Applied Cryptography

Unit-2 Symmetric key Cryptography

Lecture Notes 5:

Substitution Permutation Network (SPN)

Recommended Reading:

Katz-Lindell: Chapter 6:6.2.1

1. Introduction:

Private Key cryptography usability can be achieved by the right usage of the symmetric key algorithms. And most of the symmetric key based algorithms rely on the ciphers to be treated either in the form of stream or block ciphers. Throughout our chapter the algorithms that we discuss like DES and AES are block cipher-based algorithms. It becomes very essential to understand the underlying basic design principle of the block cipher. The mechanism or design paradigm do include certain specific type of network that helps in generating truly pseudorandom functions. In this lecture, we will understand how SPN is utilized to produce a truly pseudorandom function.

2. Why Substitution Permutation Network?

To understand this truly, we must understand that the key generated in blocks should be truly pseudorandom and it demands building a set of permutations with concise description. Is it really feasible without such type of networks means, its absolutely not possible because if the input string length is say 'k' bits this can produce 2^k ! permutations. As the value of k increases, the meaning of concise description seems as not feasible. That's the reason we are going for SPN type of network that can generate set of permutations with concise description. Next we need to assure that this random set been generated using such type of networks is effective against attackers. So, we wanted to make sure that if a change in single bit of the key should at least make half changes in the cipher/plain text. This implies that a one-bit change in the input should "affect" every bit of the output. (Note that this does not mean that all the output bits will be changed—that would be different behaviour than one would expect for

a random permutation. Rather, we just mean informally that each bit of the output is changed with probability roughly half.

3. **Structure of SPN networks:**

- In his 1949 paper Shannon also introduced the idea of substitution-permutation (S-P) networks, which now form the basis of modern block ciphers.
- An SP network is the modern form of a substitution-transposition product cipher.
- S-P networks are based on the two primitive cryptographic operations.
- Generally, the substitution and permutation network try to construct a Lager block permutation F from many smaller random looking permutations $\{f_i\}$.
- Eg: F to have a block length of 128 bits.
- We define F as below:
  - The key k for F will specify 16 permutations $f_1, f_2 \ldots . f_{16}$.
  - Given an input $x \in \{0, 1\}$ 128, we parse it as 16 bytes $x_1 \cdot \cdot \cdot x_{16}$ and then set
    $Fk(x) = f_1(x_1)\ k \cdot \cdot \cdot k f_{16}(x_{16})$.

3.1 Confusion and Diffusion Paradigm:

- Attacker should not be able to guess the key and this confusion achievement is obtained by using the round function. A single bit change in key affects most of the bits in cipher.
- The output obtained with F may not be pseudorandom, so introduction of confusion and diffusion comes into play.
- **To achieve Pseudorandomness in the F a** diffusion step is introduced whereby the bits of the output are permuted, or "mixed," using a mixing permutation. This has the effect of spreading a local change (e.g., a change in the first byte) throughout the entire block. The confusion/diffusion steps—together called a round—are repeated multiple times. This helps ensure that changing a single bit of the input will affect all the bits of the output.

3.2Typical Steps of a Round Operation:

1. a two-round block cipher following this approach would operate as follows. First, confusion is introduced by computing the intermediate result $f_1(x_1)k \cdots kf_{16}(x_{16})$
2. The bits of the result are then "shuffled," or re-ordered, to give x_0. Then $f_1'(x_1')k \cdots kf_{16}(x_{16})$ is computed using possibly different functions f_i, and the bits of the result are permuted to give output x''. The {fi}, {f_i'}, and the mixing permutation(s) could be random and dependent on the key.

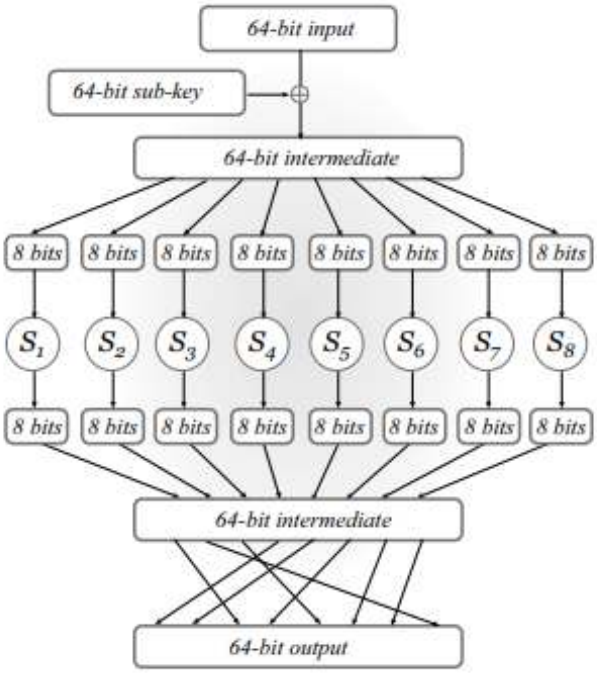
3.3Basic Structure of SPN:

- The round functions have a particular form rather than being chosen from the set of all possible permutations on some domain.
- Specifically, rather than having (a portion of) the key k specify an arbitrary permutation f, we instead fix a public "substitution function" (i.e., permutation) S called an S-box, and then let k define the function f given by f(x) = S(k $\oplus$ x).

3.4Example Network for Understanding SP network:

64 bit block length 8 bit S-boxes as shown in figure.

1. **Key Mixing:**
   Set x := x $\oplus$ k, where k is the current-round sub-key
2. Substitution:
   x := S1(x1)k · · · kS8(x8), where xi is the ith byte of x
3. Permutation:
   Permute the bits of x to obtain the output of the round. The output of each round is fed as input to the next round. After the last round there is a final key-mixing step, and the result is the output of the cipher



The output of each round is fed as input to the next round. After the last round there is a final key-mixing step, and the result is the output of the cipher. (By Kerckhoffs' principle, we assume the S-boxes and the mixing permutation(s) are public and known to any attacker. This means that without a final key-mixing step, the last substitution and permutation steps would offer no additional security since they do not depend on the key.)

3.5Keys:

- The key to the block cipher is sometimes referred to as the master key.
- The sub-keys that are XORed with the intermediate results in each round are derived from the master key according to a key schedule.
- Key schedule is often very simple and may work by just taking subsets of the bits of the key, though more complex schedules can also be defined.

Design Principles in SPN:

1. **Invertertibility:**
   With the key the SPN is invertible, entire SPN can be inverted by working from the final round. Inverting a single round means just reordering of bits and then appropriate XORed with sub-key to obtain the original input.

2. **Avalanche Effect:**

What is Avalanche?

- "Small Change in Input makes major changes in Output"
- Doesn't mean changing one bit of input affects every bit of output but it do have some effect on every bit"

Proof of Avalanche in SPN:

1.The S-boxes are designed so that **changing a single bit of the input to an S-box changes at least two bits in the output of the S-box**.

2.The mixing permutations are designed so that the output bits of any given S-box are spread into different S-boxes in the next round.

- Consider now what happens when the block cipher is applied to two inputs that differ by only a single bit:

1. **After the first round, the intermediate values differ in exactly two bit-positions.**

This is because XORing the current sub-key maintains the 1-bit difference in the intermediate values, and so the inputs to all the S-boxes except one are identical.

In the one S-box where the inputs differ, the output of the S-box causes a 2-bit difference. The mixing permutation applied to the results changes the positions of these differences, but maintains a 2-bit difference.

2. By the second property mentioned earlier, the mixing permutation applied at the end of the first round spreads the two bit-positions where the intermediate results differ into two different S-boxes in the second round.

So, in the second round there are now two S-boxes that receive inputs differing by a single bit. Following the same argument as before, we see that at the end of the second round the intermediate values differ in 4 bits.

- One might expect that the "best" way to design S-boxes would be to choose them at random.
- Interestingly, this turns out not to be the case, at least if we want to satisfy the above criterion.
- For example, consider the case of an S-box operating on 4-bit inputs and let x and x' be two different inputs.
- Let $y := S(x)$, and now consider choosing $y' \neq y$ at random as the value of $S(x')$. 33

- There are 4 strings that differ from y in only 1 bit, and so with probability 4/15 we will choose y' that does not differ from y in two or more bits.
- The problem is compounded when we consider all inputs, and becomes even worse when we consider that multiple S-boxes are needed.
- We conclude based on this example that, as a general rule, it is best to carefully design S-boxed with certain desired properties (in addition to the one discussed above) rather than choosing them blindly at random.

Security at SPN:

- Attack on a single-round substitution-permutation network: Let F be a single-round substitution-permutation network.
- We demonstrate an attack where the adversary is given only a single input/output pair (x,y) for a randomly-chosen input value x, and easily learns the secret key k for which y = Fk(x).
- The adversary begins with the output value y and then inverts the mixing permutation and the S-boxes. It can do this because the specification of the permutation and the S-boxes is public
- The intermediate value that the adversary computes from these inversions is exactly $x \oplus k$ (assuming, without loss of generality, that the master key is used as the sub-key in the only round of the network).
- Since the adversary also has the input x, it immediately derives the secret key k. This is therefore a complete break.