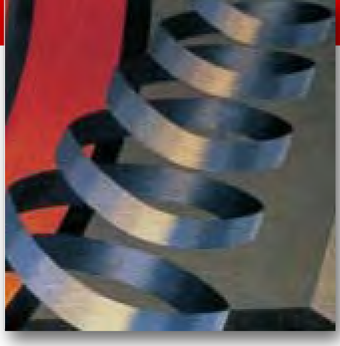
Is your randomness predictable?
(or, how to properly seed crypto libraries)

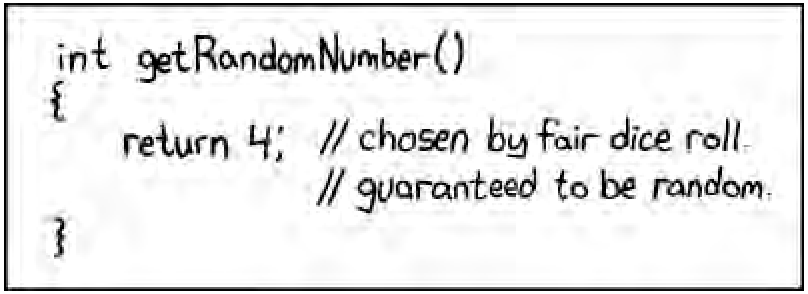
Keep Security Weird / BSides Austin, 2012

David Ochel
david@atsec.com



Agenda

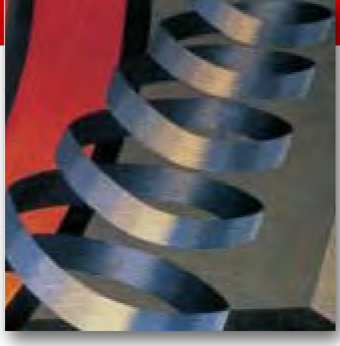
- Motivation
- Randomness, and how to generate it
- Threats against randomness (mistakes and attacks)
- Now what?



http://xkcd.com/221/
https://creativecommons.org/licenses/by-nc/2.5/

Disclaimer: I’m just a user.
I’m neither a cryptographer, nor a mathematician.

Motivation:
Assembling a crypto system

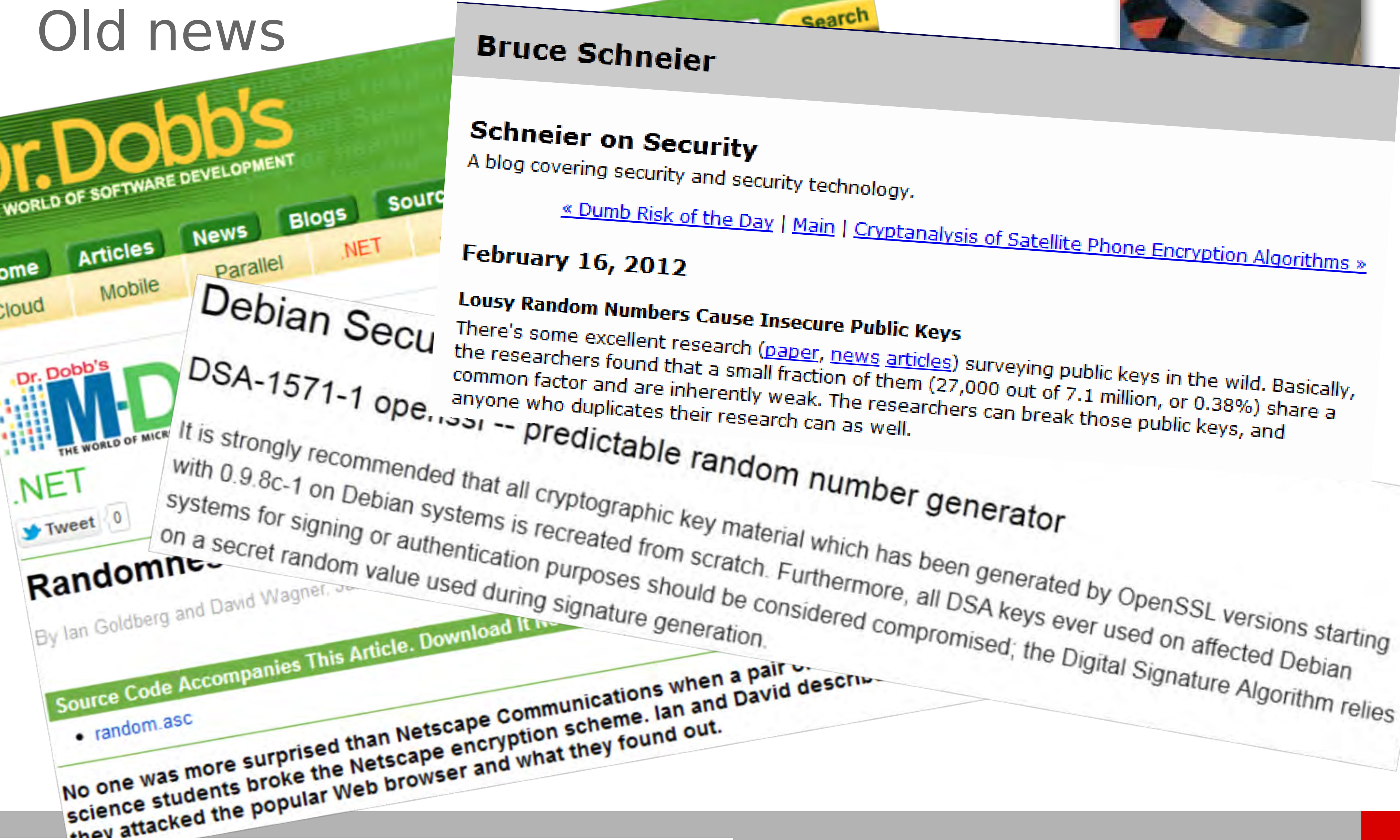


- The ~~2~~ 3 most important aspects in a crypto system:
  1. “Known-to-be-good” algorithm implementations
  2. Key management!
  3. High-quality keys => based on random (!) numbers
     - and nonces, IVs, salts, …

- Other applications:

…

Motivation: (cont.)
Old news



Bruce Schneier

Schneier on Security

A blog covering security and security technology.

« Dumb Risk of the Day | Main | Cryptanalysis of Satellite Phone Encryption Algorithms »

February 16, 2012

Lousy Random Numbers Cause Insecure Public Keys

There's some excellent research (paper, news articles) surveying public keys in the wild. Basically, the researchers found that a small fraction of them (27,000 out of 7.1 million, or 0.38%) share a common factor and are inherently weak. The researchers can break those public keys, and anyone who duplicates their research can as well.

Debian Secu...

DSA-1571-1 ope...ssl -- predictable random number generator

It is strongly recommended that all cryptographic key material which has been generated by OpenSSL versions starting with 0.9.8c-1 on Debian systems is recreated from scratch. Furthermore, all DSA keys ever used on affected Debian systems for signing or authentication purposes should be considered compromised; the Digital Signature Algorithm relies on a secret random value used during signature generation.

Dr. Dobb's

THE WORLD OF SOFTWARE DEVELOPMENT

Search

Home | Articles | News | Blogs | Source...

Cloud | Mobile | Parallel | .NET

Dr. Dobb's M-D...

.NET

Tweet 0

Randomne...

By Ian Goldberg and David Wagner...

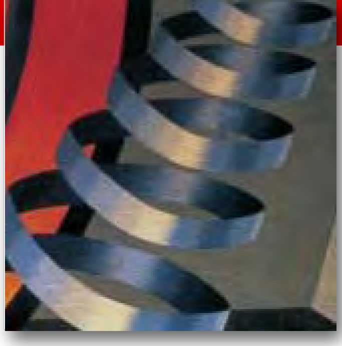
Source Code Accompanies This Article. Download It N...

- random.asc

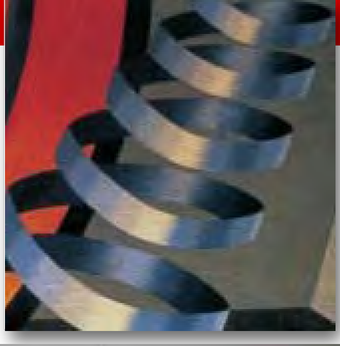
No one was more surprised than Netscape Communications when a pair o... science students broke the Netscape encryption scheme. Ian and David descri... they attacked the popular Web browser and what they found out.

https://www.schneier.com/blog/archives/2012/02/lousy_random_nu.html
http://www.debian.org/security/2008/dsa-1571
http://drdobbs.com/windows/184409807

Randomness



- Random?
  - uniform probability distribution
- Entropy
  - measure of randomness, i.e.
    uncertainty about information before observing an event
  - result of flipping a “fair” coin: 1 bit of entropy
  - time between random key strokes: ???
    - OS sampling rate of keyboard inputs
    - likelihood of the user’s typing patterns being predictable
    - ...
- **<u>Goal: not (efficiently) predictable by an attacker!!</u>**

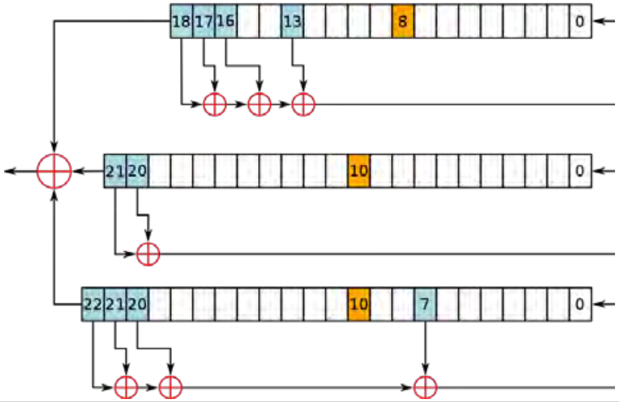


Random Number Generation

- True random data:
  - observation of unpredictable physical / environmental phenomena: atmospheric noise, radio-active decay, lava lamps, thermal noise, …
  - arguably: execution time of processes, disk head movements, …

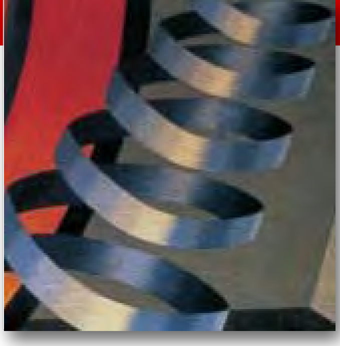
RANDOM.ORG anno 1998 (historic)
http://www.random.org/history/

- Pseudo-random data:
  - algorithms expand initial state into a sequence of derived numbers with certain amount of statistical distribution



stream cipher

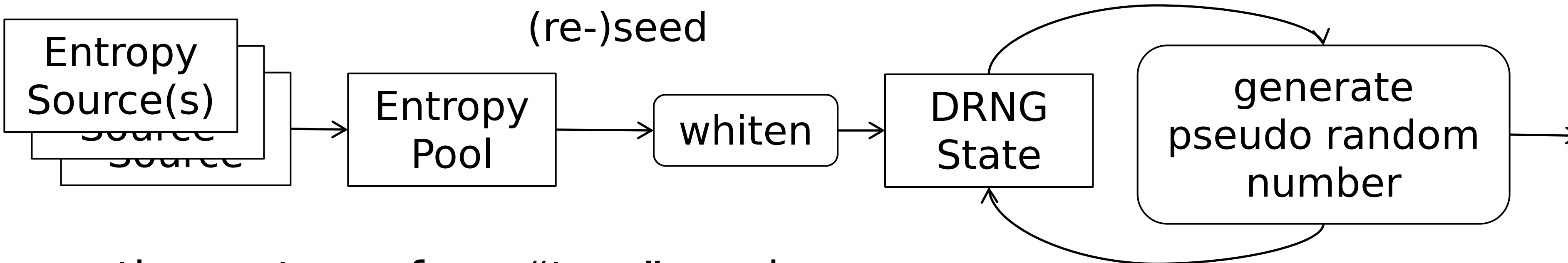
True vs. deterministic random numbers



- “true” RNGs
  - observation of external events
  - can estimate lower bound of entropy
  - hard to predict (needs to be justified)
  - sometimes unreliable

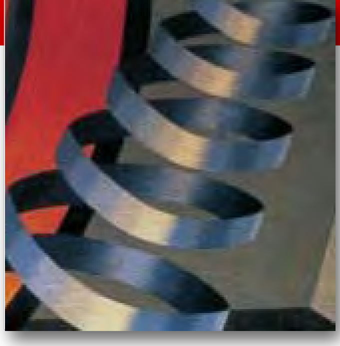
- “pseudo” RNGs
  - based on algorithmic computation
  - require a “seed” to obtain entropy
  - deterministic (i.e., reliable)

The best of both worlds

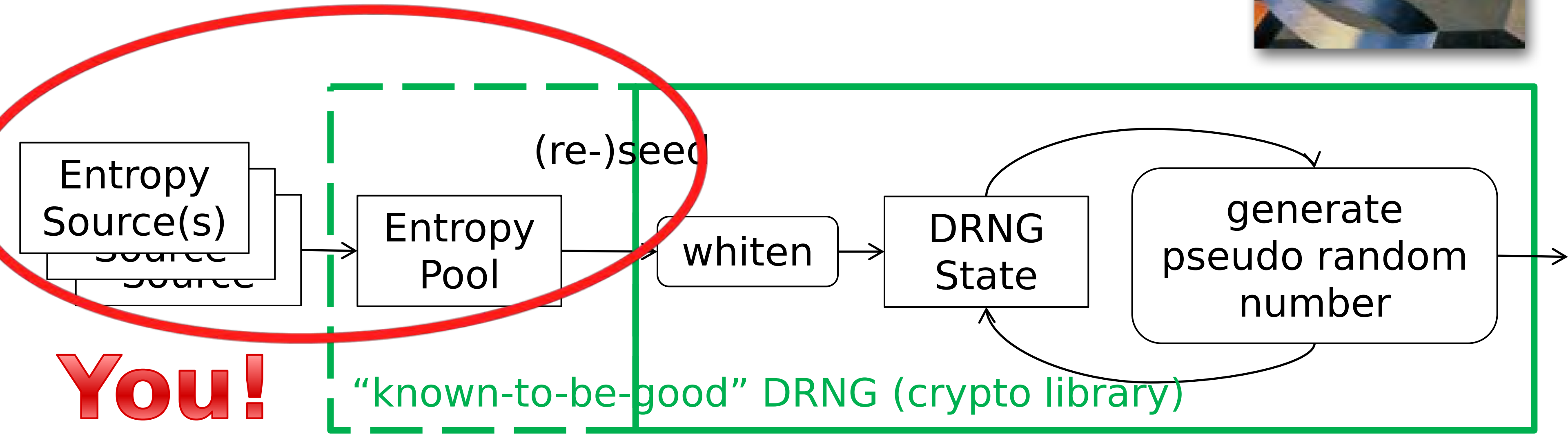


- gather entropy from “true” random sources
  - hard to predict
  - use to seed DRNG (aka DRBG – bit generator)

- use (cryptographically secure) DRNG to:
  - “whiten” entropy input (remove unwanted properties)
  - “spread“ seed into larger amounts of outputs

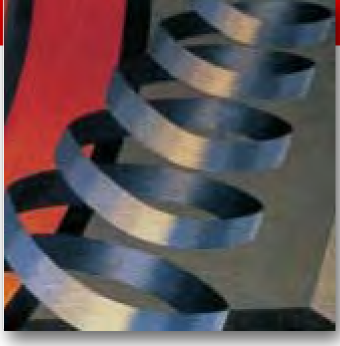


Why am I telling you all this?



- general purpose computing crypto libraries can’t account for the quality of the entropy they might encounter on a system!
- You need to ensure your RNGs are properly seeded!
  - if you feed a DRNG 13 bits of entropy, your 128-bit key will have at most 13 bits of entropy! (looks random, but isn’t really)

Common mistakes



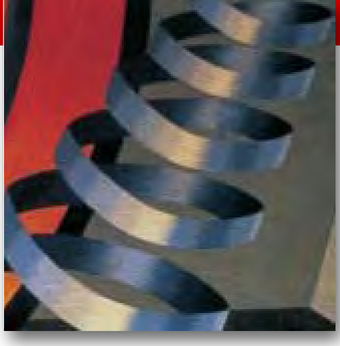
- Insufficient seeding
  - using low-entropy sources: solid-state (non-spinning) disks, time of day, fixed variables, …
  - overestimating the real entropy of sources
  - seeding during bootstrapping (or, full disk encryption!)
  - not re-seeding often enough

- Insufficient tests of the “liveliness” of entropy sources
  - sources may die, become biased, be tampered with, …

Common Mistakes (continued)

- Relying on tests of DRNG output
  - it will look random, regardless of the entropy of the seed
- Insufficient protection of the “entropy pool”
  - in the running system
  - when stored on disk (between re-boots)
- No consideration of other external threats

Some external threats…

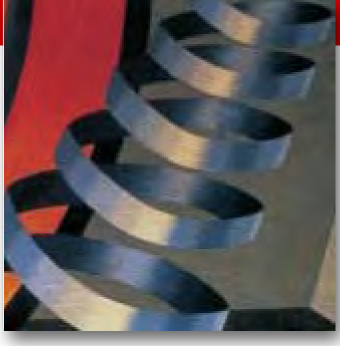


- direct tampering with software parts of RNG

- observation of (initial) state/seeds
- predicting (guessing) input from “random” entropy sources:
  - system time; Ethernet MAC address; …
- influencing value of “random” seeds:
  - network traffic; temperature
- seed source failure
  - transistors die, source properties may change

- exhaustive guessing
  - if that’s the only concern left, you win!

What to do?



1. Use “known-to-be-good” libraries
   – …and read their documentation!

2. Seed with high-entropy source
   – /dev/random (not /dev/urandom) (estimates entropy)
   – haveged - http://www.issihosts.com/haveged/ (time jitters)
   – EntropyKey - http://www.entropykey.co.uk/ (“avalanche noise”)
   – Intel RdRand (?)
   – … (your mileage may vary!)

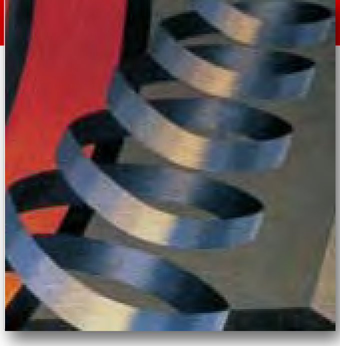
What to do? (continued)

3. Re-seed on a regular basis

4. Save RNG state or entropy pool during re-boot
   - requires appropriate protection

5. Identify and mitigate external threats

References



- RFC 1750 [1994]: Randomness Recommendations for Security
  https://www.ietf.org/rfc/rfc1750.txt

- Menezes, et al. [1996]: Handbook of Applied Cryptography,
  ISBN 0-8493-8526-7

- Ferguson, Schneier [2003]: Practical Cryptography
  ISBN 0-471-22894-X