

/dev/erandom: The Inner Workings of a Provably Secure PRNG

Seth Hardy



tsumego

FOUNDATION

shardy@tsumego.com

<http://www.tsumego.com>

ToorCon 101
September 28, 2003

Let's start with a few questions...

- What is a *pseudorandom number generator (PRNG)*?
- Why do we even need a new PRNG?
- What do *random* and *pseudorandom* mean? How about *quasirandom*?
- How can I tell whether a PRNG is good? What does “good” mean?
- What does “provably secure” mean in this context?
- How does `erandom` compare to the existing `/dev/{u}random`?

Why reinvent the wheel?

- Additional features... from a PRNG?
- Provable security... but what does that mean?
- Speed and efficiency: different platforms and uses have different needs.
- Simplicity of overall design: ease of use, understanding.
- Modularization: ease of extension, customization.

Section I: Background

PRNGs

What is a “pseudorandom number generator”?

$$G : \{0, 1\}^k \rightarrow \{0, 1\}^n$$

However, we'd like to see G have a few specific properties for it to be *useful*:

- n larger than k
- $G(x)$ computationally indistinguishable from random
- Hard to predict output even with some knowledge of the system.

So what would a good measure of evaluating these properties be?

Random? Sequence 1

Let's start by looking at a simple sequence:

1, 1, 1, 1, 1, . . .

Is this random?

Random? Sequence 1

Let's start by looking at a simple sequence:

1, 1, 1, 1, 1, . . .

Is this random?

We need to define the set we're picking from. What if $S = \{1\}$?

Yes! This is a random sequence, if we are picking from the above set.

Random? Sequence 2

Now let's assume that $S = \{1, 2, 3, 4\}$, and look at the same sequence:

1, 1, 1, 1, 1, . . .

Is this still random?

Random? Sequence 2

Now let's assume that $S = \{1, 2, 3, 4\}$, and look at the same sequence:

1, 1, 1, 1, 1, . . .

Is this still random?

We still haven't defined the probability of picking each $x \in S$. What happens if we pick according to these probabilities:

$$\Pr[x = 1] = 1 \quad \Pr[x = 2] = 0 \quad \Pr[x = 3] = 0 \quad \Pr[x = 4] = 0$$

Yes! This is also a random sequence, according to the above set and probability distribution.

Uniform Distribution

We can pick randomly according to any “probability distribution”:

$$\mathcal{D} : S \rightarrow \mathbb{R}$$

The probability distribution \mathcal{D} assigns a nonnegative probability to each $x \in S$, such that

$$\sum_{x \in S} \mathcal{D}(x) = 1$$

What if we want to pick something “*at random*”?

When most people say “at random”, what’s usually meant is “*uniformly at random*”. The *uniform distribution* \mathcal{U} is the probability distribution where everything is picked equally often:

$$\text{If } |S| = n, \text{ then } \Pr[x = s] = \frac{1}{n} \text{ for each } s \in_{\mathcal{U}} S.$$

Statistical Distance

How can we tell how “far apart” distributions are?

The *statistical distance* (also known as the L_1 *metric*) between two probability distributions \mathcal{D} and \mathcal{E} is:

$$d(\mathcal{D}, \mathcal{E}) = \frac{1}{2} \left| \sum_{x \in S} \mathcal{D}(x) - \mathcal{E}(x) \right|$$

Often we want to see how close a distribution is to the uniform distribution \mathcal{U} . If $d(\mathcal{D}, \mathcal{U}) \leq \epsilon$, then we say \mathcal{D} is ϵ -close to uniform.

Alternatively, we could say \mathcal{D} is *quasirandom within ϵ* .

Entropy

Entropy is a common term used when looking at randomness.

But what exactly is entropy?

- A measure of information?
- A measure of randomness?
- A measure of redundancy?

What about different types of entropy? Shannon entropy, Renyi entropy, min entropy...

Shannon Entropy

The *Shannon entropy* H (often just called *entropy*) is the basic measure of information:

$$H(\mathcal{D}) = - \sum_{x \in S} \mathcal{D}(x) \log_2 \mathcal{D}(x)$$

Shannon entropy is measured in bits per “symbol” (each element in S).

For example, $H(\text{English}) = 2.62$. (We are looking here at the probability distribution over the set $S = \{A, B, C, \dots, Z\}$.)

However, $\log_2 26 \approx 4.70$, showing that there are approx. two bits of redundant information in each English character.

Min Entropy

Min entropy H_∞ can be thought of as a measure of the worst possible case of a probability distribution:

$$H_\infty(\mathcal{D}) = \min\{-\log_2 \mathcal{D}(x) : x \in S\} = -\log_2 \max\{\mathcal{D}(x) : x \in S\}$$

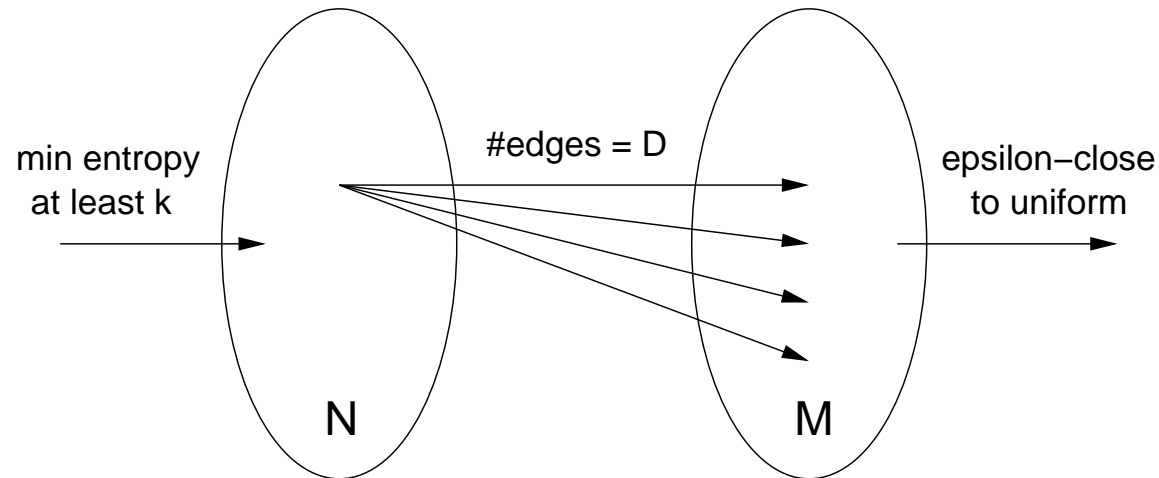
It is possible for a distribution to have a fairly high Shannon entropy, but a small min entropy.

For example, let $\mathcal{D}(x) = \frac{1}{2}$ for some $x \in S$ and some very small probability for all other $x' \in S$.

Section II: Theory

Extractors

Definition. Let $[N], [M]$ be sets of vertices with sizes N and M , and E be a set of edges going from $[N]$ to $[M]$. The graph $G = ([N], [M], E)$ is a (k, ϵ) -extractor if, for any probability distribution \mathcal{D} on $[N]$ with $H_\infty(\mathcal{D}) \geq k$, $\Gamma(\mathcal{D})$ is ϵ -close to uniform on $[M]$.



How “bad” is the input distribution \mathcal{D} ? – What is the min entropy of \mathcal{D} ?

How “good” is the output distribution \mathcal{E} ? – How close to uniform is \mathcal{E} ?

Provable Security

Extractors take a “bad” distribution on $[N]$ and random bits, and use the additional randomness to “smooth” out the distribution into a “good” one over $[M]$.

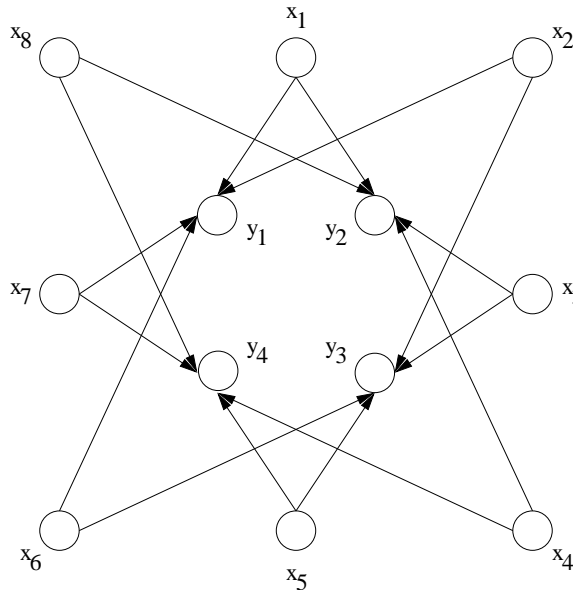
This is a provable level of security: we know how close to uniform the output will be, as long as the input meets the entropy requirement.

Provable security in this sense does not mean that it is unbreakable!

(this last line was in a box because of how important it is; read it again)

There is a provable bound on computation needed to distinguish output from uniform; this does not cover implementation, etc.

A Sample Extractor

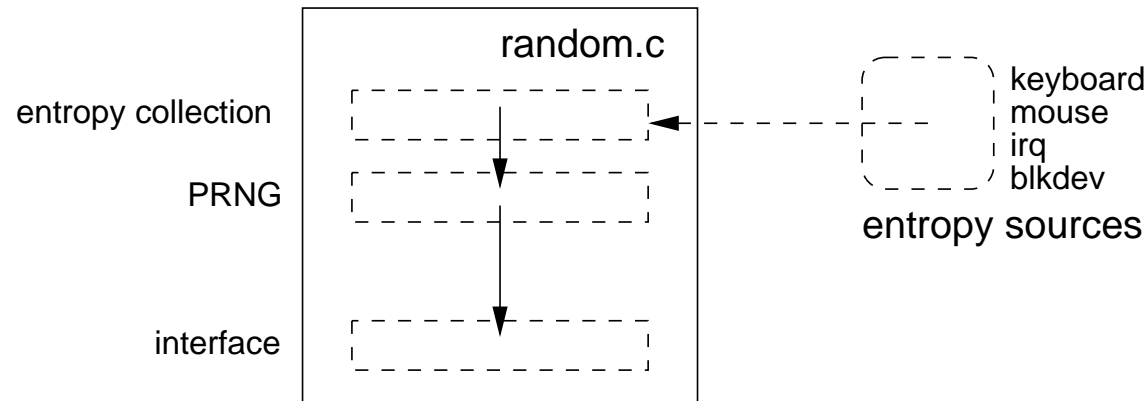


- x_i are the “bad” bits; edges from x_i to y_j are the “good” bits.
- y_j are the output bits.

Can anyone tell whether this is a good extractor?

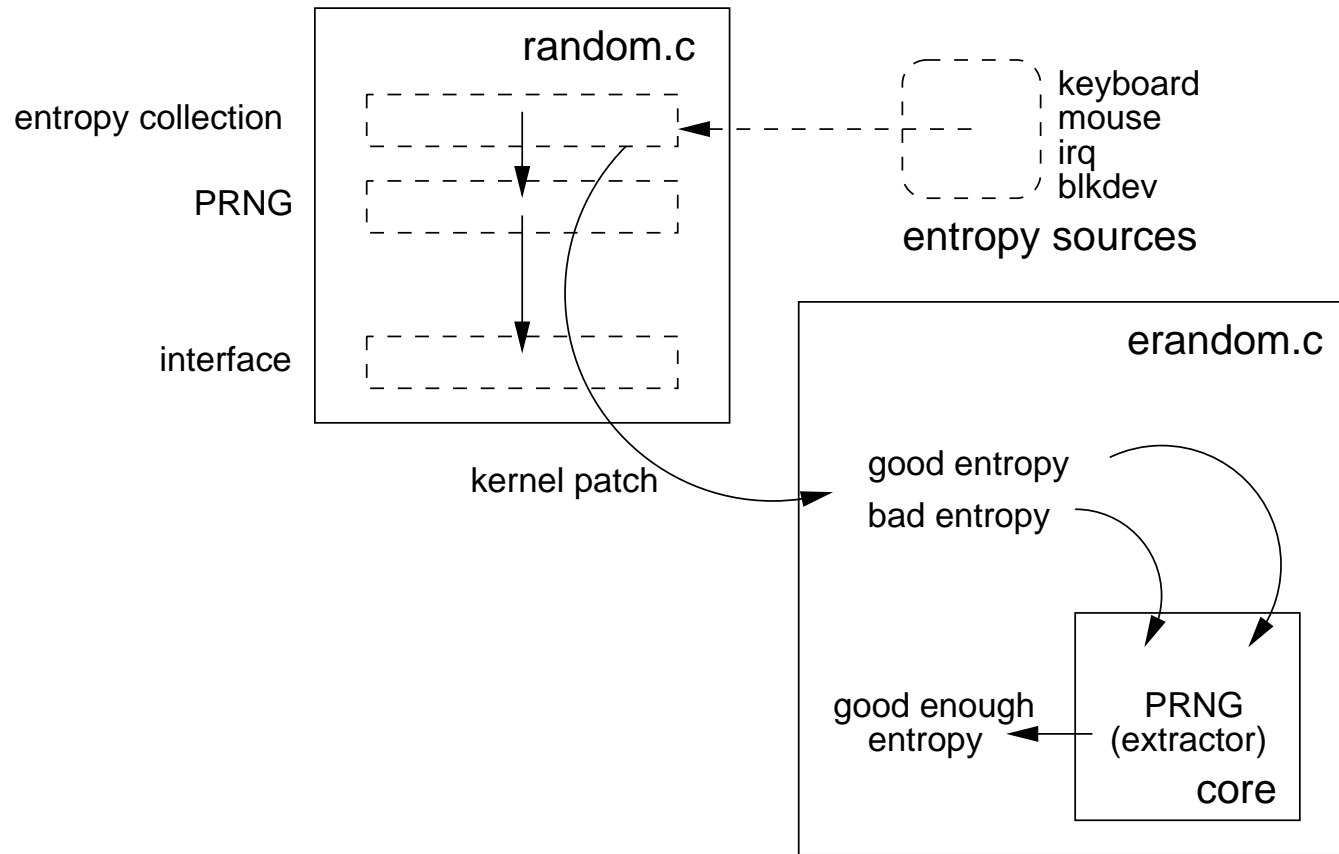
Section III: Implementation

/dev/{u}random Overview



- Exports functions to get timings from various sources (keyboard, mouse, etc.).
- Gathers entropy, uses as input to PRNG.
- Maintains internal “entropy pool” and number of good bits in pool.
- Hashes pool and provides bits as output when requested.
- `random` blocks when good bit count hits 0; `urandom` does not.

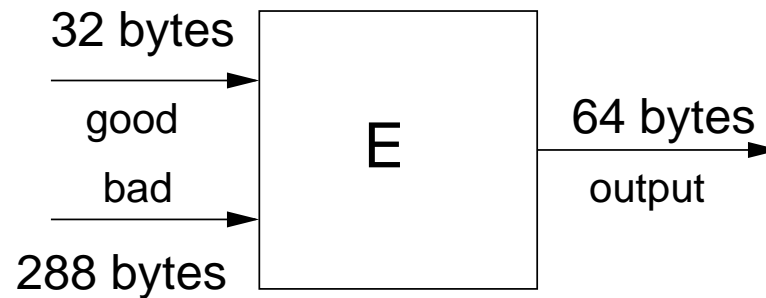
/dev/erandom Overview



/dev/erandom Overview, Continued

- A kernel patch is required to intercept the entropy from `random.c`.
- `erandom` runs as a module, and requires an extractor core.
- Extractor cores are also loaded as modules. Support for multiple cores possible.
- `erandom` uses the lowest order bit as “good” and the rest as “bad.” Core independent.
- The core implements an extractor, and registers the function with `erandom`.
- `erandom` provides an interface to the core as a character device, just like `/dev/random`.

Sample Core: $(k - 1)$ -U Hash Family Extractor



For $a \in \mathbb{F}_{2^{256}}$, let

$$f_a(x_0, \dots, x_9) = x_0 + \sum_{i=1}^9 x_i a^i$$

As a graph: view x_0, \dots, x_9 as the vertex on the left, a as an edge, and f_a as the vertex on the right.

The output of the extractor is $a \circ f_a(x_0, \dots, x_9)$.

Section IV: Conclusions

(Hopeful) Improvements

These are the features that `erandom` (hopefully!) gives us:

- Fewer assumptions about the quality of entropy gathered.
- Provable level of security; assuming input bits are good enough, output bits maintain a particular level of security.
- Speed of internal operations (extractors can be fast and simple).
- Simplicity of design.
- Modularity: support for multiple cores, letting users write their own to suit particular needs.

Future Work

The eventual goals of this project are to:

- Finish `erandom` as a kernel device compatible with `/dev/{u}random`.
- Separate entropy gathering from the PRNG in `random.c` and elsewhere in the Linux kernel.
- Add support for alternate entropy gathering sources.
- Finish support for multiple cores.
- Create tutorials to make it easy for users to design their own cores.
- Implement the `erandom` framework on other OSes, while keeping cores portable.

Questions?