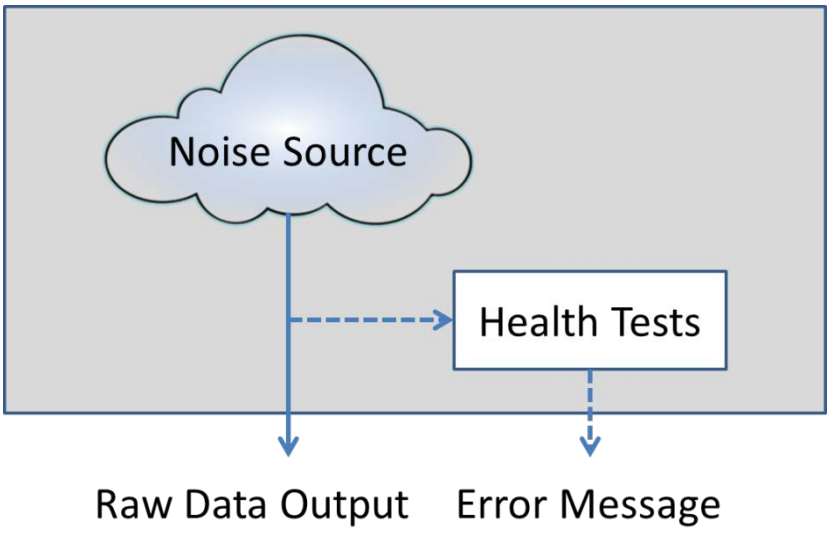
Minimizing false negative and false positive errors on entropy health tests

Scott Fluhrer, Cisco Systems
sfluhrer@cisco.com

Abstract: this paper gives an alternative model of continuous health tests that allows us to drastically reduce the error rates; that is, it allows us to better detect when a noise source has degraded to the point where it imperils security, while at the same time reducing the number of false alarms.

The current NIST SP 800-90B draft proposes a model for an entropy source. In this model, there is a noise source, which generates a (hopefully) unpredictable signal. This signal is scanned by the continuous health tests, which will raise an error if it detects that the signal exhibits signs that indicate that it may be more predictable than expected. These health tests can exhibit both false negative errors (not detecting a problem when there is one) and false positive errors (claiming that there's a problem when there isn't one). What we propose is to a more sophisticated model of these health tests; by introducing the idea of a consumer entropy rate, which is the maximum rate at which the entropy output by the device will be consumed, and having that value be smaller than the rate that the health tests are tuned for, we can minimize the probability that failures will be undetected while also keeping the false positives at a low level.

The NIST 800-90B model on the entropy source can be illustrated in the below diagram, which is a simplification of Figure 1 of 800-90B (with the parts not immediately relevant to this discussion omitted).



The continuous health tests monitor the noise source, and triggers an error message when it believes it detects a problem.

One potential problem that such a health test may run into is a false negative; where the noise source in question has insufficient unpredictability, but the health test does not detect it. We all agree that it is important to minimize this possibility. The draft itself notes that (section 4.2), the current design "may result in poor entropy source outputs for a time, since the error is signaled once significant evidence has been accumulated, and these values may already have been output by the entropy source." It would be good to reduce this window of vulnerability.

The other potential problem is a false positive; when the health test claims there is a problem on a correctly operating system. One requirement that NIST demands is that the approved continuous health tests be tuned to have a false positive error rate (that is, the probability that it claims that there is an error when the system is functioning properly) of at least 2^{-50}. The reasoning they give is that they want to have the tests have enough power to detect major failures, that is, to reduce the false negative error rate.

While 2^{-50} may sound like an impressively low false positive error rate, we would claim that it is still may be too high in some scenarios. It has been estimated that there will be approximately 6 billion IOT devices in use in 2016; increasing to 20 billion devices by 2020. In addition, each device may sample their entropy a large number of times over the course of their lifetime. When we multiply these two factors, it becomes likely that there will be multiple devices in the field that will trigger a false positive.

In addition, there is a second factor that may increase the number of false positives. The noise source is ultimately an analogue device (even if it is made up of digital components, say, several ring oscillators, it is still dependent on the gate delays, which are analogue), and different analogue devices differ in subtle ways, as well as may degrade slightly over time and environmental conditions. If we tune the health tests to give a false positive rate of 2^{-50} for the nominal device (that is, the device that the entropy was initially measured at), there will likely be devices in the field that have slightly worse entropy characteristics. These characteristics might not be enough to cause a real problem; however they would significantly increase the chance that the health tests would trigger.

> To take a concrete example, consider a binary noise source that has individual samples which are unbiased, but has correlations between samples; specifically, the probability of the next sample is the same as the previous bit is > 0.5.
>
> If the intrabit probability was 0.70 (that is, each bit had a probability of 0.70 of being the same as the previous), then the minentropy rate of this source H is approximately 0.515, as the maximum probability of any specific n bit string occurring is $0.70^n \approx 2^{-0.515n}$
>
> To obtain a trigger probability of 2^{-50} on the repetition count test (section 4.4.1; the appropriate test for a source with this bias), we would have a value C = 97; that is, it would trigger if we ever saw 97 repetitions of the same bit.
>
> Now, consider the case where the noise sample degraded to the point that the intrabit probability was actually 0.75. In this case, performing the same health test, we obtain a trigger probability of $2^{-40.2}$ or, in other words, 1000 times larger.

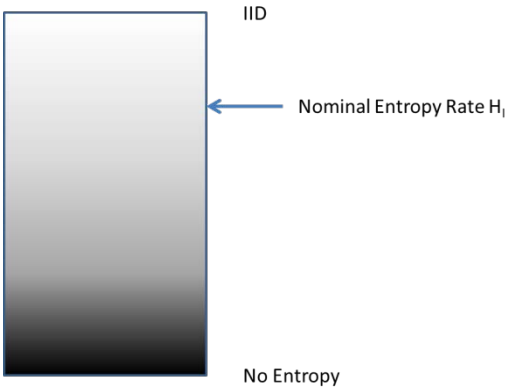
If such degradation is to be considered innocuous, we've significantly raised the false trigger rate. On the other hand, if such degradation is considered dangerous, we actually have only a small probability of detecting it, and so we have a probable false negative condition.

Another question that would need to be addressed is the actual action taking when the health tests detect a possible failure. If the device is indeed malfunctioning in a way that prevents it from operating securely, we would ideally want to force it to shut down; generating an error message would be too easily ignored (even if the documentation states that such you shouldn't ignore such error messages). However, if false positives were likely, shutting down the device may be expensive (both in that it is inconvenient to users, and force the implementor to handle an expensive field call). In addition, false positives may hide true positives; if the service personal see enough false positives, they may ignore all of them.

Now, if this false positive probability were required to achieve an acceptably low false negative probability, this would be an acceptable tradeoff. However, we would suggest an alternative approach that can both improve on the false negative rate (which would enhance security) and the false positive rate (which would lower the cost, and allow implementators to make more drastic actions on detected failure, which would also enhance security), at a cost which is low in some scenarios.

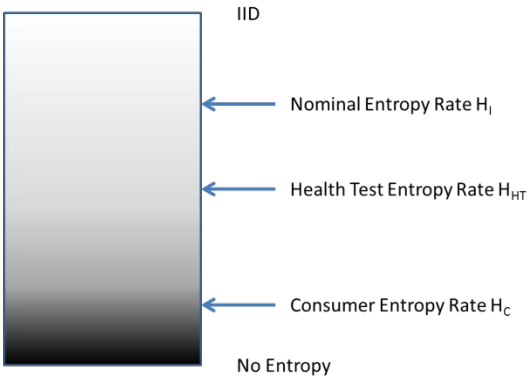
The current draft uses the nominal entropy rate H_I, that is, the rate that was measured as a part of the initial entropy estimate, for both the health test measurements, and the entropy rate given to the consuming application (after the optional conditioning, which cannot add entropy).

This can be illustrated by the below diagram, where the shading within the rectangle gives the relative probability of the health test triggering a failure:



Because H_I gives a slight probability of triggering a failure, the rectangle is slightly shaded there.

What we suggest is instead you specifically allow a model where there are three explicitly stated entropy rates, illustrated as follows:



These three rates are defined as follows:

- The nominal rate H_I is the initial entropy estimate as current; it is the minimum of the entropy rate by design, and the rate that is measured by the entropy tests
- The health test entropy rate H_{HT} is the expected rate where the health tests give a failure rate of 2^{-40}. It is specifically allowed to be lower than the nominal rate. The shading in the diagram has been lowered to reflect the tuning at the lower entropy rate.
- The consumer entropy rate H_C is the entropy rate that is assumed by devices that use the raw data output. That is, if the consumer entropy rate is H_C bits of minentropy per symbol, and the consumer (the entity that uses the entropy) asks for *n* bits of min entropy, then the noise source generates at least n/H_C symbols (possibly more depending on conditioning). The consumer entropy rate is specifically allowed to be lower than the health test entropy rate; it is forbidden to be higher.

Making H_{HT} lower than H_I reduces the false positive rate (as it makes it less likely that a correctly functioning noise source would accidentally generate an output that appeared to have an entropy rate significantly less than H_I). Making H_C lower than H_{HT} reduces the false negative rate (as security won't be at risk unless the actual entropy rate drops below H_C, and having that considerably below H_{HT} maximizes the chances that the health tests would recognize it quickly).

> To continue our example, let us assume that we have the same noise source, but in addition to having an H_I of 0.515, we declare ourselves to have a health test entropy rate $H_{HT} = 0.25$ and a consumer entropy rate $H_C = 0.04$.
>
> In this case, the repetition count test (with probability 2^{-40}) would have a C = 161. The consumer entropy rate of 0.04 means that, when the consumer asks for 256 bits of minentropy, we give it at least 256/0.04 = 6400 sampled noises sources. If the noise source is sufficiently degraded that we don't actually have 256 bits of minentropy in those 6400 samples, that means that our noise source has an intrabit probability of >0.972; at that probability, the repetition count test with C = 161 would trigger with a probability >0.88 during the collection of those 6400 samples. Hence if the device is degraded to such an extent to actually cause a security issue, we are likely to report it before it has output a single entropy sample. Hence, the false negative probability is minimized.

In addition, C=161 gives us a false negative rate on the fully functional device of 2^{-82}, and a false negative rate on the somewhat degraded device (with an interbit probability of 0.75) of 2^{-66}. Both are sufficiently small that they are likely never to appear in the field, and so only devices that are, in fact, misbehaving would report failures. Hence, we effectively have no false positives.

Note: it turns out that we lower the consumer rate this much because this test turns out not to trigger with high probability unless the correlation is extreme; in other scenarios, it is likely that we wouldn't need to resort to such low rates. On the other hand, as shown below, this relatively extreme lowering might be of acceptable cost.

Let us highlight a major point of the above example: if the device has an entropy source which is degraded to the extent that it actually poses a security threat, then (with probability > 88%) the health tests would trigger (and presumably the device would shut down) before we use a single low-entropy sample.

In addition, because we have reduced the false positive probability to such an extent, it is realistic to shut down the device when the health tests trigger.

Now, what would be the cost of implementing this idea? The cost is that we collect more samples from the noise source what we would otherwise need to. In our example, for 256 bits of minentropy, we would actually need only 500 samples from the noise source on a correcting operating device; we actually end up collecting 6400 samples, thirteen times more than what we actually needed. Also remember that, even though we internally collect 6400 samples, we need not actually deliver that large of a sample set outside of the noise source; the FIPS model allows the implementation to include a conditioner within the noise source, which can reduce that large sample to a more reasonable size.

Some implementations have hardware entropy sources, such as ring oscillators. For these devices, collecting 6400 samples has no significant cost over collecting 500 samples; so here this idea gives us the benefit of drastically reducing the positive and negative error rates, at essentially no cost. On the other hand, other implementations have expensive entropy sources, and so this idea would not be universally applicable.

We make the following recommendations:

- We strongly recommend that the 800-90B draft be amended to specifically allow an implementation to do this. It is not clear whether this idea is permitted under the current 800-90B draft; a reference lab may very well decide that it is not. Because this idea would allow us, on some devices, to reduce both negative error rates (which would improve security), as well as reduce positive error rates (which would reduce cost), it would make sense to explicitly allow this as an option within NIST SP 800-90B.
- We further recommend that this model (or something similar) of three distinct entropy rates be considered for incorporation into 800-90B. That is, it would be an allowable option to have distinct $H_I \geq H_{HT} \geq H_C$ rates, which would be published in the FIPS documentation (and verified as a part of the FIPS certification).
  Documenting these rates would have the following advantages:

    - It would allow the reference lab to verify that the H_{HT} rate has the expected failure rate, and that the consuming application respects the H_C rate.
    - It would give more transparency to the user of the entropy source as to have well it works; it would allow them to select entropy sources with extremely low false positive and false negative rates.
- We also recommend that better health tests be developed. While this idea can be helpful in reducing false negatives, it assumes that the health test will actually detect very low entropy conditions. It is not clear that the current health tests in the 800-90B will reliably detect such conditions with real noise sources; more robust health tests (which are designed to detect the failure modes of the noise sources they are used with) would be desirable.