CSL7480 Cryptography **27 Jan 2023**

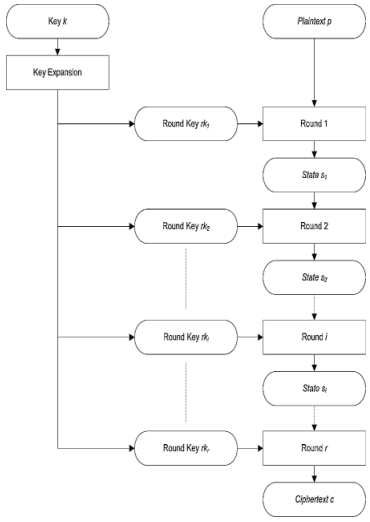
Lecture 10: Cryptography

Lecturer: Somitra Sanadhya *Scribe: Pranav Goswami (B20CS016)*

10.1 Iterated Block Cipher

An iterated block cipher encrypts a plaintext block by a process that has several rounds. In each round, the same transformation or round function is applied to the data using a subkey. The set of subkeys are usually derived from the user-provided secret key by a key schedule.

The number of rounds in an iterated cipher depends on the desired security level and the consequent trade-off with performance. In most cases, an increase in number of rounds improves security offered by a block cipher, but for some, some ciphers the number of rounds required to achieve adequate security will be too large, i.e. practically not feasible to implement on hardware.



10.1.1 Advantages

- Implementation is cheaper with respect to both hardware and software.
- Analysis is easier.

Here functions used in each round are deterministic, making iterative block cipher a deterministic algorithm. Along with that, each round function is invertible.

10.2 Data encryption standard (DES)

DES uses a block cipher and encrypts data in blocks of size of 64 bits each, which means 64 bits of plain text go as the input to DES, which produces 64 bits of ciphertext. The same algorithm and key are used for encryption and decryption, with minor differences.

- Block Size = 64 bits.
- Secret Key = 56 bits.
- Round Key = 48 bits.

The initial key consists of 64 bits, however before DES process starts, every 8th bit of the key is discarded to produce a 56-bit key. That is bit positions 8, 16, 24, 32, 40, 48, 56, and 64 are discarded. DES consists of 16 steps, each of which is called a round. The output of last round is not swapped and forms cipher text.

10.2.1 DES Encryption

Uses round keys - k1, k2, k3, k4, ... , k16.

Algorithm 1 DESEncrypt(k, msg)

```
for i = 1 to n do
  RoundFunction(K[i], State[i]);
end forswap( State[i] );
cipher = State[i];
Output: cipher
```

10.2.2 DES Decryption

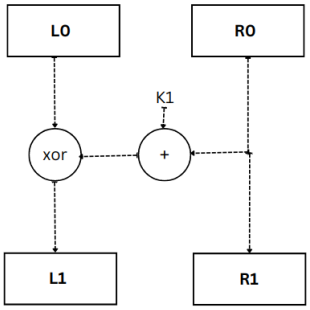
Uses the same round keys just in reverse order - k16, k15, k14,....,k1.

Algorithm 2 DESDencrypt(k, cipher)

```
for i = n to 1 do
  RoundFunction(K[i], State[i]);
end forswap( State[ i ] );
msg = State[ i ];
Output: msg
```

10.2.3 Proof

Let us formally proof the above-given argument that DES Decryption can be done in the same algorithm just by using keys in reverse order.
Given a message (m) split into two halves, L0 (MSB 32 bits) and R0 (LSB 32 bits).

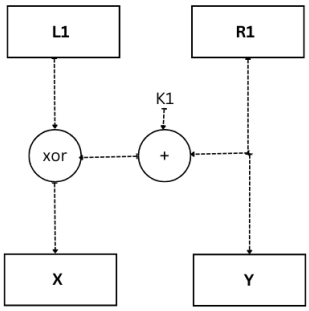


From the above figure for only a single round, we get,

$$L1 = L0 \oplus F(k1, R0)$$

$$R1 = R0$$

Now let us reverse this, that is



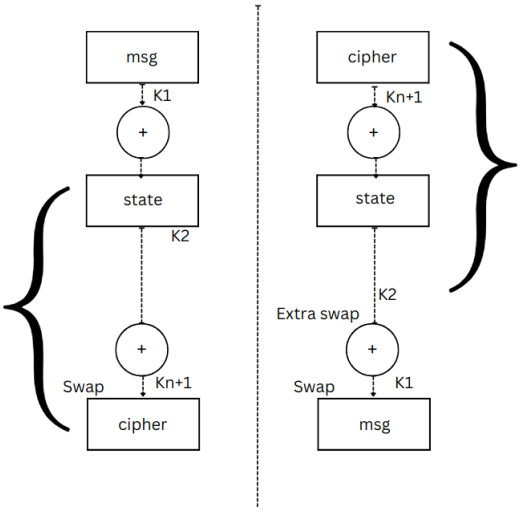
We can formally prove that

$$Y = R1 = R0$$

$$X = L0 \oplus F(k1, R0)$$

$$X \oplus F(k1, R0) = L0$$

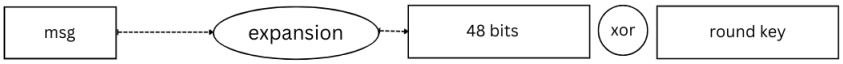
From the above-given equation, we get X = L1 and it is evident that Y = R1 = R0.
Hypothesis: We know that this algorithm works for N rounds.
To Prove: Algorithm works for (N + 1) rounds.
Induction step:



We know from our hypothesis that the algorithm works for N rounds (K2 - Kn+1) which is equivalent to (Kn+1 - K2) shown in the right part of the figure with an extra swap. With this we only remain with 1 round which is already proved earlier.

10.2.4 Determine round keys and function

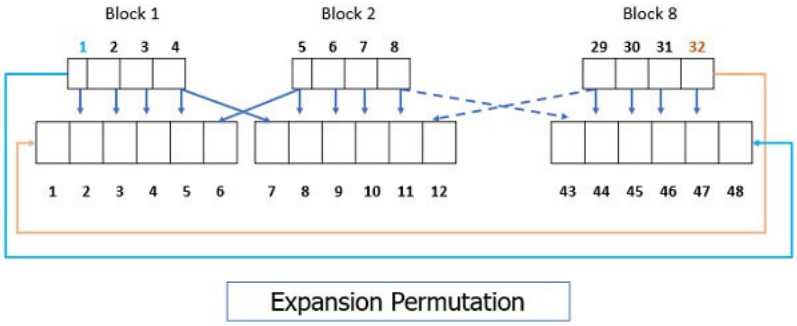
Function F takes a 32 bit plain text and a 48 bit round key as input to produce an output of 32 bit. We aim to achieve keys to be uniformly random and knowing that XOR operation can kill bias of highly biased variable we prefer doing XOR operation on given input of function F.
The trouble lies in the size of both parameters, on being 32-bit and the other 48-bit. So we perform expansion on plain text to make it 48-bit and then perform XOR operation.



And then we use substitution boxes (lookup tables) (Si) to again return back to 32-bit from 48-bit output of expansion.

10.2.4.1 Expansion Algorithm

In the expansion algorithm, one makes 8 batches of 4 bits in a circular fashion and then appends two extra bits, one in front and other at rear of batch. Front bit = Rear bit of previous batch and similarily Rear bit = Front bit of next batch. Pictorially represented below.



10.3 References

- http://x5.net/faqs/crypto/q55.html
- https://www.geeksforgeeks.org/data-encryption-standard-des-set-1/
- https://binaryterms.com/data-encryption-standard-des.html