Class229: Pseudo-Random Generators

Virendra Sule

Practical alternatives to Perfect secrecy

Perfect secrecy requires exchange of a random key stream for every session which makes it difficult to use in practice. What can be Practical alternatives to Perfect Secrecy?

- Using a cipher which satisfies perfect secrecy approximately. For the finite sample size of plaintexts the condition for perfect secrecy is statistically verified. True distinction between $pr(P|C)$ and $pr(P)$ is realised for very large plaintext spaces and should not be feasible in polynomial time.
- Use a PR keystream such that it satisfies randomness tests and is not distinguishable from random stream in practically feasible time (and memory space for computation).
- If $s \in \{0,1\}^n$ is a truly random short seed (a random string which cannot be predicted) a PR generator is a map $G : \{0,1\}^n \rightarrow \{0,1\}^{l(n)}$ which generates a sequence of length $l(n)$ which cannot be distinguished from a random sequence of length $l(n)$ in polynomial time.

Conditions for a PR sequence

- Long period, no repetitions.
- Linear complexity: large, linear complexity profile close to line of 1/2 slope.
- uniform k-tuple distributions, runs of k bits uniformly distributed.
- Confusion: every bit of the stream is a complex transformation of bits of the seed and IV.
- Diffusion: seed and IV dissipated upto long range statistics.
- Nonlinearity criteria: correlation with m bits, distance from a linear function, avalanche criteria etc.

These are not proofs of security of the PR generator and separate cryptanalysis is required. Most of the tests are statistical hence only empirical evidence of randomness is gathered.

Some examples of PRGs

- Muliple bit generator: patented by NSA. A stream generator like a FSR is used. At time k, bits $y_1, y_2, \ldots, y_m$ of the output of FSR are loaded in a shift register R. A random control m-bit vector $(v_1, \ldots, v_m)$ is IV.

$$w_i = y_i \wedge v_i \; z = w_1 \oplus \ldots \oplus w_m$$

  z is the output at time k. Then R is shifted right and new bit is pushed in. Multiple output streams are generated using different random control vectors.
- Cellular automaton generator: Initial array of bits $a_1, a_2, \ldots, a_n$. Update function

$$a'_k = a_{(k-1)} \oplus (a_k \vee a_{(k+1)}$$

- Blum-Micali generator. g is a primitive root of p. p is a large prime. Key x_0.

$$x_{(i+1)} = g^{(x_i)} \mod p$$

  Output is 1 if $x_i < (P-1)/2$, 0 otherwise.

- Blum Blum Shub generator. Depends on the theory of quadratic residuosity. Blum integer $n = pq$ large primes such that $p, q = 3 \mod 4$. x another random integer coprime to n.

$$\begin{aligned} x_0 &= x^2 \mod n \\ x_i &= x_{(i-1)}^2 \mod n \end{aligned}$$

  i-th output bit $b_i = \text{lsb}\, x_i$.

Study important PR generators and their security for the next assignment submission.

Using PR streams for encryption

- Let $G(.)$ generates a PR sequence r of length $l(n)$ for a true random input seed s of n-bits.
- Note that $r = G(s)$ is far from truly random sequence, since the range of G cannot exceed 2^n while the number of random sequences of length $l(n)$ are $2^{l(n)}$.
- However, if a battery of statistical tests running in polynomial time cannot distinguish r from a random sequence then G is an acceptable PR generator.

To encrypt a plaintext stream $\{p_i\}$, the Vernam cipher is

1. choose a random IV and seed s.
2. compute stream $c_i = p_i \oplus G(s||IV)$
3. ciphertext stream $C = IV||\{c_i\}$.

OWFs using PR sequences

- Define a function $F : \{0,1\}^n \rightarrow \{0,1\}^m$. Consider s to be an n-bit random seed. Compute the sequence

$$y(k) = F(s+k), k = 0, 1, 2, \ldots$$

  where $s + k$ is the integer addition. Repeat the sequence $s + k$ after $s + k$ reaches 2^n.
- $F(.)$ is called a OWF if the sequence $y(k)$ is PR and there is no other practically feasible way known to compute x given $y = F(x)$.
- $F(.)$ can be used for encryption in many different ways.

Practical security

Security in the sense of Perfect Secrecy is not practical. However approximatation of Perfect Secrecy in the sense of computational infeasibility is a practical way to carry out secure encryption. However such encryption is without proof of security. This is another issue to explore, "What is meant by proof of security"?