COMP 3704 Computer Security

Christian Grothoff
christian@grothoff.org

http://grothoff.org/christian/

Motivation

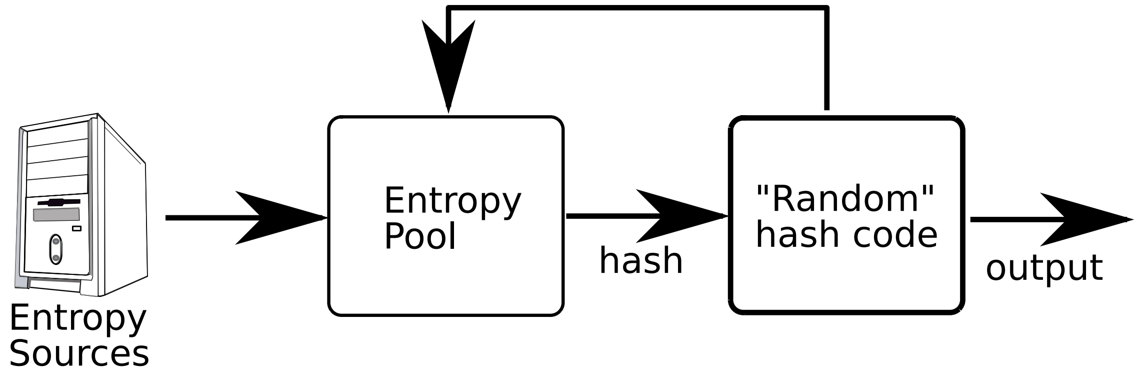
- How do we seed the PRNG algorithm?
- What if our PRNG is too predictable?

⇒ Need “physical” sources of randomness!

Security Requirements

- Randomness
- Forward Security (satisfied by PRNG)
- Backward Security (not satisfied by PRNG)

Basic Structure



Candidates from `/proc`

- Dirty pages
- Page faults
- Free pages
- System uptime
- Disk I/O (total number, merged, sectors, time spent)
- IO operations in progress
- Hardware interrupt counters

Candidates from /proc/net/

- Number of packets received
- Number of packets sent
- Transmission errors
- Network latency

Timing

- Date and time
- CPU cycle count:

```
__inline__ uint64_t rdtsc() {
  uint64_t x;
 __asm__ volatile ("rdtsc" : "=A" (x));
 return x;
}
```

- CPU performance counters (cache misses)
- Timing of events (keystrokes, mouse)

Entropy, again!

- For each entropy source, we would like to know how much entropy it actually provides!
- Estimating entropy is costly/difficult!
- Key idea: use **compression!**
- Compression estimates (non)randomness in data
- Can even use lossy data compression for estimation
- Compression ratio of 1:10 is typical!

Cheaper Entropy Estimation

Let t_n denote the timing of event number n. Define:

$$\delta_n = t_n - t_{n-1}$$
$$\delta_n^2 = \delta_n - \delta_{n-1}$$
$$\delta_n^3 = \delta_n^2 - \delta_{n-1}^2$$

Estimate the entropy added by event t_n to be:

$$\log_2(min(|\delta_n|, |\delta_n^2|, |\delta_n^3|)) \tag{1}$$

Main Design Choices

Choices for the RNG implementation:

- Non-blocking or blocking or both?
- Hashing method
- Entropy sources and entropy estimators
- Deterministic modifiers (counters)
- Pool size

End-user: Blocking, non-blocking or deterministic?

Usability Issues

A safety check to test whether the PRNG of OpenSSL was properly initialized was added to version 0.95. User responses were to seed it with:

- a constant
- output from rand()
- public key
- the executable
- /etc/passwd, /var/syslog

Usability Issues

Cryptographers: The device should refuse to work unless sufficient entropy is available.

Product developers: Cannot ship device that refuses to function.

- 0.01% of users have discipline to handle RNG failures
- 99.99% will see RNG failure as defective product

⇒ Actual security is a hard sell!

Questions

?

Problem

Design a protocol that does **not** use public key cryptography which allows Bob to prove his identity to Alice. Assume that Bob and Alice share a secret key K. Make sure that your protocol is safe against replay attacks; it does not have to be secure against a man-in-the-middle attack.

Homework Hint

- https://gnunet.org/svn/Extractor/src/plugins/hash/