Overview

Today:
- HW 5 quiz
- Pseudorandom generation - concepts and simple techniques

To do before Thursday:
- Do HW 6
- Read “Security Models” handout
- Finish project phase 2 - report submitted in class, code in BitBucket

What do we mean by “randomness”?

Common perception - random physical events
- Flipping a coin
- Rolling a die
- Blind draw from a bag

Some properties:
- Statistically uniform
  - Non-uniform randomness is possible, but less interesting in crypto
- Independence
- Unpredictability (next numbers can’t be guessed)

Key concept: Entropy
- Measures amount of randomness from a random source
- Example 1: 64 true random bits has 64-bit of entropy
- Example 2: English language entropy is about 2-bits per letter
Random Number Generators

Delivers an unbounded-length sequence (stream)

TRNG - True Random Number Generator
- Sometimes called NRBG (non-deterministic random bit generator)
- Based on physical randomness
- OS can gather physical randomness - disk timing, mouse moves, ...
  - /dev/random in Linux - blocking random source
- Can also be special-purpose device (noisy diode, even a lava lamp)

PRNG - Pseudo Random Number Generator
- Sometimes called DRBG (deterministic random bit generator)
- Sequence computed from a seed
- Consumer of stream typically doesn’t know seed
- Computing again with same seed gives same sequence (repeatable)

TRNG/PRNG hybrids
- True randomness “mixed in” to pseudorandom generator
- /dev/urandom in Linux - non-blocking random source

Some applications and properties

What properties are needed in different applications?

<table>
<thead>
<tr>
<th>Application</th>
<th>Good Statistics</th>
<th>Unpredictable (fwd)</th>
<th>Repeatable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random simulation</td>
<td>Must have</td>
<td>No need</td>
<td>Depends</td>
</tr>
<tr>
<td>Nonce</td>
<td>Must have</td>
<td>Must have</td>
<td>No need</td>
</tr>
<tr>
<td>Stream cipher</td>
<td>Must have</td>
<td>Must have</td>
<td>Must have</td>
</tr>
</tbody>
</table>

Observations
- Cannot use a TRNG for a steam cipher (can for others)
- All applications need good statistical properties (uniformity, independence)
- In crypto applications, unpredictability is important

A warning when thinking about PRNGs

If numbers are computed, they aren’t random!

Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin. For, as has been pointed out several times, there is no such thing as a random number — there are only methods to produce random numbers, and a strict arithmetic procedure of course is not such a method. - John von Neumann, 1951

Computation cannot increase entropy
- 1000 bits output from a PRNG with 16-bit seed has at most 10 bits of entropy!
Good PRNG Importance

Security often fails just because of bad PRNG use

Can fail because of either:
- Bad seeding (not random or not large enough)
- Bad algorithm

Example 1: The original SSL implementation (Netscape Navigator)
- Seeded with process id (15 bits) and current time (a few bits or uncertainty)
- Made cryptographic keys guessable - completely destroyed security

Example 2: Bad algorithm in NIST standards - Dual EC DRBG
- Exposed as a possible backdoor after Snowden leaks

Dual EC DRBG

The potential backdoor is exposed

People have always worried about NSA backdoors - this one appears to have been real!

Was adopted by NIST as a standard.
Withdrawn from standard after discoveries

But… Dual EC DRBG is super-slow anyway - surely no one uses it… right?

Oops - people DID use it - maybe even unknowingly!

Not only used, but was the default DRBG in RSA’s BSAFE library!
Fast PRNG from a block cipher

Widely-used technique: CTR mode
- Key and initial counter are seed
- Basically the XOR pad from CTR mode (ignoring plaintext)

Key property: If AES-CTR mode is a secure encryption scheme (technically, is IND-CPA secure) then this is a secure PRNG

To think about: If K is fixed and secret (embedded in hardware) and only V is the seed, can it be "backdoored" (HW problem)