IDEA

IDEA(INTERNATIONAL DATA ENCRYPTION ALGORITHM) este un algoritm simetric de criptare care foloseste o cheie pentru criptare si aceeasi cheie secreta si pentru decriptare. IDEA a fost scris de Xuejia Lai si James L. Massey. IDEA lucreaza cu blocuri de 64 de biti, iar cheia are 128 de biti. Procesul de criptare necesita efectuarea a 8 runde. Decriptarea are loc in aceeasi maniera imediat dupa ce s-au calculat subcheile de decriptare din subcheile de criptare.

IDEA foloseste 52 de subchei fiecare de 16 biti.

Sunt obtinute dintr-o cheie de 128 de bitsi care e shiftata ciclic prin rotatie la stanga si impartire ac cheii in 6 parti a cate 16 bitsi.

In verilog shiftarea ciclica cu 25 de bitsi a unei chei de 127 de bitsi poate fi obtinuta prin:
KEY[127:0] = KEY << 25 | KEY >> 103;

Acest proces se repeta pana cand sa obtin 52 de chei.

Cheile se folosesc in 8 runde complete si o runda finala (half-round).

Input-ul primit este impartit in felul urmator:
{X1, X2, X3, X4} = DATA_IN;

Primele 8 runde respecta pasii:

- se înmulţeşte X_0 cu prima subcheie;
- se adună X_1 la a doua subcheie;
- se adună X_2 la a treia subcheie;
- se înmulţeşte X_3 cu a patra subcheie;
- XOR între rezultatele paşilor l şi 3;
- XOR între rezultatele paşilor 2 şi 4;
- se înmulţeşte rezultatul pasului 5 cu subcheia numărul 5;
- se adună rezultatele obţinute în cadrul paşilor 6 şi 7;
- se înmulţeşte rezultatul de la pasul 8 cu subcheia numărul 6;
- se adună rezultatele obţinute la paşii 7 şi 9;

S0 = XOR între rezultatele paşilor l şi 9;
S1 = XOR între rezultatele paşilor 3 şi 9;
S2 = XOR între rezultatele paşilor 2 şi 10;
S3 = XOR între rezultatele paşilor 4 şi 10.

Intre doua runde (mai putin intre runda 8 si half-round) se interschimba S2 cu S3. Astfel pentru runda urmatoare inputul va fi:
{S0, S2, S1, S3}.

Ultima runda sau (half-round):

se înmulţeşte X_0 cu prima subcheie;
- se adună X_1 la a doua subcheie;
- se adună X_2 la a treia subcheie;
- se înmulţeşte X_3 cu a patra subcheie.

Algoritmul IDEA foloseste operatiile:
-XOR pe 16 bitsi
-suma pe 16 bitsi mod 2 ^ 16
-inmultire pe 16 bitsi mod 2 ^ 16 + 1

Inmultirea este tratata mai special deoarece se cere ca toate elementele din grupul multiplicativ sa fie inversabile. Aceasta necesita urmatorul artificiu:

0,1,…,2 ^ 16 – 1 sunt mapate la 1, 2, … 2 ^ 16
Unde 0 va fi reprezentarea lui 2 ^ 16 pe 15 bitsi.

Inmultirea pe 16 bitsi modulo 2 ^ 16 + 1 se va face in felul urmator:

```
input [15:0]A;
input [15:0]B;
output reg [15:0]C;

reg [31:0]result;
reg [15:0] lo;
reg [15:0] hi;

always @(*) begin
   result = A * B;
   if (result == 32'b0) begin
      C = (-1) * A + (-1) * B + 1;
   end else begin
      hi = result >> 16;
      lo = result;
      if (lo > hi) begin
         C =  lo - hi;
      end else begin
         C = lo - hi + 1;
      end
   end
end
```

Decriptarea : Este asemanatoare procesului de criptare dar cu modificari asupra cheilor.
Avand in vedere ca acesta este un algoritm de criptare simetrica se va pleca de la aceeasi cheie cu care s-a realizat criptarea.
Pasii de obtinere a celor 52 de chei necesare pentru decriptare:

1. Asemanator ca la criptare cheia este in mod succesiv impartita in 6 parti a cate 16 bitsi. Dupa fiecare astfel de impartire, cheia este shiftata ciclic la stanga cu 25 de btisi.
2. Primele 4 chei din cele 52 se inverseaza astfel:
   KD(1) = 1/K(49)
   KD(2) = -K(50)
   KD(3) = -K(51)
   KD(4) = 1/K(52)
3. Urmatoarele chei se vor inversa dupa regula:
   KD(5) = K(47)
   KD(6) = K(48)

   KD(7) = 1/K(43)
   KD(8) = -K(45)
   KD(9) = -K(44)
   KD(10) = 1/K(46)

   Regula de mai sus se aplica din 6 in 6. Indicele fiecarui KD va fi incrementat cu 6 iar indicele K va fi decrementat cu 6.

   Ex:

```
for (i = 5; i <= 52; i = i + 6) begin
    InversedKeys[i] = Keys[52 - i];
    InversedKeys[i + 1] = Keys[52 - i + 1];

    InversedKeys[i + 2] = inverseMulOut[52 - i - 4];
    InversedKeys[i + 3] = -Keys[52 - i - 2];
    InversedKeys[i + 4] = -Keys[52 - i - 3];
    InversedKeys[i + 5] = inverseMulOut[52 - i - 1];

  end
```

Inversarea inmultirii in cadrul algoritmului idea poate fi implementata astfel:
(avand in vedere ca 2 ^ 16 + 1 este coprim cu orice numar intre [0 si 2 ^ 16])

```
int modInverse(int a, int m)
{
    int m0 = m, t, q;
    int x0 = 0, x1 = 1;

    if (m == 1)
      return 0;

    while (a > 1)
    {
        // q is quotient
        q = a / m;

        t = m;

        // m is remainder now, process same as
        // Euclid's algo
        m = a % m, a = t;

        t = x0;
```

```
            x0 = x1 - q * x0;

            x1 = t;
        }

        // Make x1 positive
        if (x1 < 0)
            x1 += m0;

        return x1;
    }
```

Schema algoritmului idea:
Unde:

- Cu cercuri albastre este notat XOR
- Cu cercuri rosii inmultirea modulo 2 ^ 16 + 1
- Cu patrate verzi suma modulo 2 ^ 16