

RSA[®]CONFERENCE2014

FEBRUARY 24 – 28 | MOSCONE CENTER | SAN FRANCISCO

Share.
Learn.
Secure.

Capitalizing on
Collective Intelligence

Entropy, Random Numbers And Keys: What's Good Enough?

SESSION ID: ASEC-T07A

John Leiseboer

CTO

QuintessenceLabs

JL@quintessencelabs.com



NSA's broken Dual_EC random number generator has a "fatal bug" in OpenSSL

No plans to fix a bug in "toxic" algorithm that no one seems to use.

by Dan Goodin - Dec 20 2013, 7:05am +1000

Security Analysis of Pseudo-Random Number Generators with Input: /dev/random is not Robust

Yevgeniy Dodis¹, David Pointcheval², Nicolas Vergnaud², and Daniel Wichs⁴

Stealthy Dopant-Level Hardware Trojans

Francesco Regazzoni², Christopher P. Burleson¹

Fatal crypto flaw in some government-certified smartcards makes forgery a snap

With government certifications this broken, the NSA may not need backdoors.

by Dan Goodin - Sept 17 2013, 1:25am +1000

GOVERNMENT HACKING 66



On the Possibility of a Back Door in the NIST SP800-90 Dual Ec Prng

Dan Shumow
Niels Ferguson
Microsoft

MIT Research: Encryption Less Secure Than We Thought

Posted by Soulskill on Wednesday August 14, 2013 @02:50PM
from the but-still-pretty-darn-secure dept.



A group of researchers from MIT and the University of Ireland has presented [a paper](#) (PDF) showing that [one of the most important](#) [things behind cryptographic security is wrong](#). As a result, certain encryption-breaking methods will work better than previously

Deliberately flawed? RSA Security tells customers to drop NSA-related encryption algorithm

Published time: September 20, 2013 10:31
Edited time: September 20, 2013 13:15

[Get short URL](#)



Problem statement

- ◆ There are deployed systems with shared keys or common factors
- ◆ There are evaluated and certified products with RNG weaknesses
- ◆ Snowden leaks have triggered a crisis in confidence
 - ◆ NSA recommended Dual_EC_DRBG for NIST SP 800-90A standard
 - ◆ Concern over potential backdoors in hardware
 - ◆ Concern over compromised software
- ◆ Concerns about /dev/urandom
- ◆ Many APIs poorly documented

Uses of random

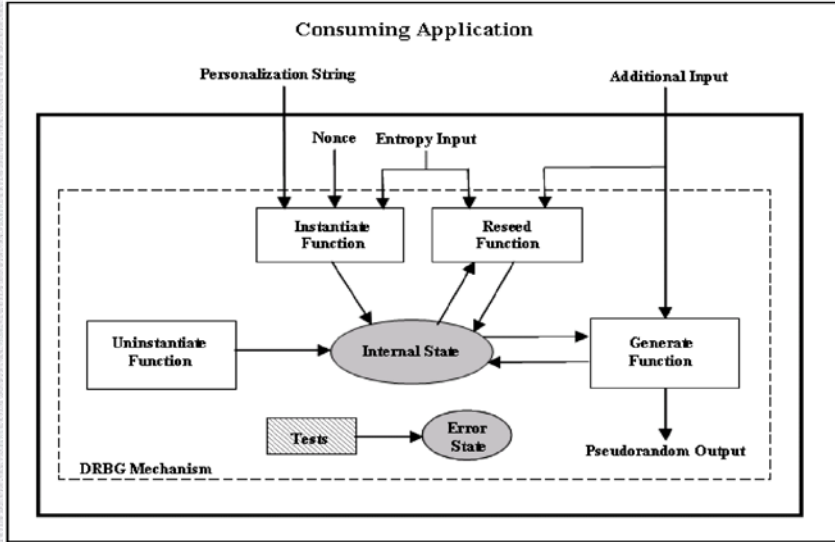
“Traditional” uses

- ◆ RNG seed
- ◆ Key generator seed
- ◆ IV
- ◆ Nonce
- ◆ Random challenge
- ◆ Authentication
- ◆ DSA signing

Other uses

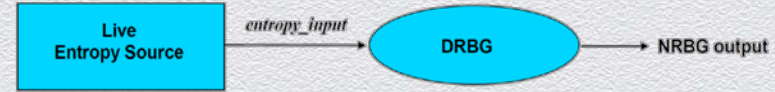
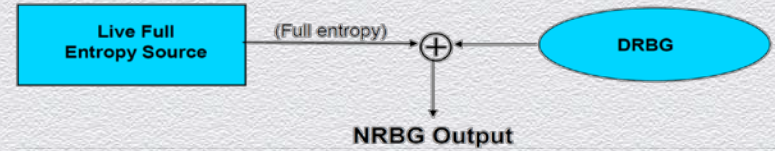
- ◆ One-time pad cipher
- ◆ Zero knowledge proof
- ◆ E-voting
- ◆ Random beacon
 - ◆ Transaction protection
 - ◆ PII protection
 - ◆ Cloud entropy

RNG Construction

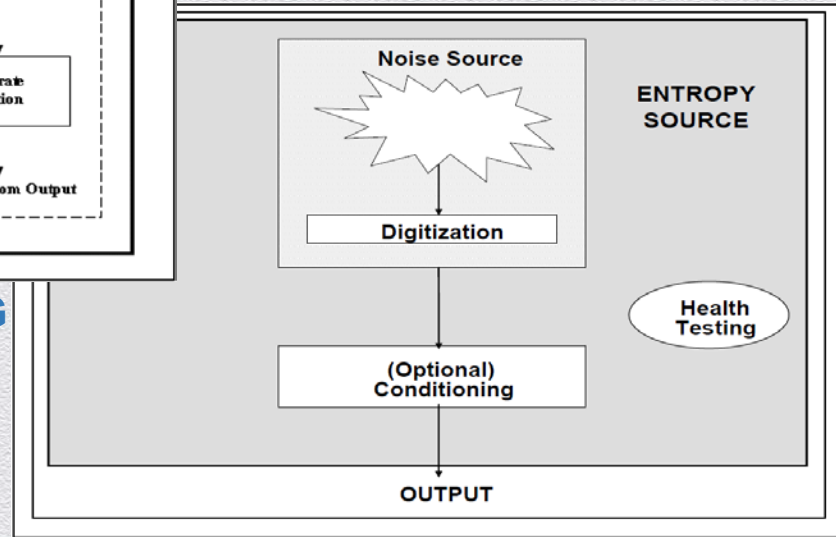


NIST SP 800-90A - DRBG

`/dev/random`
`/dev/urandom`



NIST SP 800-90C – NRBG



NIST SP 800-90B – Entropy Source

Estimating entropy

- ◆ 9 9 9 9 9 9 9 9 9 9 9 9 9 ...
- ◆ 3 1 4 1 5 9 2 6 5 3 5 9 ...
- ◆ Independent and identical distribution (IID)

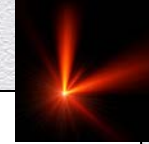
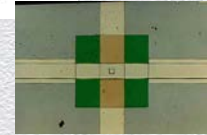
Non-i.i.d. Min-Entropy Estimation Test Results:

Collision test	4.310757 bits per 8-bit symbol
Partial collection test	2.467009 bits per 8-bit symbol
Markov test	5.692803 bits per 6-bit symbol (remapped due to test limit)
Compression test	3.401208 bits per 8-bit symbol
Frequency test	6.699898 bits per 8-bit symbol

Sanity Check Results:

Compression test	passed - (840168 bits, 840800 bits, 840336 bits, 839440 bits, 840688 bits, 839776 bits, 840312 bits, 839688 bits, 839800 bits, 840016 bits)
Collision test	passed - (0 13-symbol values with a count of 3 or more, 0 colliding 13-symbol values in total)

Min-entropy estimate is 2.467009 bits per 8-bit symbol, based on a 95% confidence interval.



Random number tests

- ◆ NIST STS

- ◆ <http://csrc.nist.gov/groups/ST/toolkit/rng/index.html>
- ◆ *“...no set of statistical tests can absolutely certify a generator as appropriate...”*

- ◆ Dieharder

- ◆ <http://www.phy.duke.edu/~rgb/General/dieharder.php>
- ◆ *“dieharder is a tool designed to permit one to push a weak generator to unambiguous failure ...”*

Practical issues

- ◆ Entropy sources
- ◆ Hardware and software implementations
- ◆ Standards compliance
- ◆ Performance requirements
- ◆ Security requirements
- ◆ User experience – application developer and end-user
- ◆ Trust

Application interfaces


- ◆ PKCS#11: C_GenerateRandom()
 - ◆ Java: SecureRandom()
 - ◆ Microsoft CAPI and CNG: CryptGenKey()
 - ◆ OpenSSL: RAND_bytes()
 - ◆ Others
-
- ◆ OASIS KMIP – client/server network protocol

Recommendations

- ◆ Entropy
 - ◆ Identify sources of entropy and assess min-entropy
- ◆ RBG construction
 - ◆ Ensure that the RBG contains required and approved components
- ◆ Seeding and re-seeding PRNGs
 - ◆ Seed and re-seed PRNG as required to meet security requirements
- ◆ Use the API correctly
 - ◆ Use the correct functions in the correct order

RSA[®]CONFERENCE2014

FEBRUARY 24 – 28 | MOSCONE CENTER | SAN FRANCISCO



John Leiseboer
QuintessenceLabs
JL@quintessencelabs.com