Pseudorandom generator (PRG)

Pseudorandom Generator (PRG)

- Given a seed, it outputs a sequence of random bits

  PRG(seed) -> random bits

- It can output arbitrarily many random bits

PRG security

- Can PRG(K) be truly random?

No. Consider key length |K|=k. Have 2^k possible initial states of PRG. Deterministic from then on. There are more random states.

- A secure PRG suffices to “look” random (“pseudo”) to an attacker (no attacker can distinguish it from a random sequence)

Example of PRG: using block cipher in CTR mode

If you want m random bits, and a block cipher with E_k has n bits, apply the block cipher m/n times and concatenate the result:

PRG(K | IV) = E_k(IV|1) | E_k(IV| 2) | E_k(IV|3) … E_k(IV| ceil(m/n)), where | is concatenation

Application of PRG: Stream ciphers

- Another way to construct encryption schemes
- Similar in spirit to one-time pad: it XORs the plaintext with some random bits
- But random bits are not the key (as in one-time pad) but are output of a pseudorandom generator PRG

Application of PRG: Stream cipher

Enc(K, M):

- Choose a random value IV
- C = PRG(K | IV) XOR M
- Output (IV, C)

Q: How decrypt?

A: Compute PRG(K | IV) and XOR with ciphertext C

Q: What is advantage over OTP?

A: Can encrypt any message length because PRG can produce any number of random bits, and multiple times because IV is chosen at random in Enc