Efficient Privatization of Random Bits

Marius Zimand
School of Computer & Applied Sciences
Georgia Southwestern State University
Americus, GA 31709 USA

Abstract

The paper investigates the extent to which a public source of random bits can be used to obtain private random bits that can be safely used in cryptographic protocols. This process is called privatization of random bits. We consider the case in which the party privatizing random bits has a small number of private random bits. Using techniques from the theory of pseudo-random generators and finely tailoring them for the specifics of this problem, we show that starting with cn private bits and using a long but public random string, one can produce 2^{dn} random bits that cannot be distinguished (but with exponentially small bias) from real random bits by any adversary circuits of size $2^{0.499n}$.

Keywords: one-way function, pseudo-random generator, random bits.

1 Introduction

It is commonly accepted that random bits are a valuable computational resource. Unfortunately, random bits are hard and expensive to produce. Generating them by special-purpose devices such as Geiger counters or Zener diodes is slow and the outcomes must still be further processed (see [SV86] and [Blu84]). One solution to reduce the costs of getting good random bits is to share the random source. We can think of a scenario where there is a public agent that provides random bits to anyone requesting them. It is conceivable that this is more practical than connecting a Geiger counter to your PC. However, in some (in fact, in most) cryptographic protocols one needs private random bits, i.e., bits that cannot be predicted by the adversary. Consider for example the perfect encryption system, the one-time pad, that was invented in 1917 by Major Joseph Mauborgne and Gilbert Vernam [Kah67]. In this system both the sender and the receiver have a common key consisting of a long string of random bits, called the one-time pad. Encryption is done by XORing bitwise the message with the one-time pad and decryption is done by XORing the ciphertext with the one-time pad. While the system is provably resistant to any cryptanalysis attack, it has the drawback of requiring a key that is as long as the message and this creates big problems with key distribution and storage. These problems would disappear if there was a public agent distributing random one-time pads from which the legitimate sender and receiver could extract strings that look random to any adversary of some significant computational power. Thus, the problem reduces to *privatizing random bits*, i.e., to obtaining private random bits out of a public source.

In this work we investigate theoretical aspects related to the privatization of random bits. We do not address here the issue of producing the public random string. We just assume that there is an agent capable of a significant one-time effort targeted at producing a long high-quality

random string. Some speculations about possible practical variants of the algorithm are briefly discussed in Section 6. In this respect, it is noteworthy that George Marsaglia has produced a 600MB high quality random string that is publicly available on a CD-ROM [***96a] or from the internet [***96b].

Clearly the party ("we") that wants to obtain private random bits must have some advantage over the adversary ("they"), as otherwise "they" could simply reproduce the operations performed by "us." For example, one such advantage is for "us" to possess more computational power than "they" but this is not a sound assumption in a cryptographical context. The type of advantage that we consider in this work is much more interesting for cryptographical applications and assumes that "we" possess a small number of private random bits. The theoretical issue on which we focus is that of making the privatization work with an advantage as small as possible.

In this setting the privatization of random bits amounts to building a *pseudo-random generator*, i.e., a function f that takes as input a small number of random bits (which in our context are private) and produces a larger number of bits that look random to the adversary. Again this must hold with high probability over the choices of public random bits. In the presence of a public random string (that can be viewed as an oracle) this seems to be an easy task. "We" could simply use the private string as the address of a block in the public random string and output that block. This seems to be safe, since the adversary does not know what block has been selected. However, this approach (which we dub "naive") is neither theoretically secure, nor practically feasible. Maurer [Mau92] has presented a solution that is provably secure with high probability over the joint distribution of the public string and the private string (a similar solution is presented in [Zim95]). In other words the public source should continuosly produce and distribute long random strings and the legitimate parties ("we") must synchronize perfectly. We present a solution in which a good public random string (for example something similar to Marsaglia's CD-ROM) can be reused forever. Our construction uses $O(n)$ private random bits, $2^{O(n)}$ public random bits and produces up to $2^{\Omega(n)}$ bits that are not distinguishable with a bias better than $2^{-\Omega(n)}$ by any adversary circuit of size 2^{an}, where $a < 1/2$. Moreover, every bit of the output is produced in $n^{O(1)}$ time. This holds with probability over the choices of public random bits larger than $1 - 2^{-\Omega(n)}$. If we assume that "they" have polynomial-time computational capabilities, in order to produce an output of length m, the construction uses a private string of length $(\log m)^{1+\epsilon}$. Since a private string of length $c \log m$ is clearly not enough against a polynomial-time adversary, it follows that our solution is almost optimal with respect to the length of the private key.

The construction is done by transforming a function built by Impagliazzo [Imp96], which is strongly one-way with respect to a random oracle and is safe against non-uniform adversaries, into a randomized pseudo-random generator. Impagliazzo inferred the existence of such a randomized pseudo-random generator from the general result of Håstad, Impagliazzo, Levin, and Luby [HILL91] that shows that the existence of one-way functions is equivalent to the existence of pseudo-random generators. Our main concern here is the efficiency of the privatization process with respect to the length of the private string. We present a construction that is significantly more efficient than the construction implied by [HILL91]. Thus, the algorithm presented here uses cn private random bits and produces an output of length 2^{dn} for some constants c and d, while the general method requires order of n^2 private random bits. Moreover, by using finely tuned extractors (rather than just universal hashing functions), the constant c is small, being approximately equal to 3 (and it can be reduced to approximately 2).

We recall that Σ^* is the set of finite binary strings and Σ^n is the set of binary strings of length n; if $x \in \Sigma^*$, then $|x|$ is the length of string x; $x(i)$ denotes the i-th bit of x; if S is a set,

then $|S|$ is the cardinality of S; and, finally, if y is a real number, then $|y|$ is the modulus of y. The notation $Prob_{x\in_D X}(A)$ represents the probability of A when x is randomly chosen in X according to a distribution D. Sometimes x, X, or D are clear from the context and are therefore omitted.

2 Formal Framework

The access to a public source of random bits is formalized by using functional oracles. That is the machine computing the pseudo-random generator is an oracle machine and we stipulate that the oracle is a function R such that if x is on the query tape and the machine enters a query state, then $R(x)$ is immediately provided on some specially designated answer tape. This models both the situation when the request to the public-server is done on-line and the situation in which the random bits are transferred in a precomputation phase and stored, say, on the hard-disk. The function R consists in fact of a family of functions $(R_n)_{n\in\mathbf{N}}$, where $R_n : \Sigma^{i(n)} \to \Sigma^{m(n)}$. We stipulate that the machine that privatizes the bits taken from the server asks on inputs of length n only queries of length $i(n)$. The probabilistic space on inputs of length n is denoted by $\mathcal{R}_n$ and consists of the set of all functions R_n as above and the uniform distribution on this set (note that $i(n)$ and $m(n)$ are the same for all functions in $\mathcal{R}_n$). Such a function R_n can be encoded in $2^{i(n)}m(n)$ bits, and if the oracle machine M works with R_n as its oracle on inputs of length n, we say that M uses $2^{i(n)}m(n)$ public random bits.
A pseudo-random generator is a function f that maps short strings to longer strings such that any adversary of some significant computational power cannot distinguish with a good bias between the uniform distribution on the set of long strings and the distribution induced by f when its input is uniform randomly selected in the set of short strings. Thus the function f takes as input short random strings and outputs long strings that look random to the adversary. In our setting, the pseudo-random generator takes as input a short private random string and, using a public source of random bits, produces a long string that looks random to the adversary even though this one knows the public string that has been used.
In order to make the definition below more clear, let us recapitulate the ideas leading to the definition of a pseudo-random generator. We consider families of distributions, also called ensembles, $X = (X_n)_{n\in\mathbf{N}}$, where each X_n is a distribution on Σ^n. The statistical difference between two ensembles X and Y is defined by $\Delta(X_n, Y_n) = \sum_{\alpha\in\Sigma^n} \mid Prob(X_n = \alpha) - Prob(Y_n = \alpha) \mid$. Ideally, we would like that the pseudo-random generator $G : \Sigma^s \to \Sigma^l$ induces a distribution $(G(x))_{x\in\Sigma^s}$ that is ϵ-close to the uniform distribution U_l on Σ^l for some small ϵ, i.e., that has $\Delta(G(x), U_l) \leq \epsilon$. If this were possible, then an adversary of arbitrarily large size could not distinguish between the two distributions with a bias greater than ϵ, simply because there is no such statistical difference between the two distributions. It is easy to see that if $s < l$ this is not possible for $\epsilon = o(1)$. However, the two distributions could still be computationally indistinguishable. Let us consider for two ensembles X and Y, $d(X_n, Y_n) = \max_{A\subseteq\Sigma^n} \mid Prob_{X_n}(A) - Prob_{Y_n}(A) \mid$. It is not difficult to see that $d(X_n, Y_n) = 1/2\Delta(X_n, Y_n)$. Now, if an adversary circuit C cannot find a set $A \subseteq \Sigma^n$ such that $\mid Prob_{X_n}(A) - Prob_{Y_n}(A) \mid > \epsilon$, then for him the two distributions behave as if they had a statistical difference bounded from above by 2ϵ. Therefore, we say that $G : \Sigma^{short(n)} \to \Sigma^{long(n)}$ is a pseudo-random generator with security (ϵ, μ) and expansion factor $long(n) - short(n)$, if for any circuit C of size $\mu(n)$,

$$\mid Prob_{x\in\Sigma^{short(n)}}(C(G(x)) = 1) - Prob_{y\in\Sigma^{long(n)}}(C(y) = 1) \mid \leq \epsilon(n).$$

The randomized version follows the same ideas with the modification that the above relation is only required to hold with high probability.

Definition 2.1 *Let $\epsilon,\ \mu,\ \delta : \mathbf{N} \to \mathbf{R}$, $long, short : \mathbf{N} \to \mathbf{N}$, M be an oracle machine, and f_R be the function computed by the machine M with oracle R. The function computed by M is a randomized pseudo-random generator that has security (ϵ, μ) with probability δ and expansion $long(n) - short(n)$ if (a) M runs in polynomial time on all inputs and with all oracles, (b) f_R on inputs of length $short(n)$ produces outputs of length $long(n)$ for all R, and (c) with probability of R in $\mathcal{R}_n$ at least $\delta(n)$ it holds that for any family $C = (C_n)_{n\in\mathbf{N}}$ of oracle circuits of size less than $\mu(n)$, the following relation is true for $n \in \mathbf{N}$ sufficiently large:*

$$\mid Prob_{x\in\Sigma^{short(n)}}(C^R_{long(n)}(f_R(x)) = 1) - Prob_{y\in\Sigma^{long(n)}}(C^R_{long(n)}(y) = 1) \mid \leq \epsilon(n).$$

C^R_n denotes the fact that the oracle circuit C_n runs with the function R as the function oracle (i.e., if the the input bits at an oracle gate form the string x, then the string $R(x)$ is produced as the output bits of the oracle gate.).

3 The naive approach

The naive approach consists of simply taking at random a block from the global random string. "At random" means that the address of the block is given by the local random string. Let us first observe that this method is not practically feasible. In order to preclude an exhaustive search of the private string, the current recommendations require a private string of length at least 64 (and the tendency is to view this length as unsecure nowadays). Since the private key is an address in the public string, this implies that the public key should be 2^{64} bits long. Clearly, producing and distributing such a long string is not doable in the real world. The naive method is not satisfactory at the theoretical level either as it seems to do the job only in the case in which the adversary does not have access to the public server. Indeed, let us suppose that we want to produce random bits of length m that look private to adversaries that have the computational power of polynomial-size circuits. Let the global random string R consist of n blocks of length m, where $n = (1 + \ln(1/\delta))/(2\epsilon^2)$. Let $Y(r, R)$ be the r-th block in R. Note that $|r| = \lfloor \log n \rfloor + 1$. The key point is that for almost all R, the distribution of $Y(r, R)$ (with R fixed and r uniformly at random selected among the strings of length $\lfloor \log n \rfloor + 1$) cannot be distinguished with enough bias from the uniform distribution on strings of length m by circuits of size polynomial in m. Indeed, let C be a circuit of size *size*. We compare $Prob_{y\in\Sigma^m}(C(y) = 1))$ and $Prob_{r\in\Sigma^{\lfloor \log n \rfloor+1}}(C(Y(r, R) = 1)$. Let $\nu = Prob_{y\in\Sigma^m}(C(y) = 1)$ and $\xi_i = 1$ if $C(Y(r_i, R)) = 1$ and 0, otherwise, where $i = 1, \ldots, n$. The expected value over R of each ξ_i is ν. So, by Chernoff bounds

$$Prob(|\nu - \frac{1}{n}\sum_{i=1}^{n} \xi_i| > \epsilon) \leq 2e^{-2\epsilon^2 n} = 2e^{-(1+\ln(1/\delta))} < \delta.$$

Thus, the probability that C can distinguish $Y(r, R)$ from the uniform distribution is less than δ. There are less than $(4size^2)^{size}$ circuits C of size *size*. Therefore, if we take $\delta = 2^{-2^{\log^2 m}}$ and $\epsilon = 2^{-\log^2 m}$, we get that with probability greater than $1 - 2^{-q(m)}$, where q is an arbitrary polynomial, for all circuits C of size polynomial in m,

$$|Prob_{y\in\Sigma^m}(C(y) = 1) - Prob_{r\in\Sigma^{\lfloor \log n \rfloor+1}}(C(Y(r, R) = 1)| \leq 2^{-\log^2 m}.$$

Observe, that by using a local random string r of length $\approx \log n = O(\log^2 m)$ and a public random string R of length nm, we have produced a string $Y(r, R)$ of length m that can be used as a private random string against adversary circuits of size polynomial in m. The naive approach is indeed simple, but the above proof dose not work for the general case in which the adversaries can see the global random string R. One problem is that the number of functions computed by such circuits with various R's is much larger and the proof breaks down.

4 Some technical tools

Let X and Y be two distributions on the same sample space S. X and Y are computationally ϵ-close for circuits of size s, if for any circuit C of size s,

$$\Delta^s_{comput}(X,Y) = |Prob_{x\in_X S}(C(x)=1) - Prob_{y\in_Y S}(C(y)=1)| \leq \epsilon.$$

When the size s is clear from the context, we drop the superscript s. This concept is extended in the natural way to the case in which X and Y are ensembles of distributions and C is a family of circuits.
Let D be a distribution on Σ^n. The *maximum mass* of D is defined to be $max-mass(D) = \max_{x\in\Sigma^n} D(x)$. The collision probability of D is $Prob_D(X=Y) = \sum_{x\in\Sigma^n} D^2(x)$, where X and Y are independent random variables having distribution D. The following facts are well known (see the Appendix for the proof).

Lemma 4.1 *Let D be a distribution on Σ^n with collision probability at most $2^{-n}(1+\epsilon^2)$. Then D is ϵ-close to the uniform distribution on Σ^n.*

Lemma 4.2 *Let D be a distribution on Σ^n with $max-mass(D) = p/2^n$. Then the collision probability of D is upper bounded by $p/2^n$.*

At some point (in Step 3) in our construction we will use extractors. An extractor is a bipartite graph that can be used to reduce the maximum mass of distributions. More precisely, an (n,k,d,m,ϵ)-extractor is a regular bipartite graph having 2^n nodes in the left-hand side and 2^m nodes in the right-hand side, with the degree of each node in the left-hand side equal to d, and such that if node x is randomly chosen in the left-hand side according to a distribution that has maximum mass k and y is uniformly at random chosen among the edges going out from x, the distribution of $E(x,y)$ is ϵ-close to the uniform distribution on Σ^m. ($E(x,y)$ denotes the node on the right-hand side that is reached from x following y.)
Ideally, we would like to be able to effectively and efficiently build extractors. In fact, quite good extractors have been constructed in recent years [TS96], [Zuc96] (see also the survey paper [Nis96]). However, in the best such extractors, the length of y is polylog in the length of x and $1/\epsilon$ and, since we need to have $\epsilon = 2^{-\Omega(|x|)}$, this is too large for our purposes. In our setting, y will be part of the local random string, and thus, we would like to use an as short y as possible. We will achieve $|y| < O(|x|)$ with a small multiplicative constant and ϵ exponentially small in $|x|$, as required above.
The key ingredient in this part of the construction is a special type of extractor, which we call a simple extractor.

Definition 4.3 *Let $G=(V_1,V_2,E)$ be a bipartite regular graph with the left-hand side $V_1=\Sigma^n$, the right-hand side $V_2=\Sigma^m$ and degree $D=2^d$. The edges are represented by the function $E:\Sigma^n\times\Sigma^d\to\Sigma^m$ ($E(a,y)=b$ means that starting from $a\in V_1$ and following the edge $y\in\Sigma^d$ we reach $b\in V_2$). The graph G is a (n,m,d,ϵ)-simple extractor if for all $a,a'\in V_1$ with $a\neq a'$,*

$$Prob_{y\in\Sigma^d}(E(a,y)=E(a',y)) \leq 2^{-m}(1+\epsilon).$$

The terminology is justified by the following result.

Theorem 4.4 *Let $G=(V_1,V_2,E)$ be a (n,m,d,ϵ)-simple extractor. Then the distribution of $(E(x,y),y)$ when x is chosen in V_1 according to a distribution with maximum mass $p/2^n$ and y is uniformly at random chosen in Σ^d, is $\sqrt{\epsilon+\frac{p}{2^{n-m}}}$-close to the uniform distribution. In other words, G is an $(n,p/2^n,d,m,\sqrt{\epsilon+\frac{p}{2^{n-m}}})$ extractor.*

Proof : We first evaluate the collision probability of $(E(x,y),y)$.

$$\begin{aligned} &Prob_{x,y,x',y'}((E(x,y),y)=(E(x',y'),y')) \\ &= Prob_{y,y'}(y=y')Prob_{x,x',y}(E(x,y)=E(x',y)) \\ &= \tfrac{1}{D}(Prob_{x,x'}(x=x')+Prob_{x,x',y}(E(x,y)=E(x',y)\mid x\neq x')). \end{aligned} \tag{1}$$

The first term is bounded by $p/2^n$ (we have used the hypothesis on the maximum mass of the distribution of x and Lemma 4.2). We evaluate the second term.

$$\begin{aligned} &Prob_{x,x',y}(E(x,y)=E(x',y)\mid x\neq x') \\ &= \textstyle\sum_{u\neq u'} Prob_y(E(u,y)=E(u',y))Prob_{x,x'}(x=u \text{ and } x'=u' \mid x\neq x') \\ &\leq (1+\epsilon)2^{-m}\textstyle\sum_{u\neq u'} Prob_{x,x'}(x=u \text{ and } x'=u'\mid x\neq x') = (1+\epsilon)2^{-m}. \end{aligned}$$

We have used the fact that G is an (n,m,d,ϵ)-simple extractor. It follows that equation (1) is bounded from above by

$$\frac{1}{D}(\frac{p}{2^n}+(1+\epsilon)2^{-m}) = \frac{1}{D\cdot 2^m}(1+(\frac{p}{2^{n-m}}+\epsilon)).$$

Taking into account Lemma 4.1, the conclusion follows. ∎

It turns out that a random regular bipartite graph is with high probability a simple extractor. More precisely, the following theorem holds.

Theorem 4.5 *Let $G=(V_1,V_2,E)$ be a random regular bipartite graph with the left-hand side $V_1=\Sigma^n$, the right-hand side $V_2=\Sigma^m$, and degree $D=2^d=9n2^m1/(\epsilon^2)$. Then with probability of G at least $1-2^{-n}$, G is an (n,m,d,ϵ)-simple extractor.*

Proof : For fixed $a,a'\in\Sigma^n$, with $a\neq a'$, and for each $y\in\Sigma^d$, let X_y be 1, if $E(a,y)=E(a',y)$, and 0, otherwise. Clearly, $Prob(X_y=1)=2^{-m}$ and the random variables X_y are independent. By Chernoff bounds, $Prob_G((\sum X_y)/D\geq 2^{-m}(1+\epsilon))\leq e^{-(1/3)\cdot\epsilon^2\cdot D\cdot 2^{-m}} < 2^{-3n-1}$. Thus, the fraction of edges y such that $E(a,y)=E(a',y)$ is $\geq 2^m(1+\epsilon)$, only with probability of G less than 2^{-3n-1}. The probability that there is a pair a,a' as above is less than $2^{-(n+1)}$. The conclusion follows. ∎

Thus simple extractors are extractors and, as in the case of extractors, a random graph is a simple extractor. One big advantage of simple extractors is that checking whether a bipartite graph is a simple extractor can be done in polynomial time, whereas the same operation for extractors is NP-complete [Zim97]. Thus in practice it is not too difficult to build good extractors. One merely generates randomly a graph as in Theorem 4.5 and then checks if it is a simple extractor.

Moreover, for the combination of parameters that we need, we can construct deterministically a simple extractor in a very simple and efficient way. Namely, let n,m be two integers such that $n=2m$. We construct a bipartite graph $G=(V_1,V_2,E)$ as follows. We take $V_1=\Sigma^n$ and $V_2=\Sigma^m$. We also identify V_2 with the field $GF(2^m)$. Each $a\in V_1$ is viewed as the linear function $p_a: GF(2^m)\to GF(2^m)$ defined by $p_a(y)=cy+d$, where $a=cd$ and $|c|=|d|=n/2=m$. Each node $a\in V_1$ has degree $D=2^m$ and is connected to $E(a,y_1)=p_a(y_1),\ldots,E(a,y_{2^m})=p_a(y_{2^m})$, where $y_1,\ldots,y_{2^m}$ are all the elements in $\Sigma^m=GF(2^m)$.

Lemma 4.6 *The above graph is an $(n,m,m,0)$-simple extractor.*

Proof : Since two polynomials of degree at most one intersect in at most one point, for all $a\neq a'\in V_1$,

$$Prob(E(a,y)=E(a',y))\leq\frac{1}{2^m}.$$

∎

5 Construction of the randomized pseudo-random generator

The construction consists of five steps. Since the last two steps do not involve the use of the private random string and since they closely follow well-known techniques, we focus mainly on the first three steps.

Step 1 (The goal of this step is the construction of a randomized one-way function) Let $\mathcal{R}$ be the set of all functions $R : \Sigma^n \to \Sigma^n$. For each $R \in \mathcal{R}$, let

$$f_R(x) = R(x).$$

Impagliazzo [Imp96] has shown that for most R f_R behaves like a one-way function. More precisely, let $q(n)$ be an arbitrary polynomial and $size_0$ be a bound on the size of adversary circuits with $size_0 \leq 2^{an}$, where $a < 1/2$. Then, with probability of R in $\mathcal{R}$ at least $1 - 2^{-q(n)}$

(a) All strings in Σ^n have at most $p = 2eq(n) + n$ preimages under f_R ($e = 2.71\ldots$ is the Euler constant).

(b) For $t = 2size_0 \log 2size_0 + 2q(n)$, there are at least $2^n - 2e^2 p \cdot size_0 \cdot t$ strings x in Σ^n that map via f_R into strings that are noninvertible by any circuit C of size $size_0$, i.e., $f_R(C^R(f_R(x))) \neq f_R(x)$. Let $c_0 > 0$ be such that $2^{-c_0 n} = (2^n - 2e^2 p \cdot size_0 \cdot t)/2^n$.

(c) Let $g_R(x, s) = (f_R(x), s)$, where $|x| = n$ and $s = 2n$. We partition Σ^{3n} into $K = 2^{(3-2b)n}$ segments each having 2^{2bn} elements (b is a constant that will be specified later and that is less than 0.25). Then for each $i = 1, \ldots, K$,

$$Prob_{x \in \Sigma^n, s \in \Sigma^{2n}}(g_R(x, s) \text{ is in the } i\text{-th segment}) \leq \frac{1}{K} + \frac{1}{K^{1+b/3}},$$

where b is a positive constant that will be specified in Step 2. The proof of the above statements is provided in the Appendix. Let $\mathcal{R}_0$ be the subset of $\mathcal{R}$ consisting of the elements R that satisfy (a), (b) and (c).

Step 2 (The randomized one-way function is expanded with some hidden bits.) Let $g_R(x, s) = (f_R(x), s)$, where $|x| = n$ and $|s| = 2n$. Let $b_i(x, s)$ be the inner product modulo 2 of x and $(s_i, \ldots, s_{i+n-1})$, where s_i is the i-th bit of s. Let

$$l = 1 + bn,$$

where b is a constant that will be specified later, and define

$$b(x, s) = (b_1(x, s), b_2(x, s), \ldots, b_l(x, s)).$$

Then by the results of Goldreich and Levin [GL89], the function $b(x, s)$ provides hidden bits for $g_R(x, s)$. More precisely, there are positive constants $b < 0.25$ and c_1 such that for all $R \in \mathcal{R}_0$, $(g_R(x, s), b(x, s))$ and $(g_R(x, s), y)$ are $2^{-c_1 n}$ computationally close for circuits of size $size_1 = size_0 \cdot 2^{-(2/3)c_0 n}$ working with oracle R, when x, s and y are chosen at random in Σ^n, Σ^{2n}, and respectively in Σ^l. The constant c_1 can be taken such that $2^{c_1 n} = 2^{-(1/3)c_0 n} \cdot 2^l \cdot (5n)^{1/3}$.

Step 3 (We apply an extractor to g_R to obtain a randomized pseudo-random generator that expands its input by one bit.) From (a) in Step 1, it follows that, when x is randomly chosen

in Σ^n and s is chosen randomly in Σ^{2n}, the maximum mass of $g_R(x,s)$ is $p/2^{3n}$. We construct a random regular bipartite graph H with $V_1 = \Sigma^{3n}$ and $V_2 = \Sigma^{(3-b)n}$ as follows. We partition V_1 into $K = 2^{(3-2b)n}$ segments of equal cardinality that we call $V_{1,1}, \ldots, V_{1,K}$. We do the same with V_2 obtaining $V_{2,1}, \ldots, V_{2,K}$. Note that for all i, $|V_{1,i}| = |V_{2,i}|^2 = 2^{2bn}$. H will only have edges between $V_{1,i}$ and $V_{2,i}$ for $i = 1, \ldots, K$. More precisely $V_{1,i}$ and $V_{2,i}$ are connected as in Lemma 4.6. Thus H consists of K copies of a $(2bn, bn, bn, 0)$-simple extractor. Taking advantage of the fact that each segment $V_{1,i}$ has, according to the distribution $g_R(x,s)$, probability at most $K^{-1} + K^{-(1+b/3)}$, we show that when x, s and y are randomly chosen in Σ^n, Σ^{2n}, and respectively Σ^{bn}, $(E(g_R(x,s),y),y)$ is $2^{-c_2 n}$ close to the uniform distribution for some constant c_2.

Lemma 5.1 *Let R be in $\mathcal{R}_0$. The distribution $(E(g_R(x,s),y),y)$ is $2^{-c_2 n}$ close to the uniform distribution for some constant c_2, when x, s and y are randomly chosen in Σ^n, Σ^{2n}, and respectively Σ^{bn} .*

Proof : Let w be a string in Σ^{3n} randomly chosen according to the distribution $g_R(x,s)$ defined at Step 2. We estimate the collision probability of the distribution $(E(w,y),y)$.

$$\begin{array}{l} Prob_{w,y,w',y'}((E(w,y),y) = (E(w',y'),y')) = \\ Prob(y = y') \cdot Prob_{w,w',y}(E(w,y) = E(w',y)) = \\ \frac{1}{D}(Prob(w = w') + Prob_{w,w',y}(E(w,y) = E(w',y) \mid w \neq w')). \end{array} \quad (2)$$

The first term in the above sum is bounded from above by $p/2^{3n}$. The second term, denoted A, is equal to

$$A = \sum_{i=1}^{K} Prob_{w,w',y}(E(w,y) = E(w',y) \text{ and } w, w' \in segment\ i \mid w \neq w').$$

Next,

$$\begin{array}{l} Prob_{w,w',y}(E(w,y) = E(w',y) \text{ and } w, w' \in segment\ i \mid w \neq w') = \\ \sum_{u \neq u', u, u' \in segment\ i} Prob_y(E(u,y) = E(u',y)) \cdot Prob(w = u \text{ and } w' = u' \mid w \neq w') \leq \\ 2^{-bn} \cdot \sum_{u \neq u', u, u' \in segment\ i} Prob(w = u \text{ and } w' = u' \mid w \neq w'). \end{array}$$

We have used the fact the i-th segments of V_1 and V_2 form a $(2bn, bn, bn, 0)$-simple extractor. Since

$$Prob(w \neq w') \geq 1 - \frac{p}{2^{3n}},$$

and

$$\sum_{u \in segment\ i} (Prob(w = u))^2 \leq (Prob(w \in segment\ i))^2 \leq (\frac{1}{K} + \frac{1}{K^{1+b/3}})^2,$$

we deduce that $Prob_{w,w',y}(E(w,y) = E(w',y) \text{ and } w, w' \in segment\ i \mid w \neq w')$ is bounded from above by

$$2^{-bn} \cdot (1 + \frac{p}{2^{3n} - p})(\frac{1}{K} + \frac{1}{K^{1+b/3}})^2.$$

Thus, A is bounded from above by

$$2^{-bn} \cdot (1 + \frac{p}{2^{3n} - p})(\frac{1}{K} + \frac{2}{K^{1+b/3}} + \frac{1}{K^{1+(2b)/3}}).$$

It follows that the relation from equation (2) is bounded from above by

$$\frac{1}{D \cdot 2^{bn} \cdot K}(1 + 2^{-dn})$$

for some constant d. Taking into account Lemma 4.1 the conclusion follows. ∎
Now take $G_R(x,s,y) = (E(g_R(x,s),y), y, b_R(x))$. We have that for circuits of size $size_1$,

$$\Delta_{comput}(G_R(x,s,y), u_{(3+b)n+1})$$
$$\leq \Delta_{comput}((g_R(x,s),b(x)),(g_R(x,s),u_l) + \Delta_{comput}((E(g_R(x,s),y),y),u_{3n-bn+|y|}),$$

where u_j, for various values of j, denotes an element of length j chosen according to the uniform distribution.
The first term is bounded by $2^{-c_1 n}$, and the second term is bounded by $2^{-c_2 n}$. Thus, there is some constant c_3 such that for all $R \in \mathcal{R}_0$, $G_R(x,s,y)$ is $2^{-c_3 n}$ computationally close to the uniform distribution. Observe that $|x| + |s| + |y| \leq 3.25n$ and that G_R outputs a string that is one bit longer than its input. For simplicity, we assume that $|x| + |s| + |y| = 3.25n$.

Step 4 (Double the extension of the randomized pseudo-random generator.) We define $I_R : \Sigma^{3.25n} \to \Sigma^{2\cdot 3.25n}$ by

$$I_R(x) = (s_1, s_2, \ldots, s_{2|x|}),$$

where $s_1, \ldots, s_{2|x|}$ are bits defined inductively as follows: $x_0 = x$ and for $i = 1, \ldots, 2|x|$,

$$s_i = \text{ the first bit of } G_R(x_{i-1}) \text{ and}$$
$$x_i = \text{ the last } |x| \text{ bits of } G_R(x_{i-1}).$$

Again, by an application of the hybrid method, there exists a positive constant c_4 such that, for all $R \in \mathcal{R}_0$, $I_R(X)$ is $2^{-c_4 n}$ close to the uniform distribution for circuits of size $size_2 = size_1 - 4(3.25n)^2 \cdot t_G(3.25n)$, where X is the uniform distribution on $\Sigma^{3.25n}$ and $t_G(3.25n)$ is the size of a circuit that calculates G_R on inputs of size $3.25n$. The constant c_4 can be taken so that $2^{-c_4 n} = 2^{-c_3 n} 6.5n$.

Step 5 (Get a randomized pseudo-random generator with more expansion) Let $I_0(x)$ and $I_1(x)$ be the first and respectively the second half of the string $I_R(x)$. Let $j = c_5 n$, where c_5 is a constant such that $0 < c_5 < c_4$. Define $F_R : \Sigma^{3.25n} \to \Sigma^{2^{c_5 n}}$ as follows. The $\alpha_1\alpha_2\ldots\alpha_j$ bit of $F_R(x)$ is the first bit of $I_{\alpha_1}(I_{\alpha_2}(\ldots(I_{\alpha_j}(x))\ldots)$. The techniques of Goldreich, Goldwasser, and Micali [GGM86] show that for $c_6 = c_4 - c_5$, for all $R \in \mathcal{R}_0$, $F_R(X)$ is $2^{-c_6 n}$ computationally close to the uniform distribution for circuits of size $size_3 = size_2/(2^{c_5 n} \cdot t_I(3.25n))$ working with oracle R, where X is the uniform distribution on $\Sigma^{3.25n}$ and $t_I(3.25)$ is the size of a circuit that calculates I_R on inputs of size $3.25n$.
Observe that for all $R \in \mathcal{R}_0$, F_R takes an input of size $3.25n$ and produces an output of size $2^{c_5 n}$ that cannot be distinguished by any circuit of size bounded by 2^{an}, with $a < 1/2$, working with oracle R.

Consequently we have proved:

Theorem 5.2 *For every size of adversaries $\mu(n)$ with $n \leq \mu(n) \leq 2^{an}$ where $a < 1/2$, there exists a randomized pseudo-random generator that has security $(\mu(n), 2^{-c_6 n})$ with probability $1 - 2^{-q(n)}$ and expansion $2^{c_5 n} - 3.25n$, where q is any polynomial and c_5 and c_6 are positive constants. Moreover, every bit of the output can be computed in time polynomial in the length of the private seed.*

Observation 1. At Step 3, a universal family of hash functions could have been used to get about the same effect (see [Gol93] and [Zim96]). The use of extractors reduces the length of the local random string by almost 3 times.

Observation 2. It is important to observe that a good public random string can be reused ad infinitum. On the other hand, the actual queries should be done over a secure channel. One simple way to do this is to copy the whole public random string off-line and to make the queries locally.

Observation 3. If we want the pseudo-random generator to produce a string of length m that is secure against any family of circuits of size polynomial in m then at Step 5 we should take $j = n^\gamma$ with $\gamma < 1$. In this case, the method is using $\log^{1+\epsilon} m$ private random bits and approximately $m^{\log^\epsilon m}$ public random bits, where $\epsilon = (1-\gamma)/\gamma$.

6 Final remarks

As presented, our solution to the privatization problem has no practical significance. By working through the numerous constants that appear in the construction, we have estimated that one should use in Step 1 $n = 1200$ in order to produce a 1GB string that cannot be distinguished with bias at least 2^{-20} from a random string by adversaries capable of doing 2^{90} readings from the public random string. This means that the public random string should be $1200 \cdot 2^{1200}$ bits long! Even if we want to produce a 1MB string, we still need $n \approx 900$.
However, if we sacrifice the provable security, it is plausible that a value of $n \approx 35$ is good enough to produce a long string that looks random to powerful adversaries. If $n = 35$, the private key is approximately 75 bits long and thus the adversary cannot do an exhaustive search. The bits of the public string are mangled in a complicated manner that is dictated by the private string and this seems to preclude a smart search strategy. Of course, one can use any other mixing technique. The current recommendations [ECS94] suggest the use of a keyed hash function (see [MvOV96]) that is based on DES or on a hash function such as MD5. These functions use tables of constants whose randomness is doubtful (in the case of DES) or that are obtained using widely used mathematical functions (sin for MD5). Compared to this approach, the method presented here, scaled down so that to be feasible, does not use any constants and has some theoretical support. Thus, subject to confirmation by further studies, it can be a viable candidate for practical applications.

References

[***96a] ***. Diehard. `http://stat.fsu.edu/~geo/diehard.html`, 1996.

[***96b] ***. Marsaglia's random bits. `ftp://ftp.cs.hku.hk/pub/random`, 1996.

[Blu84] M. Blum. Independent unbiased coin flips from a correlated biased source: A finite state Markov chain. In *Proceedings of the 25th IEEE Symposium on Foundations of Computer Science*, pages 425–433, 1984.

[ECS94] D. Eastlake, S. Crocker, and J. Schiller. RFC 1750 - Randomness requierements for security. Internet Request for Comments 1750, December 1994.

[GGM86] O. Goldreich, S. Goldwasser, and S. Micali. How to construct a random functions. *Journal of the ACM*, 33(4):792–807, 1986.

[GL89] O. Goldreich and L. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the 21st ACM Symposium on Theory of Computing*, pages 25–32, 1989.

[Gol93] O. Goldreich. Foundations of cryptography (fragments of a book), February 1993. ECCC Technical report, available at http://www.eccc.uni-trier.de/local/ECCC-Books/eccc-books.html.

[HILL91] J. Hastad, R. Impagliazzo, L. Levin, and M. Luby. Construction of a pseudo-random generator from any one-way function. Technical Report 91-68, ICSI, Berkeley, 1991.

[Imp96] R. Impagliazzo. Very strong one-way functions and pseudo-random generators exist relative to a random oracle. (manuscript), January 1996.

[Kah67] D. Kahn. *The codebreakers: the story of secret writing.* New York, MacMillan, 1967.

[Mau92] U. Maurer. Conditionally-perfect secrecy and a provably-secure randomized cipher. *Journal of Cryptology*, 5(1):53–66, 1992.

[MvOV96] A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1996.

[Nis96] N. Nisan. Extracting randomness: how and why. A survey. In *Proceedings of the 11th Computational Complexity Conference*, pages 44–58, 1996.

[SV86] M. Santha and U. Vazirani. Generating quasi-random sequences from semi-random sources. *Journal of Computer and System Sciences*, 33:75–87, 1986.

[TS96] A. Ta-Shma. On extracting randomness from weak random sources. In *Proceedings of the 26th ACM Symposium on Theory of Computing*, pages 276–285, 1996.

[Zim95] M. Zimand. On randomized cryptographic primitives. Technical Report 586, Department of Computer Science, Univ. of Rochester, Rochester, NY, 1995.

[Zim96] M. Zimand. How to privatize random bits. Technical Report 616, Department of Computer Science, Univ. of Rochester, Rochester, NY, April 1996.

[Zim97] M. Zimand. Checking if a graph is an extractor is NP-complete. (manuscript), November 1997.

[Zuc96] D. Zuckerman. Randomness-optimal sampling, extractors, and constructive leader election. In *Proceedings of the 26th ACM Symposium on Theory of Computing*, pages 286–295, 1996.

A Appendix

We provide the proofs of Lemma 4.1 and Lemma 4.2 and some more technical details about Step 1 of the construction in Theorem 5.2.

Proof of Lemma 4.1

$$\begin{aligned}
&\textstyle\sum_{x\in\Sigma^n} |D(x) - 2^{-n}| \le \sqrt{2^n}\sqrt{\sum_{x\in\Sigma^n}(D(x)-2^{-n})^2} \quad \text{(Cauchy-Schwartz inequality)}\\
&= \sqrt{2^n}\sqrt{\textstyle\sum_{x\in\Sigma^n} D^2(x) + \sum_{x\in\Sigma^n} 2^{-2n} - 2\cdot 2^{-n}\sum x\in\Sigma^n D(x)}\\
&\le \sqrt{2^n}\sqrt{2^{-n}(1+\epsilon^2) - 2^{-n}} = \epsilon.
\end{aligned}$$

∎

Proof of Lemma 4.2 The collision probability of D is $\sum_{x\in\Sigma^n} D^2(x)$. Since $\sum_{x\in\Sigma^n} D(x) = 1$ and $D(x) \le p/2^n$, this expression is maximized for distributions D allocating $p/2^n$ probability

mass to $2^n/p$ elements in Σ^n and 0 to the rest of the elements. In this case, the collision probability is $2^n/p$. ∎

Step 1 is based on the following facts proven by Impagliazzo [Imp96].

Fact A.1 *With probability of $R \in \mathcal{R}_n$ at least $1 - 2^{-2q(n)}$, no string in Σ^n has more than $2eq(n) + n$ preimages under f_R.*

Proof of Fact A.1. Take $p = 2eq(n) + n$. By Markov's inequality, the probability over $\mathcal{R}$ that a fixed y in Σ^n has p preimages under f_R is at most the expected number of sets A of size p such that all elements in A map to y. The number of these sets is $\binom{2^n}{p}$. Thus the above expected value is $\binom{2^n}{p}2^{-np} < (e2^n/p)^p 2^{-np} = (e/p)^p < (1/2)^{2q(n)+n}$. Summing over all y in Σ^n, the probability that there is one string with more than p preimages is at most $2^{-2q(n)}$. ∎

Let C be an oracle circuit that attempts to invert f_R and let $size \leq 2^{an}$, with $a < 1/2$, be its size. Without loss of generality we can assume that $C^R(y)$ outputs z only if $f_R(z) = y$. We say that C^R inverts y if $f_R(C^R(y)) = y$ and that C^R inverts a set $T \subseteq \Sigma^n$ if it inverts all elements of T.

Fact A.2 *With probability of $R \in \mathcal{R}$ at least $1 - 2^{-(t-2size \log 2size)}$, no oracle circuit C of size $size$ inverts a set with $2e^2 \cdot size \cdot t$ elements.*

Proof of Fact A.2. Let T be a set of size t. The probability that C^R inverts T is at most $\binom{t \cdot size}{t}(t/2^n)^t$. Indeed, the probability that a queried string α satisfies $f_R(\alpha) \in T$ is $t/2^n$. The probability that for all y in T, $C^R(y)$ finds a query α such that $f_R(\alpha) \in T$ is bounded from above by the probability that t questions out of the possible $t \cdot size$ total number of questions are mapped by R into elements in T and this latter probability is $\binom{t \cdot size}{t}(t/2^n)^t$.
Thus, the expected number of sets T of cardinality t that are inverted by C^R is at most $\binom{2^n}{t}\binom{t \cdot size}{t}(t/2^n)^t \leq (e2^n/t)^t (et \cdot size/t)^t (t/2^n)^t = (e^2 \cdot size)^t \stackrel{\text{def}}{=} \mu$.
The probability that a set U of cardinality $u = 2e^2 \cdot size \cdot t$ is inverted by C^R is equal to the probability that all subsets $T \subseteq U$ of cardinality t are inverted. There are $\binom{u}{t} \geq (u/t)^t \stackrel{\text{def}}{=} k$ such sets. By Markov's inequality, the probability that k subsets of cardinality t are inverted is at most $\mu/k = (e^2 \cdot size \cdot t/u)^t = 2^{-t}$.
There are less than $(4 \cdot size^2)^{size}$ oracle circuits of size $size$. Therefore the probability that there is a circuit of size $size$ that inverts a set of cardinality $2e^2 \cdot size \cdot t$ is at most $2^{2size \log 2size} 2^{-t} = 2^{-(t-2size \log 2size)}$. ∎
We can now finalize the proof of statements (a) and (b) from Step 1. Take $t = 2size \log 2size + 2q(n)$. By Fact A.1 and Fact A.2, with probability of $R \in \mathcal{R}_n$ at least $1 - 2^{-(t-2size \log 2size)} - 2^{-2q(n)} \geq 1 - 2^{-q(n)}$, no circuit of size $size$ can invert more than $2e^2 \cdot size \cdot t$ elements and each element in Σ^n has at most $2eq(n) + n$ preimages under f_R. Thus (a) and (b) are proved. For part (c), partition Σ^{3n} into K segments called *segment* $1, \ldots$ *segment* K. We focus on *segment* i and consider the random variables X_y, with $y \in \Sigma^{3n}$ defined by $X_y = 1$ if $g_R(y)$ is in *segment* i, and 0 otherwise. Clearly the expected value of any X_y is $1/K$, and thus, by Chernoff bounds, the probability that the fraction of X_y that are equal to 1 differs from $1/K$ by more than δ is less than

$$2e^{-\frac{2\delta^2 \cdot 2^{3n}}{(1/K)(1-1/K)}}.$$

Taking $\delta = K^{-(1+b/3}$, it follows that the probability that *segment* i has probability more than $K^{-1} + K^{-(1+b/3)}$ is less than 2^{-2^n}. The probability that there is segment with probability more than $K^{-1} + K^{-(1+b/3)}$ is less than $K2^{-2^n}$, and thus, part (c) follows.
∎