Principles of Pseudo-Random Number Generation in Cryptography

Ned Ruggeri

August 26, 2006

1 Introduction

The ability to sample discrete random variables is essential to many areas of cryptography. The most obvious example is key-generation for encryption algorithms or keyed hash functions – if one uses deterministic algorithms to generate keys, then the security of the key is dependent on the secrecy of the algorithm, which is both impractical and unnecessary. Modern cryptography posits that messages and keys are the secrets, not algorithms. Only (well-constructed) algorithms incorporating random keys offer legitimate users abilities that an adversary cannot replicate.[1] Since security should not reside in uncertainty about a party's algorithm, it must instead originate from the entropy of the key-space. However, it is often impractical to generate and transfer very long strings of random bits. A good way to minimize these problems is to use cryptographically secure pseudo-random number generators (CSPRNG).

This paper hopes to be an accessible resource to introduce the principles of pseudo-random number generation in cryptography. Key topics are what it means to be a CSPRNG, the conditions for the existence of a CSPRNG, as well as more general cryptographic concepts such as 'security' and 'adversary.'

2 Preliminaries

Definition: A *discrete variable* $\mathbf{X}$ takes on values in an associated finite set X. For $x \in X$, we denote the (unconditional) probability that $\mathbf{X} = x$ as $\mathbf{Pr}[\mathbf{X} = x]$, or simply $\mathbf{Pr}[x]$ if there is no confusion. Of course, $\sum_{x \in X} \mathbf{Pr}[x] = 1$.

A *discrete random variable* $\mathbf{X}$ is a variable such that, for any event E, the conditional probability $\mathbf{Pr}[\mathbf{X} = x|E] = \mathbf{Pr}[\mathbf{X} = x]$. Put another way, no knowledge of the world gives you any insight into the expected value of a sample of $\mathbf{X}$ (beyond the insight you have from knowing X and the probability distribution on $\mathbf{X}$).

[1] Though there are important unkeyed cryptographic primitives, they do not grant unique abilities to privileged parties (e.g., anyone can hash a message).

If $\mathbf{Pr}[\mathbf{X} = x] = \frac{1}{|X|}$ for all $x \in X$, then we say that $\mathbf{X}$ is a *uniformly distributed random variable*.

By $\mathbf{X}_{\{\mathbf{0},\mathbf{1}\}^{\mathbf{n}}}$, we denote a uniformly distributed random variable over the set of bit-strings of length n. Our discussion will be about $\mathbf{X}_{\{\mathbf{0},\mathbf{1}\}^{\mathbf{n}}}$ since this is the variable of greatest cryptographic interest, though the contents of our statements aren't really specific to any one uniformly distributed random variable. Though the variable is over a set of bit-strings, it often makes sense to think of a single sample of $\mathbf{X}_{\{\mathbf{0},\mathbf{1}\}^{\mathbf{n}}}$ as equivalent to a sequence of n samples from a uniformly distributed random variable over $\{0, 1\}$ (a *coin flip*). □

NB: We do not deal with continuous random variables, which are variables over an infinite set of possible values, since due to memory boundedness there is no way to store this information in the relevant computational model (the further importance of the computational model will become apparent later). Moreover, we assume that (a) random variables exist, (b) we can sample random variables (i.e., assign their current value to some register), and (c) sampling $\mathbf{X}$ takes polynomial-time with respect to $|X|$.

Definition: [Shannon] *Entropy*, denoted $\mathbf{H}(\mathbf{X})$, is a measure of the uncertainty about the value of a future sample of a random variable $\mathbf{X}$. Specifically, $\mathbf{H}(\mathbf{X}) = -\sum_{x \in X} \mathbf{Pr}[x] \log_2 \mathbf{Pr}[x]$ (which is a lower bound on the average number of bits per sample for a Huffman encoding of $\mathbf{X}$), but none of this need concern us. The concept is really what's important – that entropy is a measure of how difficult it is to predict a future (or guess a secret) event, given all other available knowledge about the world. *Conditional entropy*, written as $\mathbf{H}(\mathbf{X}|\mathbf{Y}) = -\sum_{y \in Y} \sum_{x \in X} \mathbf{Pr}[y]\mathbf{Pr}[x|y] \log_2 \mathbf{Pr}[x|y]$, measures how much uncertainty of $\mathbf{X}$ is not due to uncertainty about $\mathbf{Y}$. □

Definition: A *function ensemble* is an indexed family of functions denoted by $f\colon \{0,1\}^{l(n)} \to \{0,1\}^{t(n)}$, where $l(n)$ and $t(n)$ are polynomial in n (for any fixed n, we could refer to a specific function f_n). Of course, computing the value of the function ensemble given an input takes a certain amount of time. We will thus often consider the *worst-case time bound* for f parameterized by n, denoted $T(n)$. A *polynomial-time function ensemble* has $T(n) \leq n^{\mathcal{O}(1)}$.[2]

Most times, for n, m, $l(n) \neq m(n)$, and so there is no ambiguity to treat the function ensemble as a single function f. If this is the case, then we are, essentially, viewing the family as not indexed by n, but by the length of the input. □

Definition: *Cryptographic primitives* are the basic building blocks for protocols. An instance of a primitive is a polynomial-time function ensemble with a security property (e.g., one-way, collision-resistant, etc.). □

Definition: An *adversary* A is a function ensemble which attempts to defeat the security property of an instance of a cryptographic primitive. They are often Monte Carlo algorithms (a randomized algorithm always returning a result, though sometimes inaccurately). The average success-probability (over inputs to the primitive instance) of an adversary depends is written ($\delta(n)$). We refer

[2] If k, l are functions, then $k(x) = \mathcal{O}(l(x))$ iff there exists c such that $k(x) \leq c \cdot l(x)$.

to the *time-success* ratio of A, which is $\frac{T(n)}{\delta(n)}$.[3]

Let S be a function of n. If there is no adversary of a primitive instance f with time-success ratio better than $S(n)$, we say that the security property is $S(n)$-*secure*, or that the *security* of f is $S(n)$. □

It is now obvious why we want to talk about function ensembles rather than simply functions. We want to consider the asymptotic behavior of the difficulty to attack a primitive *relative* to the difficulty to compute it. Perhaps today it is infeasible to attack a primitive instance given current computational limitations. But as computing power increases, an adversary's task will become easier. But if f has security $S(n) > n^{\mathcal{O}(1)}$, while taking only time $T(n) = n^{\mathcal{O}(1)}$ to compute, then we can increase the difficulty of attacking f without a commensurate increase in the difficulty to compute f.

3 Next-Bit Unpredictability

While we assume it takes polynomial-time to sample a random variable, in practice it is expensive to do so. Specialized hardware is necessary. If another party needs to be sent the random information, transmission can pose a serious task. Also, what if extra bits are required after the parties can no longer stay in touch via private channels?

If we could use some deterministic algorithm to 'stretch out' the entropy of a relatively short sequence of random samples (a *seed*), we would not have to generate or transmit as much secret information. We hope that after stretching the seed, the subsequent sequence will still appear to be a series of random samples.

Consider $\mathbf{X}_{\{\mathbf{0},\mathbf{1}\}^{\mathbf{n}}}$, and $x \in \{0,1\}^n$. Since the conditional probability that $\mathbf{X}_{\{\mathbf{0},\mathbf{1}\}^{\mathbf{n}}} = x$ given any event E is still $\mathbf{Pr}[\mathbf{X}_{\{\mathbf{0},\mathbf{1}\}^{\mathbf{n}}} = x] = \frac{1}{2^n}$, we might say that the values of future samples of $\mathbf{X}$ are unpredictable. By assumption, no amount of computing power could possibly find anything out about the value of a single sample of $\mathbf{X}_{\{\mathbf{0},\mathbf{1}\}^{\mathbf{n}}}$.[4] We call this property *next-bit unpredictability*. However, we don't have infinite computational resources anyway. What we would like to know is whether there is a weaker criteria of unpredictability which, given our computing power, suffices for cryptographic purposes.

To do this, consider a polynomial-time, deterministic function ensemble $f\colon \{0,1\}^n \to \{0,1\}^{l(n)}$, with $l(n) > n$. Assume that the input of f is always a sample from $\mathbf{X}_{\{\mathbf{0},\mathbf{1}\}^{\mathbf{n}}}$. We might consider $\mathbf{X}_{\{\mathbf{0},\mathbf{1}\}^{\mathbf{n}}}$ and f to define a variable

[3]The importance of time-success ratio is due to the fact that we can generally trade speed for accuracy. If an adversary A_1 takes time $T_1(n)$ and achieves accuracy $\delta_1(n)$, then we can write another, randomized adversary A_2 which samples a random bit to decide whether to (a) fail or (b) run A_1. $\delta_2(n) = (\delta_1(n))(\mathbf{Pr}[1]) = \mathcal{O}(\delta_1(n)$, while $T_2(n) = (T_1(n))(\mathbf{Pr}[1]) + z = \mathcal{O}((T_1(n)))$ (where z is the time it takes to sample a single bit). Regardless, we see that A_2 may be much faster than A_1, but the time-success ratio is the same (up to a constant): $\mathcal{O}(\frac{T(n)}{\delta_1(n)})$. We can similarly can hope to increase the accuracy of a randomized adversary by running it multiple times and picking the most frequent result.

[4]We can say more about long sequences of samples from $\mathbf{X}_{\{\mathbf{0},\mathbf{1}\}^{\mathbf{n}}}$. We expect half of the 'runs' to be of length 2, an eighth to have length 3, etc.

$\mathbf{F_{l(n)}}$. To sample $\mathbf{F_{l(n)}}$, one simply lets x (the seed) be a sample of $\mathbf{X_{\{0,1\}^n}}$ and computes $f(x)$.

Note that since f is deterministic, f is not onto and thus $\mathbf{F_{l(n)}}$ is not a uniformly distributed variable. Also, even if the seed is completely forgotten, $\mathbf{H(F_{l(n)}) \leq H(X_{\{0,1\}^n})}$. This immediately implies that a sample of $\mathbf{F_{l(n)}}$ is not analogous to flipping a fair coin $l(n)$ times – there is less unknown information about the bits than that. Regardless of whether the entropy is spread out over all the bits evenly, since the entropy per bit is less than 1, for at least one i, $\mathbf{H(x_i|x_{\{j \neq i\}}) < 1}$. Thus, one value of x_i is more likely than another, given the values of the other $l(n) - 1$ bits.

Definition: Fix n. An *i-th bit predictor* p_i for $\mathbf{F_{l(n)}}$, takes the first $i-1$ bits of a sample from $\mathbf{F_{l(n)}}$, and attempts to predict the value of the i-th bit before it is revealed. The success probability of the predictor is

$$\delta_i(n) = \sum_{y \in \mathbf{F_{l(n)}}} \left(\mathbf{Pr}[\mathbf{F_{l(n)}} = y]\right) \left(\mathbf{Pr}[p_i(y_{1,2,\ldots,i-1}) = y_i]\right)$$

(where $y_{1,2,\ldots,i-1}$ means the first $i-1$ bits of y).

A *next-bit predictor* p attempts to predict the next bit given some initial bits of a sample $y \in \mathbf{F_{l(n)}}$. Such a predictor can be seen as calling on a library $\{p_i\}$ of i-th bit predictors ($1 \leq i \leq l(n)$). The success probability of p is

$$\frac{\sum_{i=1}^{l(n)} \delta_p(n)}{l(n)}.$$

□

Theorem 1. *Fix n. For some i, an i-th bit predictor exists for $\mathbf{F_{l(n)}}$ (that is, a predictor with $\delta > 0$).*

Proof. Let y be a sample from $\mathbf{F_{l(n)}}$. Assume a predictor does not exist for any of the first n bits of y. Let us think of these bits as random variables. Then for $1 \leq i \leq n$, $\mathbf{Pr}[\mathbf{y_i} = 1] = \frac{1}{2}$ (if not, we need zero bits to predict $\mathbf{y_i}$). Moreover:

$$\mathbf{Pr}[\mathbf{y_j} = 1 | \mathbf{y_i} = 1] = \frac{1}{2}, \quad 1 \leq i < j \leq n.$$

If this weren't true, than knowing the value of $\mathbf{y_i}$ would let us predict $\mathbf{y_j}$ with

$$\delta_j(n) = \mathbf{Pr}[\mathbf{y_j} = 1 | \mathbf{y_i} = 1].$$

By Bayes' Theorem, we also have that:

$$\mathbf{Pr}[\mathbf{y_i} = 1 | \mathbf{y_j} = 1] = \frac{\mathbf{Pr}[\mathbf{y_i} = 1]\mathbf{Pr}[\mathbf{y_j} = 1 | \mathbf{y_i} = 1]}{\mathbf{Pr}[\mathbf{y_j} = 1]} = \frac{1}{2}.$$

So the first values of the first n bits are independent, which means

$$\mathbf{H}(\mathbf{y_i}, \mathbf{y_i}, \ldots, \mathbf{y_n}) = n = \mathbf{H}(\mathbf{F_{l(n)}})$$

which means that $\mathbf{H(y_{n+1}|x_{1,\ldots,n}) = 0}$, and thus can always be predicted successfully from the first n bits. ■

If we can find a next-bit predictor of $\mathbf{F_{l(n)}}$ with a fairly high level of accuracy, it would make sense to say that f doesn't generate a 'particularly random' variable. While a sample of $\mathbf{F_{l(n)}}$ may have a distribution of ones and zeros which might be expected from $l(n)$ random coin flips, it would diverge from the defining property of a random variable – unpredictability. Of course, we want to consider how long it takes for a next-bit predictor to run. Therefore, we define pseudo-randomness in terms of the lower-bound on the time-success ratio of a next-bit predictor.

Definition: Consider a polynomial-time, deterministic function ensemble $f\colon \{0,1\}^n \to \{0,1\}^{l(n)}$. An adversary A (a sequence of next-bit predictors) attacking f has average success probability

$$\delta(n) = \left(\frac{1}{l(n)\cdot 2^n}\right)\left(\sum_{x\in\{0,1\}^n}\sum_{\{i\in 1,2,\ldots,l(n)\}} \mathbf{Pr}[A(f(x)_{1,\ldots,i-1}) = f(x)_i]\right)$$

for predicting any given bit of the sequence, given the preceding bits.

If the time-success probability of any adversary is at least $S(n)$, we say that f is a *$S(n)$-secure pseudo-random bit-string generator*, which offers the security property that output is *S(n)-secure next-bit unpredictable* when input is chosen from $\mathbf{X_{\{0,1\}^n}}$. The variable $\mathbf{F_{l(n)}}$ is called *$S(n)$ pseudo-random.* □

4 Indistinguishability

We have now finally defined pseudo-randomness in terms of $S(n)$-secure next-bit unpredictability. The variable a $S(n)$-secure pseudo-random bit-string generator defines, from the perspective of an adversary with bounded computational power, has the essential property of a random variable: next-bit unpredictability. Yet, while our approach was intuitive, there is a seemingly distinct conception of pseudo-randomness.

Say we are presented with two variables, $\mathbf{X_0}$ and $\mathbf{X_1}$, over the same set X. Imagine a friend (randomly, fairly, secretly) chooses one of the variables and samples a value from it. He then presents the value to you and asks you to guess whether the value came from $\mathbf{X_0}$ or $\mathbf{X_1}$.

Definition: One idea is to consider all functions (including randomized functions) of the form $d\colon X \to \{0,1\}$. If $d(x) = 0$, we consider this a 'guess' that x came from $\mathbf{X_0}$, otherwise, we take d as guessing x was from $\mathbf{X_1}$. Given an element from $\mathbf{X_i}$, the success probability of d is

$$\sum_{x\in X} \mathbf{Pr}[\mathbf{X_i} = x]\mathbf{Pr}[d(x) = i],$$

and for an element chosen fairly from either set,

$$\delta = \sum_{x\in X} \frac{(\mathbf{Pr}[\mathbf{X_0} = x]\mathbf{Pr}[d(x) = 0] + \mathbf{Pr}[\mathbf{X_1} = x]\mathbf{Pr}[d(x) = 1])}{2}.$$

We say that d is a *distinguisher* with success probability δ. $\square$

This line of inquiry suggests defining a pseudo-random variable differently than we did in the prior section, based on the security property of $S(n)$-*secure indistinguishability*. However, it turns out that the two security properties are basically the same, and thus any primitive instance which offers security for one property offers (basically) identical security for the other.

Lemma 1 (Without proof – Sorry!). *This kind of thing can be found in a statistics text, I guess. We make use of the regularized, incomplete beta function:*

$$I_x(a,b) = \frac{\int_0^x t^{a-1}(1-t)^{b-1}\,dt}{\int_0^1 t^{a-1}(1-t)^{b-1}\,dt}$$

and this fact about the binomial distribution:

$$\sum_{i=0}^{k} \frac{n!(p)^i(1-p)^{(n-i)}}{i!(n-i)!} = I_{(1-p)}(n-k,k+1)$$

Theorem 2. *Consider a pseudo-random bit-string generator f. Suppose there exists a next-bit predictor A for f with time-success ratio $R(n)$. Then there exists a distinguisher of $\mathbf{F_{l(n)}}$ and $\mathbf{X}_{\{\mathbf{0},\mathbf{1}\}^{\mathbf{l(n)}}}$ with time-success ratio at least $\mathcal{O}(R(n))$.*

Proof. Fix n.

Let x be a sample from either $\mathbf{F_{l(n)}}$ or $\mathbf{X}_{\{\mathbf{0},\mathbf{1}\}^{\mathbf{l(n)}}}$. Recall that we can think of A as calling a library of i-th bit predictors p_i. We want to define another adversary A' which will use this library to attack distinguishability of the two variables. Let

$$\beta = \frac{\sum_{i=1}^{l(n)} p_i(x_{1,\ldots,i-1}) \oplus x_i}{l(n)}.$$

If $\beta > \frac{1}{2}$, that means that A was unable to predict more than half the bits, and A' should guess that x had been generated randomly, from $\mathbf{X}_{\{\mathbf{0},\mathbf{1}\}^{\mathbf{l(n)}}}$. Otherwise, A' should guess x is from the pseudo-random variable $\mathbf{F_{l(n)}}$.

If x is a sample from $\mathbf{X}_{\{\mathbf{0},\mathbf{1}\}^{\mathbf{l(n)}}}$, then we know that A (and every next-bit predictor) predicts the next bit with accuracy $\frac{1}{2}$. Thus, the probability that A' correctly guesses x is from $\mathbf{X}_{\{\mathbf{0},\mathbf{1}\}^{\mathbf{l(n)}}}$ is equal to the probability that an odd number of $l(n)$ coin flips turn up heads. This is clearly $\frac{1}{2}$.

Now, if x is from $\mathbf{F_{l(n)}}$, A' succeeds if the number of failures is less than or equal to $\lfloor \frac{l(n)}{2} \rfloor$. The average probability of failure is $1 - (\frac{1}{2} + \delta(n))$, which we call p_f. Of course, this calls upon our knowledge of the binomial distribution:

$$\mathbf{Pr}[\text{Success}] = \sum_{i=0}^{i=\lfloor \frac{l(n)}{2} \rfloor} \frac{(l(n))!(p_f)^i(1-p_f)^{(l(n)-i)}}{i!(l(n)-i)!}$$

We simply want to say that this is always greater than or equal to $\frac{1}{2}+\delta(n)$. We call upon our lemma:

$$\begin{aligned}\mathbf{Pr}[\text{Success}] &= I_{1-p_f}(l(n) - \lfloor\frac{l(n)}{2}\rfloor, \lfloor\frac{l(n)}{2}\rfloor + 1) \\ &= I_{p_s}(\lceil\frac{l(n)}{2}\rceil, \lfloor\frac{l(n)}{2}\rfloor + 1) \\ &= \frac{\int_0^{p_s} t^{(\lceil\frac{l(n)}{2}\rceil-1)}(1-t)^{\lfloor\frac{l(n)}{2}\rfloor}\,\mathrm{d}t}{\int_0^1 t^{(\lceil\frac{l(n)}{2}\rceil-1)}(1-t)^{\lfloor\frac{l(n)}{2}\rfloor}\,\mathrm{d}t}\end{aligned}$$

Clearly, if $p = \frac{1}{2} + \delta(n) = 1$, then $\mathbf{Pr}[\text{Success}] = 1 = p$. If $l(n)$ is odd, then it is easy to see that the quantity inside the integrals is symmetric about .5, and thus $p = \frac{1}{2}$ implies $\mathbf{Pr}[\text{Success}] = .5 = p$. If $l(n)$ is even, then the success probability is strictly greater than that for $l(n)+1$ (since it still takes the same number of incorrect guesses for A' to fail with $l(n)+1$ tests, but there are more chances for failure), so we have $\mathbf{Pr}[\text{Success}] > p = .5$.

Of course, the nice thing to have would be that

$$\mathbf{Pr}[\text{Success}] \geq p$$

on the interval $[.5, 1]$. We have the statement at $p = .5$ and $p = 1$. Since the function inside the integral is continuous, we can apply the First Fundamental Theorem of Calculus, to get

$$\frac{\mathrm{d}(\mathbf{Pr}[\text{Success} - p)]}{\mathrm{d}p} = \frac{(p)^{(\lceil\frac{l(n)}{2}\rceil-1)}(1-p)^{\lfloor\frac{l(n)}{2}\rfloor}}{\int_0^1 t^{(\lceil\frac{l(n)}{2}\rceil-1)}(1-t)^{\lfloor\frac{l(n)}{2}\rfloor}\,\mathrm{d}t} - 1,$$

and also $\frac{\mathrm{d}(\mathbf{Pr}[\text{Success}]-p)}{\mathrm{d}p} = 1$ at $p = -1$. Thus, if the derivative has only one root in $[.5, 1]$, we have the statement (since the derivative is clearly continuous, we apply Role's Theorem). This is fairly clear, so we can finally say that the accuracy of A' if x came from $\mathbf{F_{l(n)}}$ is at least $\frac{1}{2} + \delta(n)$.

The success-probability of A', then, is

$$\begin{aligned}\delta_{A'}(n) &= \sum_{x\in\{0,1\}^{l(n)}} (\mathbf{Pr}[\mathbf{F_{l(n)}} = x]\mathbf{Pr}[\mathbf{A'(x)} = 0] \\ &\qquad + \mathbf{Pr}[\mathbf{X_{\{0,1\}^{l(n)}}} = x]\mathbf{Pr}[\mathbf{A'(x)} = 1]) - 1 \\ &\geq \frac{1}{2} + \delta_A(n) + \frac{1}{2} - 1 \\ &\geq \delta_A(n).\end{aligned}$$

As for $T_{A'}(n)$, the adversary simply calls A, does $l(n)$ mod 2 additions, one division, and one comparison. So $T_{A'}(n) = T_A(n) + n^{\mathcal{O}(1)}$, so since $T_A \geq n^{\mathcal{O}(1)}$ (it returns $l(n)$ bits), we have $T_{A'} = \mathcal{O}(T_A)$.

So the time success-ratio of A' to mount a distinguishing attack is $R_{A'}(n) \leq \mathcal{O}(R_A(n))$. ■

Theorem 3. *If a pseudo-random bit-string generator f defines a variable $\mathbf{F_{l(n)}}$ which is distinguishable from $\mathbf{X}_{\{0,1\}^{l(n)}}$ by A with time-success ratio $R_A(n)$, then there exists an adversary A' which is a next-bit predictor with time-success ratio less than $\mathcal{O}(R(n))$.*

Still to do:

(A) Prove this (due to Yao)

(B) Talk about the existence criteria for pseudo-random generators (iff one-way functions exist, meaning necessary condition is $P \neq NP$.)

(C) Probabilistic encryption (basically, that the ciphertext space, considered as a random variable, is indistinguishable from a random variable)

References

[1] Luby, Michael George., *Pseudorandomness and Cryptographic Applications,* (Princeton, NJ: Princeton University Press, 1996).

[2] Stinson, Douglas R., *Cryptography: Theory and Practice,* (Boca Raton: Chapman & Hall, 2002).

[3] Yao, Andrew C., "Theory and applications of trapdoor functions," *In Proceedings of the Twenty-third Annual IEEE Symposium on Foundations of Computer Science,* Chicago, Illinois, November 1982, 80-91.

[4] Hastad, Johan and Impagliazzo, Russell and Levin, Leonid A. and Luby, Michael, "A Pseudorandom Generator from any One-way Function," *SIAM Journal on Computing,* **28** (4), 1364–1396.