(n)^2)$ to $\Omega(\varepsilon_P(n))$.

Knowledge about the value of $\frac{\varepsilon_P(n)^2}{\delta_P(n)}$ is required if we would like to implement **Inv**. Therefore we define a new algorithm **Inv′**, with oracle access to a predictor P, that takes an additional input h and behaves exactly as **Inv** but uses h instead of $\log \frac{\delta_P(n)}{\varepsilon_P(n)^2}$ in the first step of the algorithm. From the proof of Theorem 2 we can conclude that as long as $h \geq \log \frac{\delta_P(n)}{\varepsilon_P(n)^2}$ the probability of success will not decrease and the running time and the number of queries to P will be the same as in Theorem 2 except for that we substitute $\log \frac{\delta_P(n)}{\varepsilon_P(n)^2}$ with h. We thus have the following corollary which is useful when proving Theorem 3.

Corollary 1. *Let P be an arbitrary algorithm predicting $b(x,r)$ from $f(x)$ and r. There exists an algorithm* **Inv′** *with oracle access to P such that on input y, n and h it runs in expected time $h^2 \log_2(n) \cdot O(n^2)$ and makes $h \cdot O(n^2)$ number of expected calls and satisfies*

$$\Pr_{x \in U_n}\left[f(\mathbf{Inv'}^P(f(x), n, h)) = f(x)\right] = \Omega\left(2^{-h}\right)$$

if $h \geq \log_2(\delta_P(n)/\varepsilon_P(n)^2)$, where P has rate $\delta_P : \mathbb{N} \to [0,1]$ and advantage $\varepsilon_P : \mathbb{N} \to [0, \frac{1}{2}]$ in predicting $b(x,r)$ from $f(x)$ and r.

6 Nearly One-Sided Statistical Tests

A statistical test is a probabilistic algorithm that takes an input and outputs a bit. The purpose of a statistical test is to distinguish between different distributions. This is done by having different output distributions when the input is drawn from different distributions. The output distribution is characterized by the probability that the output is equal to 1 respectively 0. If the test rarely outputs 1 we consider the test to be nearly one-sided with respect to the input distribution. The classical way to measure how well a statistical test distinguishes between two distributions (or in fact two ensembles of distributions) is through the following definition.

Definition 3. *An algorithm $D : \{0,1\}^* \to \{0,1\}$ $\delta(n)$-distinguishes the ensembles $X = \{X_n\}$ and $Y = \{Y_n\}$ if for infinitely many values of n*

$$\left|\Pr_{x \in X_n}[D(x) = 1] - \Pr_{y \in Y_n}[D(y) = 1]\right| \geq \delta(n) \enspace .$$

The characterization of a statistical test in terms of this definition is rather coarse. In particular whether the test is one-sided or not does not show. This is remedied by the following definition which explicitly provides absolute bounds on the probability of outputting 1.

Definition 4. *An algorithm $D : \{0,1\}^* \to \{0,1\}$ $(\delta_1(n), \delta_2(n))$-distinguishes the ensembles $X = \{X_n\}$ and $Y = \{Y_n\}$ if for infinitely many values of n*

$$\Pr[D(X_n) = 1] \le \delta_1(n) < \delta_2(n) \le \Pr[D(Y_n) = 1] \ .$$

In addition, D is said to be a $(\delta_1(n), \delta_2(n))$-distinguisher for the ensembles X and Y if D $(\delta_1(n), \delta_2(n))$-distinguishes X and Y.

We now discuss the connection between statistical tests and predictors, and in particular how a nearly one-sided statistical test for the Goldreich-Levin bit easily can be turned into a low rate predictor that predicts the Goldreich-Levin bit.

Assume that a distinguisher D satisfies

$$p_1 = \Pr_{r,x \in U_n, \sigma \in U_1} [D(f(x), r, \sigma) = 1]$$

and

$$p_2 = \Pr_{r,x \in U_n} [D(f(x), r, b(r,x)) = 1] \ .$$

It is easy to transform this distinguisher into a predictor P guessing $b(r,x)$ with advantage $p_2 - p_1$. On input $(f(x), r)$ the predictor P samples a uniform bit σ, queries for $D(f(x), r, \sigma)$ and outputs σ iff $D(f(x), r, \sigma) = 1$ and otherwise outputs $1 - \sigma$. However, if the probability p_1 is very small, then the truly informative answers from D are when it returns 1. In those cases the probability that the prediction is correct is

$$\frac{\Pr_{r,x \in U_n} [D(f(x), r, \sigma) = 1 \,|\, \sigma = b(r,x)] \cdot \Pr_{\sigma \in U_1} [\sigma = b(r,x)]}{\Pr_{r,x \in U_n, \sigma \in U_1} [D(f(x), r, \sigma) = 1]} = \frac{p_2}{2p_1} \ ,$$

giving an advantage of

$$\frac{p_2}{2p_1} - \frac{1}{2} = \frac{p_2 - p_1}{2p_1} \ .$$

This should be compared to the success probability when D outputs 0

$$\frac{\Pr_{r,x \in U_n} [D(f(x), r, \sigma) = 0 \,|\, 1 - \sigma = b(r,x)] \cdot \Pr_{\sigma \in U_1} [1 - \sigma = b(r,x)]}{\Pr_{r,x \in U_n, \sigma \in U_1} [D(f(x), r, \sigma) = 0]}$$

$$= \frac{\frac{1}{2} \cdot \Pr_{r,x \in U_n} [D(f(x), r, 1 - b(r,x)) = 0]}{1 - \Pr_{r,x \in U_n, \sigma \in U_1} [D(f(x), r, \sigma) = 1]}$$

$$= \frac{\frac{1}{2} \cdot (1 - \Pr_{r,x \in U_n} [D(f(x), r, 1 - b(r,x)) = 1])}{1 - p_1}$$

$$= \frac{1 - (2 \cdot \Pr_{r,x \in U_n, \sigma \in U_1} [D(f(x), r, \sigma) = 1] - \Pr_{r,x \in U_n} [D(f(x), r, b(r,x)) = 1])}{2(1 - p_1)}$$

$$= \frac{1 - (2p_1 - p_2)}{2(1 - p_1)} \ ,$$

giving an advantage of

$$\frac{1-(2p_1-p_2)}{2(1-p_1)}-\frac{1}{2}=\frac{1-(2p_1-p_2)-(1-p_1)}{2(1-p_1)}=\frac{p_2-p_1}{2(1-p_1)} \ .$$

(Note that we have implicitly extended the notion of advantage to also apply when conditioned on the output of D.)

Assume that the test is nearly one-sided and thereby the value of p_1 is close to zero. In this case the advantage of the prediction when D outputs 1 is a factor p_1^{-1} better than if it outputs 0. This is why it is better to somewhat change P so that it still returns σ iff $D(f(x), r, \sigma) = 1$ but otherwise outputs $\perp$. This new predictor has rate p_1 and advantage $(p_2 - p_1)/2$. Thus we have transformed a nearly one-sided statistical test for the Goldreich-Levin bit into a low rate predictor that predicts the Goldreich-Levin bit.

7 Blum-Micali Pseudo Random Generator

Blum and Micali [6] constructed a pseudo-random bit generator based on a one-way permutation and a hard-core predicate associated with the permutation. As an example they used exponentiation modulo a prime as a one-way permutation and the most significant bit as its hard-core predicate. By using an arbitrary one-way permutation and the hard-core predicate of Goldreich-Levin [11] the following construction, referred to as $BMGL$, is obtained:

Construction 1 *Let $f : \{0,1\}^* \to \{0,1\}^*$ be a polynomial-time-computable length-preserving permutation and $l : \mathbb{N} \to \mathbb{N}$ where $n < l(n)$. Given the input x_0 and r such that $|x_0| = |r| = n$, define*

$$G^{f,l}(x_0, r) = b(r, x_1)b(r, x_2)\ldots b(r, x_{l(n)}) \ ,$$

where $x_i = f(x_{i-1})$ for $i = 1,\ldots,l(n)$. With $G_n^{f,l}$ we denote the output distribution of $G^{f,l}(x_0, r)$, where r and x_0 are chosen uniformly and independently from $\{0,1\}^n$.

The security of the $BMGL$ is described by our next theorem.

Theorem 3. *Let D be a $(p_1(n), p_2(n))$-distinguisher for $\{U_{l(n)}\}$ and $\{G_n^{f,l}\}$ with running time $t_D : \mathbb{N} \to \mathbb{N}$. Then there exists an algorithm* **Inv**$''$ *that runs in expected time $[(t_f(n)\cdot l(n)+t_D(n))+m(n)\log_2(n)]\cdot m(n)\cdot O(n^2)$, where $t_f(n)$ is the time to calculate the function f on a n-bit input and $m(n) \overset{def}{=} \log_2 \frac{p_2(n)l(n)^2}{(p_2(n)-p_1(n))^2}$, and satisfies*

$$\Pr_{x\in U_n}[\mathbf{Inv}''(f(x)) = x] = \Omega\left(2^{-m(n)}\right)$$

for infinitely many values of n.

Implementation of predictor P^i: The predictor has access to a distinguisher $D : \{0,1\}^{l(n)} \to \{0,1\}$. On input y and r, P^i proceeds as follows:

1. Set $x_{i+1} = y$ and calculate x_j for $j \in \{i+2, \ldots, l(n)\}$ where $x_j = f(x_{j-1})$ and $n = |y|$.
2. Toss i unbiased coins $c_1, c_2, \ldots, c_{i-1}$ and σ.
3. Create the bit string $z = c_1 \ldots c_{i-1} \sigma b(x_{i+1}, r) \ldots b(x_{l(n)}, r)$. If $D(z) = 1$ then return σ, otherwise return $\perp$.

Fig. 3. The predictor P^i

Implementation of inverter Inv$''$: The inverter has access to a distinguisher $D : \{0,1\}^{l(n)} \to \{0,1\}$. On input y, **Inv**$''$ proceeds as follows:

1. Choose $i \in [l(n)]$ uniformly, where $n = |y|$.
2. Choose j from $\{-2, -1, \ldots, \lfloor \log l(n) \rfloor\}$ with probability $2^{-(3+j)}$.
3. Call **Inv**$'$ from Corollary 1 with input y, n and $h = \lceil \log \frac{p_2(n) l(n)^2}{2^{2j}(p_2(n) - p_1(n))^2} \rceil$ providing P^i as the predictor. Return the output from $\mathbf{Inv}'^{P_i}$.

Fig. 4. The inverter **Inv**$''$

For nearly one-sided tests the reduction is an improvement with approximately a factor of $p_2(n)^{-1}$ over the previous most efficient reduction. The improvement is obtained by transforming nearly one-sided tests to low rate predictors and applying Theorem 2 that works well for that type of predictors.

Note that the running time of **Inv**$''$ only depends upon $p_1(n)$ and $p_2(n)$ logarithmically. Assuming $t_f(n) \cdot l(n) \leq t_D(n)$ and ignoring logarithmic factors the time-success ratio is $p_2(n) l(n)^2 t_D(n)(p_2(n) - p_1(n))^{-2} \cdot O(n^2)$. Furthermore, if we assume that $p_1(n) \leq c \cdot p_2(n)$ for some constant $c < 1$ the time-success ratio is $l(n)^2 t_D(n) p_2(n)^{-1} \cdot O(n^2)$, still ignoring logarithmic factors.

The correctness of the theorem is shown by defining a number of hybrid distributions H_n^i, connecting the uniform distribution $U_{l(n)}$ with the generator output $G_n^{f,l}$. In Fig. 3 a predictor is defined for each such hybrid. The inverter **Inv**$''$ uniformly selects between these predictors and calls **Inv**$'$ from Corollary 1 providing the selected predictor as an oracle. A proof of the theorem can be found in the journal version of this work [14].

8 Applications

Using pseudo-random material instead of real random material in probabilistic algorithms may be convenient for many reasons. A system can have problems obtaining enough random material or the results perhaps need to be reproducible without storing all random material used.

An important use of pseudo-random material can be found in many implementations of cryptographic primitives. The security definitions of these primitives often express either the need of authentication or that of confidentiality. In the case of authentication there is often a natural nearly one-sided test corresponding to an attacker of the protocol, but in the case of confidentiality this is not always so.

Here we consider the security consequences of using pseudo-random material from a PRBG instead of true random bits in a cryptographic system. We assume that the system that uses true randomness is secure and want to establish that the corresponding system is also secure. The standard method for showing this is done by assuming that there is a successful attacker, when using pseudo-random material in the system, and transforming this attacker into a statistical test distinguishing between output from the PRBG and the uniform distribution. This statistical test will become nearly one-sided if the successful attacker produces a certain type of breakings which we call verifiable breakings. If the PRBG is BMGL then the proof of security can prosper by our more efficient reduction established in Theorem 3. With a verifiable breaking of a cryptographic system we mean that a successful breaking can be verified, possibly with the help of secrets lying in the system such as secret keys.

8.1 Signature schemes

As an example of how natural nearly one-sided tests come up in the case of authentication, we take a closer look at signature schemes. Using the standard definition of security in an adaptive chosen message attack environment [13], an attacker may query an oracle for signatures of any messages of its choice. The attacker is considered successful if it, on a verification key as input, can output a valid signature on any message. Furthermore, the signature should of course not be identical to a signature returned by the oracle. If there is an attacker (from a certain group of attackers) that has more than a negligible probability to break the signature scheme, then the signature scheme is considered to be insecure against that group of attackers. Now assume that S is a secure signature scheme but S^G is not, where the only difference between S^G and S is that S^G uses bits from a pseudo-random generator G when S uses true random bits. Then there is an adversary A that can break the signature scheme S^G, but not S, with a non-negligible probability. This adversary can easily be transformed into a nearly one-sided statistical test D, distinguishing between the output of G and the uniform distribution, by first simulating S, using its input as the source of randomness, and then attacking S using A.

8.2 Encryption schemes

A natural way to define the security of an encryption system is to say that, given a ciphertext, it should be infeasible to produce the corresponding plaintext. If we would adopt this as our security definition, an attacker A could easily be turned into a natural nearly one-sided test that would distinguish between encryptions made by a secure encryption scheme and encryptions made by an encryption scheme that A attacks successfully. In particular this means that if we exchange the true random material that is used in a secure encryption scheme against bits produced by a generator, then a successful attacker on the resulting scheme can be turned into a nearly one-sided test distinguishing between the uniform distribution and the generator output.

Unfortunately, this natural way of defining security is not strict enough for all situations. In general we do not permit an attacker to be able to conclude anything at all about the plaintext, by looking at the ciphertext. This notion was captured by the definition of semantic security in [13]. Using this definition there is no natural way to transform an attacker into a nearly one-sided test. For example, if an adversary could predict, by looking at a ciphertext, the i'th bit of the plaintext (that was drawn from the uniform distribution) with probability $\frac{1}{2} + \varepsilon$, where ε is non-negligible, this would render the system insecure. But this adversary corresponds with a predictor with rate one, and thus not to a nearly one-sided test.

8.3 Other applications

In many algorithms, pseudo-randomness is provided by a pseudo-random function. A well-known construction of pseudo-random functions is the GGM construction [9] which uses a pseudo-random generator as a building block. We note that a nearly one-sided statistical test, distinguishing between the use of real random functions and a family of GGM-functions, can be transformed into a nearly one-sided statistical test distinguishing the uniform distribution and the underlying generator output. Briefly, this is so because the hybrid arguments used in the security proof of GGM will uphold the one-sidedness in distinguishing between real random functions and GGM-functions.

For some applications in simulations, nearly one-sided tests will occur naturally. Consider a probabilistic algorithm for a decision problem that errs with very small probability. Assume that this algorithm uses the output from a PRBG as its source of randomness and that this causes the algorithm to err with substantially higher probability. Then this algorithm can be used to produce a nearly one-sided test distinguishing the PRBG output and the uniform distribution.

9 Open Problems

In this work we have studied nearly one-sided test for the Goldreich-Levin hard-core bit. A natural extension would be to consider what impact nearly one-sided tests have on other hard-core predicates. Additional applications where nearly one-sided tests occur could also be investigated.

Acknowledgments

I am very grateful to Johan Håstad for providing me with good ideas and continuous support. I am also grateful to the referee of Journal of Cryptology for valuable advice about the presentation as well as for suggesting interesting extensions. Finally, I would like to thank Yevgeniy Dodis, Mikael Goldmann, Oded Goldreich, Jonas Holmerin, Åsa Karsberg, Mats Näslund and the anonymous referees for much appreciated help and comments.

References

1. M. Adcock and R. Cleve: *A Quantum Goldreich-Levin Theorem with Cryptographic Applications.* Proceedings, STACS 2002, LNCS 2285, 2002, pp. 323-334, Springer-Verlag.
2. M. Bellare: *Practice-oriented provable-security.* Proceedings, ISW '97, LNCS 1396, 1997, pp. 221-231, Springer-Verlag.
3. M. Bellare and P. Rogaway: *The exact security of digital signatures: How to sign with RSA and Rabin.* Proceedings, EUROCRYPT '96, LNCS 1070, 1996, pp. 399–416, Springer-Verlag.
4. L. Blum, M. Blum and M. Shub: *A Simple Unpredictable Pseudo-Random Generator.* SIAM Journal on Computing, **15**, no. 2, 1986, pp. 364–383.
5. M. Blum and O. Goldreich: *Towards a Computational Theory of Statistical Tests.* Proceedings, 33rd IEEE FOCS, 1992, pp. 406–416.
6. M. Blum and S. Micali: *How to Generate Cryptographically Strong Sequences of Pseudo-random Bits.* SIAM Journal on Computing, **13**, no. 4, 1984, pp. 850–864.
7. R. Fischlin and C. P. Schnorr: *Stronger Security Proofs for RSA and Rabin Bits.* Journal of Cryptology, **13**, no. 2, 2000, pp. 221–244.
8. O. Goldreich: *Foundations of Cryptography: Basic Tools.* Cambridge U. Press, 2001.
9. O. Goldreich, S. Goldwasser and S. Micali: *How to Construct Random Functions.* JACM, **33**. no. 4, 1986, pp. 792-807.
10. O. Goldreich, R. Impagliazzo, L. A. Levin, R. Venkatesan and D. Zuckerman: *Security Preserving Amplification of Hardness.* Proceedings, 31st IEEE FOCS, 1990, pp. 318-326.
11. O. Goldreich and L. A. Levin: *A Hard Core Predicate for any One Way Function.* Proceedings, 21st ACM STOC, 1989, pp. 25–32.
12. O. Goldreich, R. Rubinfeld, and M. Sudan: *Learning polynomials with queries: The highly noisy case.* SIAM Journal on Discrete Mathematics, **13**, no. 4, 2000, pp. 535-570.
13. S. Goldwasser, S. Micali and R. Rivest: *A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks.* SIAM Journal on Computing, **17**, no. 2, 1988, pp. 281–308.
14. G. Hast: *Nearly One-Sided Tests and the Goldreich-Levin Predicate.* Journal of Cryptology, to appear.
15. J. Håstad and M. Näslund: *Practical Construction and Analysis of Pseudo-Randomness Primitives.* Proceedings, ASIACRYPT 2001, LNCS 2248, 2001, pp. 442–459, Springer-Verlag.
16. A. Herzberg and M. Luby: *Public Randomness in Cryptography.* Proceedings, CRYPTO '92, LNCS 0740, 1992, pp. 421–432, Springer-Verlag.
17. L. A. Levin: *Randomness and Non-determinism.* Journal of Symbolic Logic, **58**, no. 3, 1993, pp. 1102–1103.
18. M. Sudan, L. Trevisan and S. Vadhan: *Pseudorandom generators without the XOR Lemma.* Journal of Computer and System Sciences, **62**, no. 2, 2001, pp. 236–266.
19. A. C. Yao: *Theory and application of trapdoor functions.* Proceedings, 23rd IEEE FOCS, 1982, pp. 80–91.