

PRNGs

Pseudo-random number generation

Randomness and Cryptography

- Randomness and pseudo-randomness are useful in cryptography:
 - To generate random and pseudo-random keys and initialization vectors for use with block ciphers;
 - To generate pseudo-random key streams for use with stream ciphers;
 - For challenge/response strong authentication protocols;
 - With several randomized cryptographic algorithms, such as the generation of public keys in the RSA algorithm.

Randomness vs. unpredictability

- True randomness is, of course, unpredictable. Within a computer, true randomness requires extra input:
 - Physical measurements of stochastic phenomena that normally occur in the standard hardware;
 - Use of specialized hardware, or;
 - Exploiting randomness of user interaction, etc.
- Pseudo-randomness, or the generation of random-looking deterministic sequences, may not necessarily be unpredictable.

Linear Congruential Generator

- A LCG is defined by an integer triple (A, B, M) , with the pseudo-random sequence (S_n) given by the recurrence relation:
 - $S_{n+1} = A S_n + B \text{ mod } M$
- Pros: Fast, short description makes it sometimes only alternative in embedded systems
- Cons: Easily crypto-analyzed (**completely insecure**), fails a statistical test for randomness (spectral test):
 - Should NEVER be used in secure applications.
 - Should not be used for simulation experiments requiring high randomness quality (e.g. Monte-Carlo methods).

Shift Feedback Registers

- Mersenne Twister:
 - word length w , and $0 < r < w$.
 - Given word W :
 - W^u = upper $w-r$ bits of W ,
 - W^l = lower r bits.
 - positive integers n, m , ($m < n$), and binary matrix A .
 - $X_{k+n} = X_{k+m} \oplus (X_k^u || X_{k+1}^l)A$
- Fast, generates good-quality sequences, but not cryptographically secure.
- It is an improved generator of a class of *feedback shift registers*.
- Linear Shift Feedback Registers (LSFR) are simpler representatives of this class that are **quite insecure**, though often used in cryptographic algorithms (example, the GSM standard), which are *then subsequently broken*.

Cryptographically Strong PRNGs

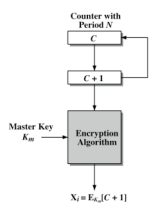


Figure 7.13 Pseudorandom Number Generation From a Counter

- **Cyclic encryption** uses a cyclically incremented counter.
- This algorithm can be used to generate other encryption keys from a Master Key.

ANSI X9.17 PRNG

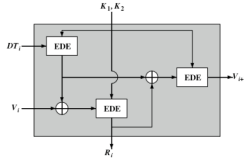


Figure 7.14 ANSI X9.17 Pseudorandom Number Generator

- Here, 3-DES EDE with two keys are used (same master keys for 3 encryption boxes).
 - DT_i = 64-bit representation of current date/time,
 - V_i = internal register, initialized with a seed.
 - R_i = generated pseudo-random value.

Blum-Blum-Shub Generator

- Two large primes p, q , with $p = q = 3 \pmod 4$. Large means $\lceil \log_2 |p| \rceil \geq 512$.
- Let $n = p q$.
- BBS(Seed s , Big n) {
 - $i \leftarrow 0; X_0 \leftarrow s^2 \pmod n$;
 - while(Generating) {
 - $X_{i+1} \leftarrow (X_i)^2 \pmod n; i \leftarrow i+1$;
 - output $B_{i+1} \leftarrow X_{i+1} \pmod 2$
 - // lower bit of X_{i+1}

Key use

- To prevent large amounts of data being encrypted with a single key, it is good practice to use a *key hierarchy*.
- A *master key* is used to derive/protect random or pseudo-random *session keys*.
- Each communication/ data encryption session is encrypted under a different session key.

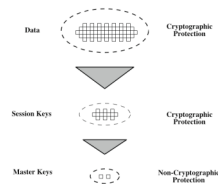


Figure 7.8 The Use of a Key Hierarchy

Key distribution

- Symmetric *master keys* must be delivered by some secure method.
 - Hardware-embedded keys
 - Administratively entered by human
 - Use of a trusted intermediary
- Session-key distribution mechanisms:
 - Generate same keys on both ends via pseudo-random sequences (from master keys)
 - Use a key distribution center (KDC)
 - Encrypt new key under old key (for key replacement)
 - Using public key cryptography/ key exchange protocols

Introduction to Public-Key Cryptography

