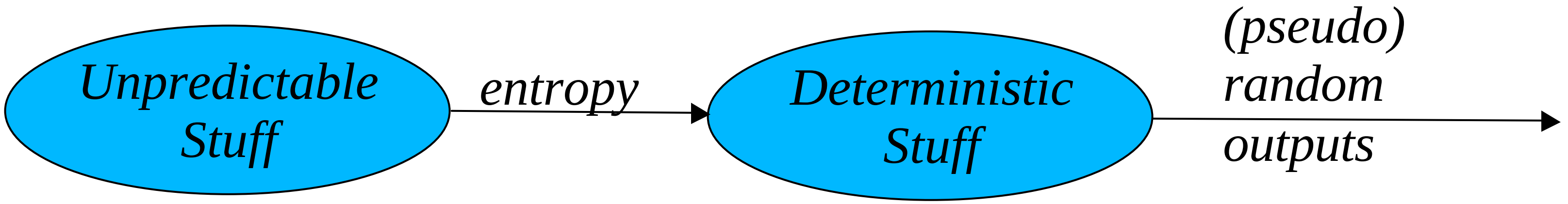
Entropy and Entropy Sources in X9.82

John Kelsey, NIST, July 2004

Overview

- Entropy Sources in X9.82
- A Disclaimer....
- Our Model of an Entropy Source
  - Nondeterministic Mechanism
  - Sampling/Measurement
  - Assessment
  - Conditioning/Buffering
  - Testing and Assurance
- Open Issues

Entropy Sources in X9.82



- Any way to generate random bits has two parts:
  - *Something nondeterministic, which provides all the ultimate unpredictability of the system*
  - A deterministic generation mechanism, which provides uniform, independent output bits with all the required security properties.

Entropy Sources in X9.82 (2)

- In X9.82, the nondeterministic part is called an *entropy source. This come in two flavors:*
- *An entropy source....*
  - *Provides bitstrings that are not entirely determinstic*
  - *Provides assessments of the min-entropy of its outputs*
- *A conditioned entropy source...*
  - *Provides bitstrings that are expected to be full entropy—uniform, independent, balanced, etc.*

Entropy Sources, DRBGs and NRBGs

- DRBG uses entropy source for seed material
  - Get_entropy(min-entropy, min-length, max-length)
  - Instantiate, Reseed, Generate (prediction resistance)
- NRBG uses entropy source constantly
  - DRBG reseeding between outputs of < *k* bits
  - DRBG ⊕ conditioned entropy outputs
  - Conditioned entropy outputs

In Part 3, we talk about bits of *security*
In Part 2, (here) we talk about bits of *entropy*

To Summarize....

- The *entropy source is where we put the nondeterminstic parts of our designs*
  - *Everything else is deterministic*
  - *Most other stuff is easy to specify*
  - *We know how to analyze most everything else*
- *Assumption: Nothing outside the entropy source knows anything about its operations, except the claimed assessed entropy.*
  - *Conditioned Entropy Sources claim full entropy.*

A Disclaimer....

- *This is the part of the document we've done the least on*
- *This includes many hard problems for standards and especially validation!*
- *Most of this presentation is subject to change as we understand the problems better*

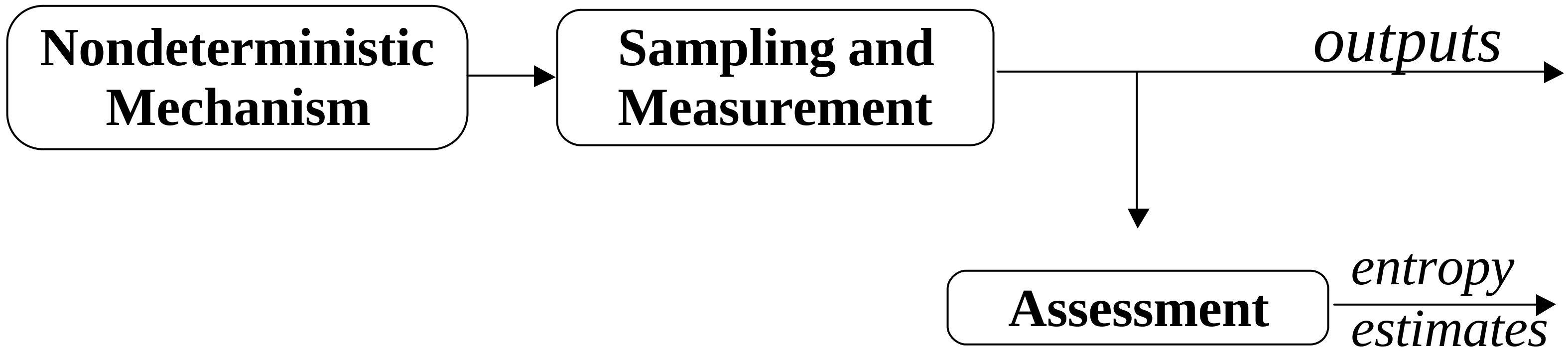
This is the part of X9.82 where we need THE MOST input!

Anatomy of an Entropy Source

Nondeterminsitic Process
Sampling and Measurement
Assessment
Conditioning and Buffering

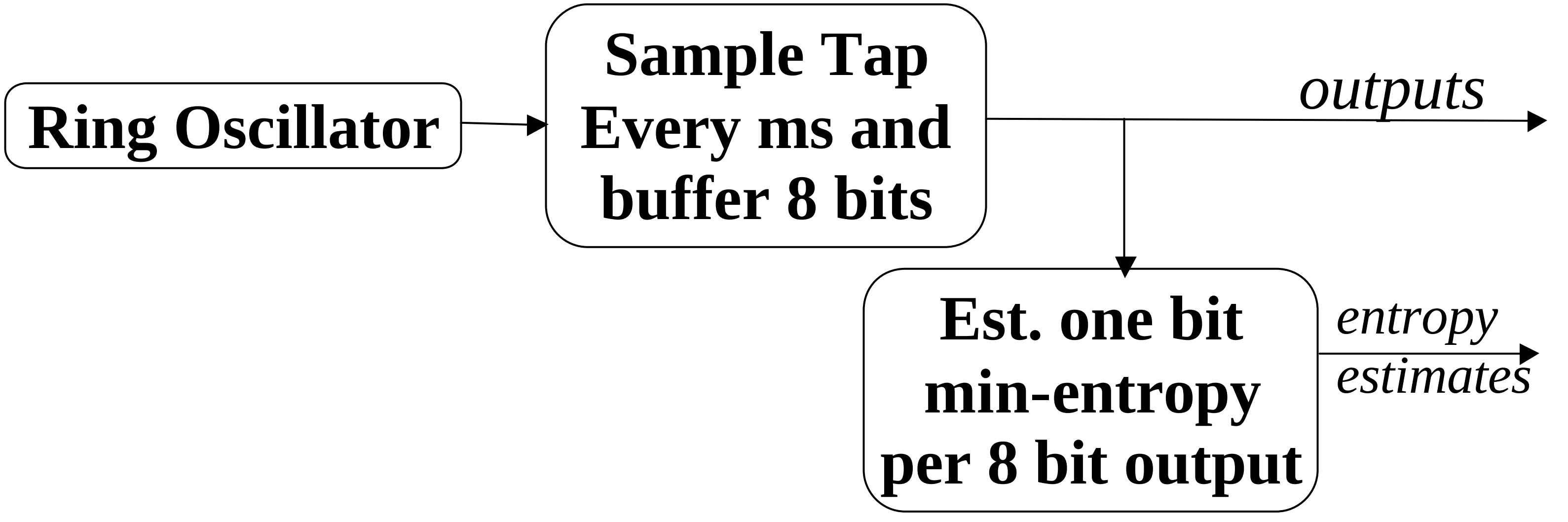
} *Probability Model*

Basic Components of an Entropy Source



- *Nondeterministic Mechanism*—provides unpredictability
- *Sampling and Measurement—converts the unpredictable events to bits in some well-understood way*
- *Assessment—estimates the min-entropy in the outputs*

An Example of an Entropy Source



- Unpredictability from phase noise in ring oscillator
- Sampling on a stable clock signal
- Fixed assessment based on model

Nondeterministic Mechanisms: Unpredictability

- *Practical Requirement: There are limits to any attacker's ability to predict/model its behavior.*
- *Three broad categories:*
  - *Physics (noisy diode)*
  - *Insufficient Info (coin flips)*
  - *Too big/complex to model (human keypress timings)*
- *To be useful, unpredictability must somehow be quantifiable.*
  - *Probability Models from Knowledge/Physics*
  - *Probability Models from Analysis of Data*

Nondeterministic Mechanisms: Quantifying Unpredictability

- We use *min-entropy as a measure of uncertainty in X9.82*
  - $-lg(max (P(event)))$
  - *Only the most probable outcome matters for min-entropy*
- *Measures probability of attacker's best guess about an unknown value*

Amount of unpredictability always based on best available model of process.

Min-Entropy Example: How Many Heads in Two Coin Flips?

- Flip 2 fair coins
  - P(0 heads) = 1 / 4 (TT)
  - **P(1 head) = 1 / 2 (HT, TH)**
  - P(2 heads) = 1 / 4 (HH)
- Min-entropy is based only on biggest prob.:
  - -lg(1 / 2) = 1 bit of min-entropy

Attacker's best guess has prob. ½ ; min-entropy=1

Nondeterministic Mechanisms: Performance

- Entropy Rate (bits min-entropy/sec):
  - Electronic systems tend to be fastest
  - OS loading and system level events slower
  - Human/computer interactions and simple physical events even slower
- Cost and problems with handling source
  - Few Geiger-counter sources in use for crypto
  - Few automated “lotto” drawings in use for crypto

Nondeterministic Mechanisms: Modeling and Failure Detection

- To be useful, must be able to be modeled
- Normal Behavior
  - Based on source of unpredictability
  - Can be very complicated model of physical source, or simple description of estimated min-entropy
- Failure Modes
  - Ideally, alternate models
- Environmental Limits/Conditions
  - Can be parameter in model, or limits

Nondeterministic Process: More Modeling Comments

- Not amenable to cookbook approach
  - Best model often research problem
  - Advanced statistical methods
  - Never a "last" statistical test or model to consider
- We hope to give some guidance
  - Stable distributions relatively easy to handle
  - Decisions on sampling/measurement can lead to simpler models

The model determines assessed entropy, health tests, etc.

Nondeterministic Mechanisms: Examples In Use

? Ring oscillators

? Amplifier + Digitized voltage levels

? OS Loading (sample fast clock in tight loop)

? Hard drive access times (with cache misses)

? Mouse movements/timings

? Key press timings

? Coins

? Dice

Sampling and Measurement

- Entropy source produces *bits*.
- Have to measure and digitize process
  - Sometimes inherent in process
    Example: coin flips
  - *Measuring can introduce errors:*
    - *Noise*
    - *Systematic bias*
- *Can be a major hassle in some designs*
  - *Are you measuring what you think you're measuring?*
  - *How much of sample variability is entropy, how much is just complexity?*

Sampling and Measurement: Modeling Issues

- Sampling / Measurement must be included in model:
- Sampling rate can solve some model problems
  - Example: Sampling voltage level/thermal noise source too fast gives correlated inputs; drop sample rate to fix
- Measurement technique can solve some model problems
  - Example: Measure difference in voltage drop from adjacent resistors to eliminate non-noise effects

Assessment

- First, we have nondeterministic process
  - Justify why it's really nondeterminstic
- Then we have model
  - Model includes process and sampling/measurement
- Finally, we have to assess entropy
  - What are limits on best possible model?
  - Describe in terms of min-entropy
- And we have to decide if something's wrong
  - Fail or stop outputting when things don't look right

Assessment with Fixed Model

- Simplest assessment: fixed model for all instances of entropy source
- *In design/analysis, work out a model and get min-entropy estimate*
- *Very common in the literature*
  - *Keystrokes*
  - *Ring Oscillator Samples*
  - *Coin Flips*
  - *Voltage measurements*

Variable Assessment

- Can also change assessed entropy based on inputs seen lately
- Examples:
  - Keystroke estimation in PGP: ignore entropy from repeated keys
  - Change entropy estimates based on second derivative of mouse position, packet arrival time, etc.
  - Feed source outputs into general purpose compressor; alter assessments based on size of outputs.
  - Von Neumann unbiasing: size of output sequence determines assessment, but still output original sequence

Altering Model Parameters Over Time

- Can update some parameters in model over time to adjust assessments
- Example: source of biased, independent bits
  - Assess entropy based on bias
  - Can precompute table of probs/assessments per bit
- Could do this at startup, to get per-device variation
  - We don't know of any real-world systems that do this

Continuous Assessment and Limiting Outputs

- Alternative way to do variable assessment: Keep assessment same, limit rate of outputs
- Example: Only output sample when different than previous N samples
- Example: As measured bias increases, XOR together more and more sampled bits to make each bit of output

Conditioning: Two Approaches

Take biased/correlated bits from sampling and map to uniform, independent full-entropy bitstrings.

- Mathematical: use precise model of process and sampling
  - Example: Von Neumann Unbiasing of biased coins (Output 1 for HT, 0 for TH, and nothing otherwise)
- Hashing: assume only that distribution doesn't interact badly with process; leftover hash lemma
  - Example: Compute CRC of output bits, when enough are collected, assess as full entropy and output

Mathematical Conditioning

- Idea: Conditioning is based on probabilty model of samples from nondeterministic process
- Example: Von Neumann Unbiasing
  - 00, 11: output nothing
  - 10: output 1
  - 01: output 0
- Good News: If model correct, no other assumptions required; doesn't waste entropy
- Bad News: Fragile! If model is wrong, bits are skewed/correlated.

Hashing

- Idea: Condition bits by mapping input strings to outputs in a random way
  - Example: Compute SHA1 on sequence of measured keystroke timings.
  - Example: Compute CRC on sequence of sampled ring-oscillator bits.
- Good News: Don't have to know much about model
- Bad News: May be based on additional assumptions about algorithm or source, need more entropy (leftover hash lemma)

Buffering and Collecting Output

- Sometimes also necessary to shorten outputs
  - DRBG needs 512-bit seed with 128 bits min-entropy
  - Low-rate from sampling of nondeterministic mechanism
  - Possible Solution: Accumulate entropy with some checksum, e.g. CRC32.
  - *How much analysis does this need?*
- This is related to conditioning, but not identical to it.

Periodic Testing: Health Tests

- Purpose: Determine whether model still describes process and sampling well.
- Tests specific to model!
  - Relatively easy for stable distributions
  - Quite complicated otherwise
  - Look for changes in behavior from expected
  - Example: See if expected mean and sigma are consistent with observed data from ring oscillator.

Can include data before sampling, if available
Should include data before accumulation

Health Tests and Conditioning

- Mathematical conditioning (tuned to model):
  - Can test both input and output
  - Determine best test and cutoffs by simulation or analysis
  - Conditioning is tuned to model; tests on output are for uniform independent unbiased bitstrings.
- Hashing (model independent)
  - Can test only input
  - Tests specific to model

Continuous Testing....

- Some tests can run continuously
- Major requirements:
  - Cheap
  - Low rate of false positives
  - Accumulate information over time
- When a continuous test fails, the device shuts down or dies.
- We have only scratched the surface here

Validation

- Not clear how validation lab should verify entropy source design
  - Documentation requirements nice, but who reads them at labs?
  - Cookbook statistical tests not so useful for complicated probability models
  - Can we require simple models (e.g., stable distributions) to make testing/validation easier?
- *How much expertise can we expect from validation labs?*

Validation Thoughts

- Validation requirements should depend on how critical entropy source is to system.
- Basic NRBG is most critical
  - Must do serious continuous tests
  - Must have very reliable design for nondeterministic processes
- Enhanced NRBGs and DRBGs
  - Less strenuous requirements if it keeps a seed file?
  - Reward conservative design?

Open Issues

- Most of this presentation is made of open issues