Assignment B Elective - III D2

Aim:

To understand random number generation and its use in cryptography

Problem Statement:

Write a program to generate a pseudorandom number generator for generating the long-term private key and the ephemeral keys used for each signing based on SHA-1 using Python/Java/C++. Disregard theuse of existing pseudorandom number generators available.

Input:

Seed for random generator.

Output:

Session keys for each message.

Theory:

A **pseudorandom number generator** (**PRNG**), also known as a **deterministic random bit generator** (**DRBG**), is an algorithm for generating a sequence of numbers whose properties approximate the properties of sequences of random numbers. The PRNG-generated sequence is not truly random, because it is completely determined by a relatively small set of initial values, called the PRNG's seed(which may include truly random values). Although sequences that are closer to truly random can be generated using hardware random number generators, pseudorandom number generators are important in practice for their speed in number generation and their reproducibility.

PRNGs are central in applications such as simulations (e.g. for the Monte Carlo method), electronic games (e.g. for procedural generation), and cryptography. Cryptographic applications require the output not to be predictable from earlier outputs, and moreelaborate algorithms, which do not inherit the linearity of simpler PRNGs, are needed.

Random generator, though called random, has a periodic behavior. It determines its initial state using a seed value, and for same seed value it always generates same set of numbers.

A simple random number generator algorithm used in this program can be describes as-

- Initialize first number p from seed values provided or using current time.
- Store length of p in len.
- For each generate call-
    1. Increment p
    2. Evaluate p^2.
    3. Evaluate length of p^2 as len2.
    4. Let a = (len2 – len)/2.
    5. Copy digits from p^2 to new number n2 at index ranging from a to end of p^2.
    6. Replace n2 as p, return p.

The use of random number generator here is to generate session keys. This is done to provide Forward Secrecy.

In cryptography, forward secrecy (FS; also known as perfect forward secrecy is a property of secure communication protocols in which compromise of long-term keys does not compromise past session keys. Forward secrecy protects past sessions against future compromises of secret keys or passwords. If forward secrecy is utilized, encrypted communications and sessions recorded in the past cannot be retrieved and decrypted should long-term secret keys or passwords be compromised in the future, even if the adversary actively interfered.

At the heart of Forward Secrecy is the use of the Diffie-Hellman key exchange. In addition, in order to gain the benefits of Forward Secrecy, ephemeral keys must be utilized during the exchange. OWASP describes ephemeral keys as:

> Ephemeral keys are temporary keys used for one instance of a protocol execution and then thrown away. An ephemeral key has the benefit of providing forward secrecy, meaning a compromise of the site or service's long term (static) signing key does not facilitate decrypting past messages because the key was temporary and discarded (once the session terminated).

Therefore, to ensure that ephemeral keys are used in a Diffie-Hellman key exchange, either the Ephemeral Diffie-Hellman (DHE) or Ephemeral Elliptic Curve Diffie-Hellman (ECDHE) modes should be used. According to OpenSSL, ephemeral Diffie-Hellman keys are designed as:

> Ephemeral Diffie-Hellman uses temporary, public keys. Each instance or run of the protocol uses a different public key. The authenticity of the server's temporary key can be verified by checking the signature on the key. Because the public keys are temporary, a compromise of the server's long term signing key does not jeopardize the privacy of past sessions.

In this implementation we have demonstrated use of pseudorandom generator to generate keys for each message transmission.

The temporary keys are not shared but evaluated at sender and receiver side independently.

The seed value acts as the shared long-term private key, which is known to both sides beforehand.

Mathematical Modeling:

Let S be the system that represents the Key generation using random numbe generator.

Initially,
S ={_}
Let,
S ={I, O, F}
Where,
I - Represents Input set
O - Represents Output set
F - Represents Function set

Input set - I:
I = {S, M}
Where,

- S - Represents the seed value.
- M – Represents message to encrypt.

Output set - O:
O = {Me}
Where,

- Me is message encrypted.

Function Set - F:
F = {F1, F2, F3**}**
Where,
• F1 - Represents a function for generating random number.
F1(S) -> {num}

• F2 - Represents a function for generating session keys.
F2(num) -> {Ks}
– Ks – Key generated using num.

F3(M,Ks) -> {Me}

– Me – Encrypted version of M.

F4(Me,Ks) -> {Md}

– Md – Decrypted version of Me, i.e. Md=M.

Conclusion:

Thus, we have studies and implemented password encryptor.