Pseudorandom number generation

Dr. Shubhangi Sapkal

Pseudorandom number generator (PRNG)

- An important cryptographic function is cryptographically strong pseudorandom number generation. Pseudorandom number generators (PRNGs) are used in a variety of cryptographic and security applications.

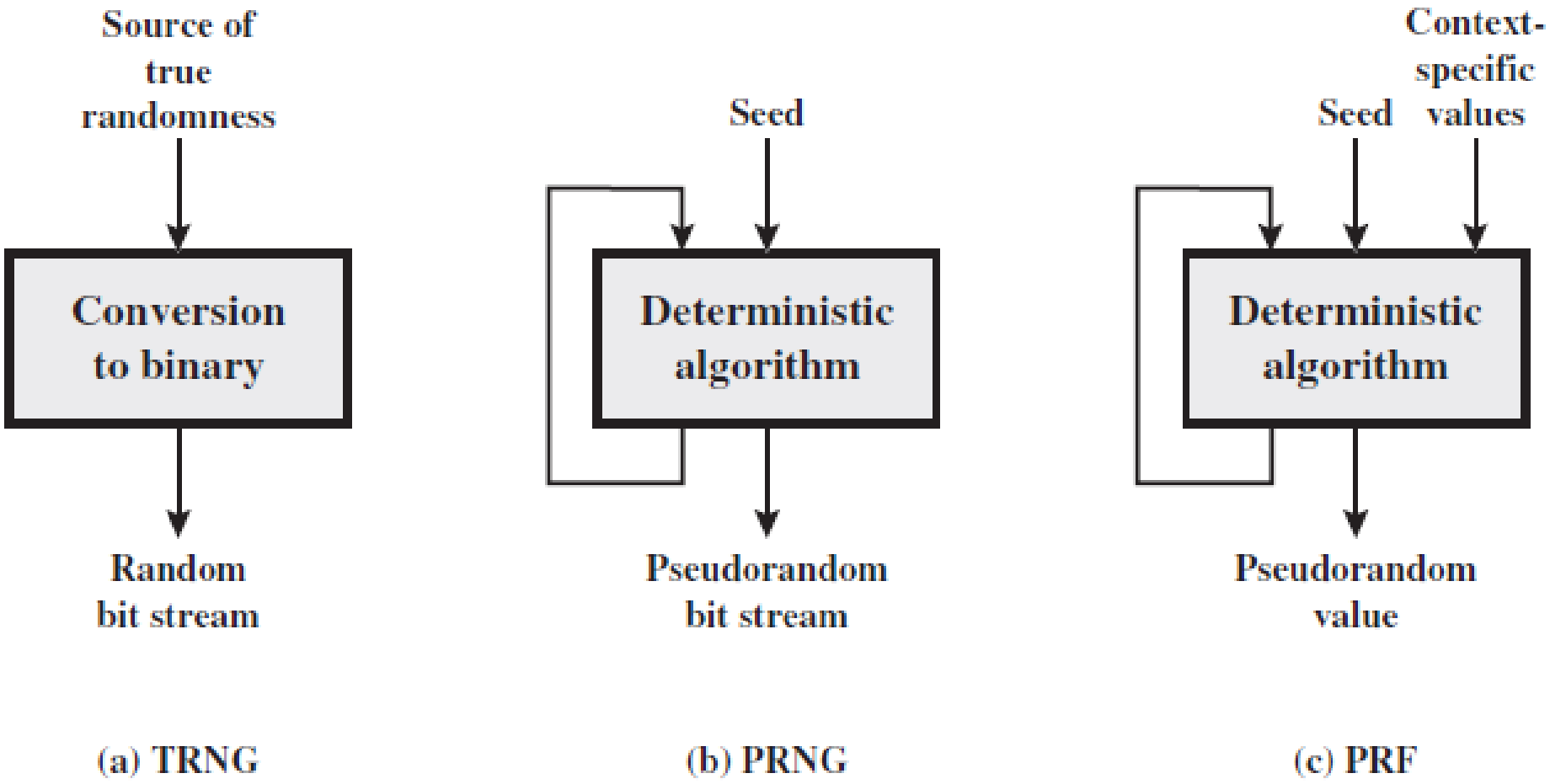
TRNGs, PRNGs, and PRFs

- Cryptographic applications typically make use of algorithmic techniques for random number generation.
- These algorithms are deterministic and therefore produce sequences of numbers that are not statistically random. However, if the algorithm is good, the resulting sequences will pass many tests of randomness. Such numbers are referred to as pseudorandom numbers.

Pseudorandom number generator (PRNG) & TRNG

- True random number generator (TRNG) takes as input a source that is effectively random; the source is often referred to as an entropy source.
- The entropy source is drawn from the physical environment of the computer and could include things such as keystroke timing patterns, disk electrical activity, mouse movements, and instantaneous values of the system clock.
- The source, or combination of sources, serve as input to an algorithm that produces random binary output. The TRNG may simply involve conversion of an analog source to a binary output.

- In contrast, a PRNG takes as input a fixed value, called the seed, and produces a sequence of output bits using a deterministic algorithm.
- Quite often, the seed is generated by a TRNG.
- The output bit stream is determined solely by the input value, so that an adversary who knows the algorithm and the seed can reproduce the entire bit stream.



TRNG = true random number generator
PRNG = pseudorandom number generator
PRF = pseudorandom function

Figure 7.1 Random and Pseudorandom Number Generators

Pseudorandom number generator (PRNG)

Two different forms of PRNGs, based on application.

- Pseudorandom number generator: An algorithm that is used to produce an open-ended sequence of bits is referred to as a PRNG. A common application for an open-ended sequence of bits is as input to a symmetric stream cipher
- Pseudorandom function (PRF): A PRF is used to produce a pseudorandom string of bits of some fixed length. Examples are symmetric encryption keys and nonces. Typically, the PRF takes as input a seed plus some context specific values, such as a user ID or an application ID.

PRNG Requirements

- When a PRNG or PRF is used for a cryptographic application, then the basic requirement is that an adversary who does not know the seed is unable to determine the pseudorandom string.
- For example, if the pseudorandom bit stream is used in a stream cipher, then knowledge of the pseudorandom bit stream would enable the adversary to recover the plaintext from the ciphertext.
- Similarly, we wish to protect the output value of a PRF.
- Randomness: In terms of randomness, the requirement for a PRNG is that the generated bit stream appear random even though it is deterministic.

PRNG Requirements

- Uniformity: At any point in the generation of a sequence of random or pseudorandom bits, the occurrence of a zero or one is equally likely, that is, the probability of each is exactly 1/2. The expected number of zeros (or ones) is n/2, where n = the sequence length.
- Scalability: Any test applicable to a sequence can also be applied to subsequences extracted at random. If a sequence is random, then any such extracted subsequence should also be random. Hence, any extracted subsequence should pass any test for randomness.
- Consistency: The behavior of a generator must be consistent across starting values (seeds). It is inadequate to test a PRNG based on the output from a single seed or an TRNG on the basis of an output produced from a single physical output.

PRNG Requirements

- Unpredictability: A stream of pseudorandom numbers should exhibit two forms of unpredictability:
  - Forward unpredictability: If the seed is unknown, the next output bit in the sequence should be unpredictable in spite of any knowledge of previous bits in the sequence.
  - Backward unpredictability: It should also not be feasible to determine the seed from knowledge of any generated values. No correlation between a seed and any value generated from that seed should be evident; each element of the sequence should appear to be the outcome of an independent random event whose probability is 1/2.

Seed Requirements

- For cryptographic applications, the seed that serves as input to the PRNG must be secure. Because the PRNG is a deterministic algorithm, if the adversary can deduce the seed, then the output can also be determined. Therefore, the seed must be unpredictable.
- In fact, the seed itself must be a random or pseudorandom number. Typically, the seed is generated by a TRNG.
- If a TRNG is available, why it is necessary to use a PRNG. If the application is a stream cipher, then a TRNG is not practical. The sender would need to generate a keystream of bits as long as the plaintext and then transmit the keystream and the ciphertext securely to the receiver.
- If a PRNG is used, the sender need only find a way to deliver the stream cipher key, which is typically 54 or 128 bits, to the receiver in a secure fashion.

Algorithm Design

- Cryptographic PRNGs have been the subject of much research over the years, and a wide variety of algorithms have been developed. These fall roughly into two categories.
- Purpose-built algorithms: These are algorithms designed specifically and solely for the purpose of generating pseudorandom bit streams. Some of these algorithms are used for a variety of PRNG applications. The most important example is RC4
- Algorithms based on existing cryptographic algorithms: Cryptographic algorithms have the effect of randomizing input data. Indeed, this is a requirement of such algorithms. For example, if a symmetric block cipher produced ciphertext that had certain regular patterns in it, it would aid in the process of cryptanalysis. Thus, cryptographic algorithms can serve as the core of PRNGs.

Algorithm Design

- Three broad categories of cryptographic algorithms are commonly used to create PRNGs: —
  Symmetric block ciphers
  Asymmetric ciphers
  Hash functions and message authentication codes

Pseudorandom number generation using a bLock cipher

- A popular approach to PRNG construction is to use a symmetric block cipher.
- For any block of plaintext, a symmetric block cipher produces an output block that is apparently random. That is, there are no patterns or regularities in the ciphertext that provide information that can be used to deduce the plaintext.
- Thus, a symmetric block cipher is a good candidate for building a pseudorandom number generator. If an established, standardized block cipher is used, such as DES or AES, then the security characteristics of the PRNG can be established.

PRNG using Block Cipher modes of Operation

- Two approaches that use a block cipher to build a PNRG :
  1) the CTR mode
  2) The OFB mode
  In each case, the seed consists of two parts: the encryption key value and a value V that will be updated after each block of pseudorandom numbers is generated. Thus, for AES-128, the seed consists of a 128-bit key and a 128-bit V value. In the CTR case, the value of V is incremented by 1 after each encryption. In the case of OFB, the value of V is updated to equal the value of the preceding PRNG block.

Ex:

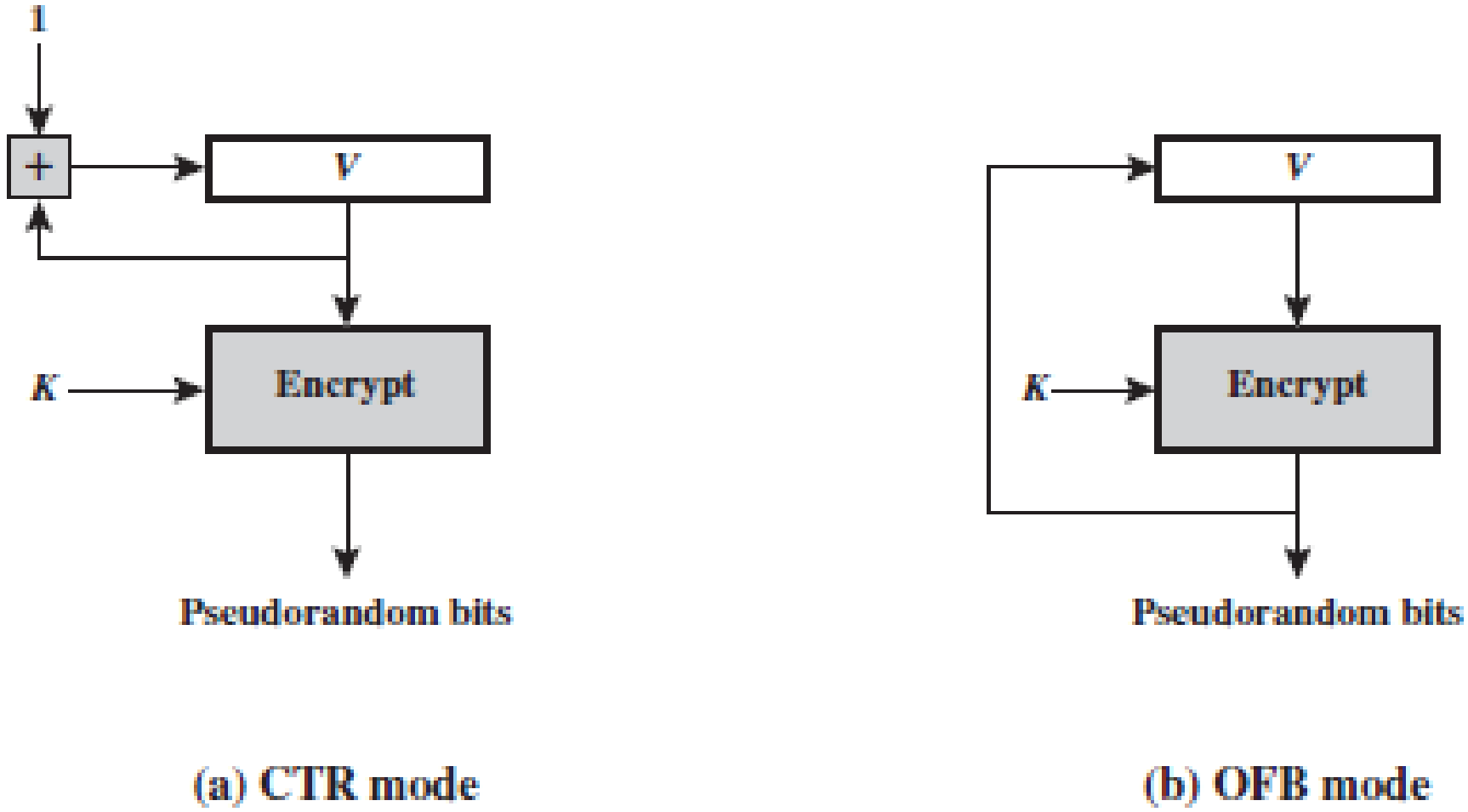
- A random bit sequence of 256 bits was obtained from random.org, which uses three radios tuned between stations to pick up atmospheric noise. These 256 bits form the seed, allocated as
- Key: cfb0ef3108d49cc4562d5810b0a9af60 V: 4c89af496176b728ed1e2ea8ba27f5a4
- The total number of one bits in the 256-bit seed is 124, or a fraction of 0.48, which is reassuringly close to the ideal of 0.5. For the OFB PRNG, Table 1 shows the first eight output blocks (1024 bits) with two rough measures of security. The second column shows the fraction of one bits in each 128-bit block. This corresponds to one of the NIST tests. The results indicate that the output is split roughly equally between zero and one bits.

Output Block	Fraction of One Bits	Fraction of Bits that Match with Preceding Block
1786f4c7ff6e291dbdfdd90ec3453176	0.57	—
5e17b22b14677a4d66890f87565eae64	0.51	0.52
fd18284ac82251dfb3aa62c326cd46cc	0.47	0.54
c8e545198a758ef5dd86b41946389bd5	0.50	0.44
fe7bae0e23019542962e2c52d215a2e3	0.47	0.48
14fdf5ec99469598ae0379472803accd	0.49	0.52
6aeca972e5a3ef17bd1a1b775fc8b929	0.57	0.48
f7e97badf359d128f00d9b4ae323db64	0.55	0.45

Table 1: Example results for PRNG using OFB

Output Block	Fraction of One Bits	Fraction of Bits that Match with Preceding Block
1786f4c7ff6e291dbdfdd90ec3453176	0.57	—
60809669a3e092a01b463472fdcae420	0.41	0.41
d4e6e170b46b0573eedf88ee39bff33d	0.59	0.45
5f8fcfc5deca18ea246785d7fadc76f8	0.59	0.52
90e63ed27bb07868c753545bdd57ee28	0.53	0.52
0125856fdf4a17f747c7833695c52235	0.50	0.47
f4be2d179b0f2548fd748c8fc7c81990	0.51	0.48
1151fc48f90eebac658a3911515c3c66	0.47	0.45

Table 2: Example results for PRNG using CTR



(a) CTR mode

(b) OFB mode

Figure 7.4 PRNG Mechanisms Based on Block Ciphers

The CTR algorithm for PRNG, called CTR_DRBG, can be summarized as follows.

```
while (len (temp) < requested_number_of_bits) do
        V = (V + 1) mod 2^128.
        output_block = E(Key, V)
        temp = temp || ouput_block
```

The OFB algorithm can be summarized as follows.

```
while (len (temp) < requested_number_of_bits) do
        V = E(Key, V)
        temp = temp || V
```

ANSI X9.17 PRNG

- One of the strongest (cryptographically speaking) PRNGs is specified in ANSI X9.17. A number of applications employ this technique, including financial security applications and PGP

It makes use of triple DES for encryption. The ingredients are as follows.

- **Input:** Two pseudorandom inputs drive the generator. One is a 64-bit representation of the current date and time, which is updated on each number generation. The other is a 64-bit seed value; this is initialized to some arbitrary value and is updated during the generation process.
- **Keys:** The generator makes use of three triple DES encryption modules. All three make use of the same pair of 56-bit keys, which must be kept secret and are used only for pseudorandom number generation.
- **Output:** The output consists of a 64-bit pseudorandom number and a 64-bit seed value.

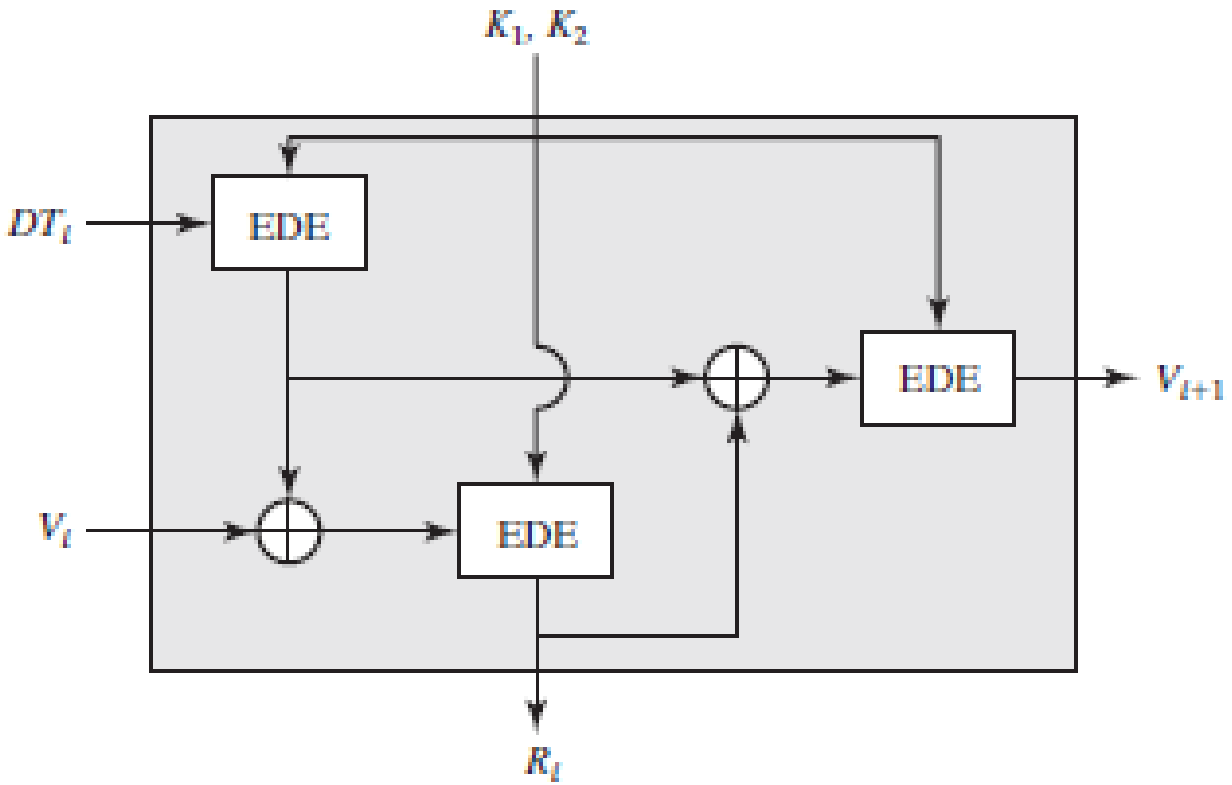


Figure 7.5 ANSI X9.17 Pseudorandom Number Generator

Let us define the following quantities.

DT_i	Date/time value at the beginning of ith generation stage
V_i	Seed value at the beginning of ith generation stage
R_i	Pseudorandom number produced by the ith generation stage
K_1, K_2	DES keys used for each stage

Then

$$R_i = \mathrm{EDE}([K_1, K_2], [V_i \oplus \mathrm{EDE}([K_1, K_2], DT_i)])$$
$$V_{i+1} = \mathrm{EDE}([K_1, K_2], [R_i \oplus \mathrm{EDE}([K_1, K_2], DT_i)])$$

RC4

- RC4 means Rivest Cipher 4
- RC4 is a stream cipher designed in 1987 by Ron Rivest for RSA Security.
- It is a variable key size stream cipher with byte-oriented operations. The algorithm is based on the use of a random permutation.
- Analysis shows that the period of the cipher is overwhelmingly likely to be greater than 10100 [ROBS95a].
- Eight to sixteen machine operations are required per output byte, and the cipher can be expected to run very quickly in software.

- RC4 is used in the Secure Sockets Layer/Transport Layer Security (SSL/TLS) standards that have been defined for communication between Web browsers and servers.
- It is also used in the Wired Equivalent Privacy (WEP) protocol and the newer WiFi Protected Access (WPA) protocol that are part of the IEEE 802.11 wireless LAN standard.
- RC4 was kept as a trade secret by RSA Security. In September 1994, the RC4 algorithm was anonymously posted on the Internet on the Cypherpunks anonymous remailers list. The RC4 algorithm is remarkably simple and quite easy to explain.
- A variable length key of from 1 to 256 bytes (8 to 2048 bits) is used to initialize a 256-byte state vector S, with elements S[0], S[1], …, S[255]. At all times, S contains a permutation of all 8-bit numbers from 0 through 255. For encryption and decryption, a byte k is generated from S by selecting one of the 255 entries in a systematic fashion. As each value of k is generated, the entries in S are once again permuted.

- **Advantages**
- RC4 stream ciphers are simple to use.
- The speed of operation in RC4 is fast as compared to other ciphers.
- RC4 stream ciphers are strong in coding and easy to implement.
- RC4 stream ciphers do not require more memory.
- RC4 stream ciphers are implemented on large streams of data.
- **Disadvantages**
- If RC4 is not used with strong MAC then encryption is vulnerable to a bit-flipping attack.
- RC4 stream ciphers do not provide authentication.
- RC4 algorithm requires additional analysis before including new systems.
- RC4 stream ciphers cannot be implemented on small streams of data.
- RC4 fails to discard the beginning of output keystream or fails to use non-random or related keys for the algorithm.

Assignment

- How to generate a random number between 0 and 1 in python?
- Which different functions are available in python module ‘random’?
- Compare between TRNG and PRNG.
- Explain RC4 algorithm in detail.