Bilgisayar ve Bilişim Bilimleri Fakültesi

Bilgisayar Mühendisliği Bölümü

Kriptolojiye Giriş Projesi

Ad – Soyadı: Waasiq MASOOD, Muhammed SAQER

Öğrenci Numarası: B181210557**,** B191210554

Konu: RC5 algoritması

RC5 Algorithm Explanation:

RC5 is a block cipher notable for its simplicity. Following is explanation for the algorithm:

- Symmetric block cipher - the same secret cryptographic key is used for encryption and decryption
- Designed by Ronald Rivest in 1994.
- RC stands for "Rivest Cipher", or "Ron's Code.
- It is quite fast as it uses only primitive computer operations.
- It allows a variable number of rounds and variable bit size key to add flexibility.
- The algorithm has been designed to work in size of 32, 64 or 128 bits.

Features of RC5:

1- Adaptable to processors of different word lengths:

For example, with 64-bit processor RC5 can exploit their longer work length Therefore the number w of bits in a word is a parameter of RCS, different choices of this parameter results different algorithms.

2- Variable number of rounds

The user can explicitly manipulate the trade-off between higher speed and higher security. So the number of rounds I is a second parameter of RC5.

3. Variable length cryptographic key

The user can choose the level of security appropriate for his application the key length b in in bytes | is thus a third parameter of RC5.

4- Simple

It is simple to implement, to analyze and evaluate, so that the cryptographic strength can be more rapidly determined.

5- Low memory requirements

It is easily implemented on devices with restricted memory.

6- Data-dependent rotations

RC5 highlight the use of data-dependent rotations and encourage the assessment of the cryptographic strength data-dependent can provide

Parameterization

RC5 is a parameterized algorithm word-oriented two-word input and two-word output)

Parameters in RC5 Algorithm:

- Word size: w (16,32,64)
- Number of rounds: r (0 ,1 ,2 …,255)
- Number of bytes in key K: b (0 ,1 ,2 …,255)
- RC5 algorithm notation: RC5 –w/r/b
- RC5 algorithm Example: RC5 -32 / 16/7
  - Similar to DES
  - Two 32-bit word inputs and outputs
  - 16 rounds
  - 7-byte (56-bit) key

Parameter	Definition	Allowable Values
w	Word size in bits, RC5 encrypts 2-word blocks	16, 32, 64
r	Number of rounds	0, 1, …, 255
b	Number of 8-bit bytes (octets) in the secret key K	0, 1, …, 255

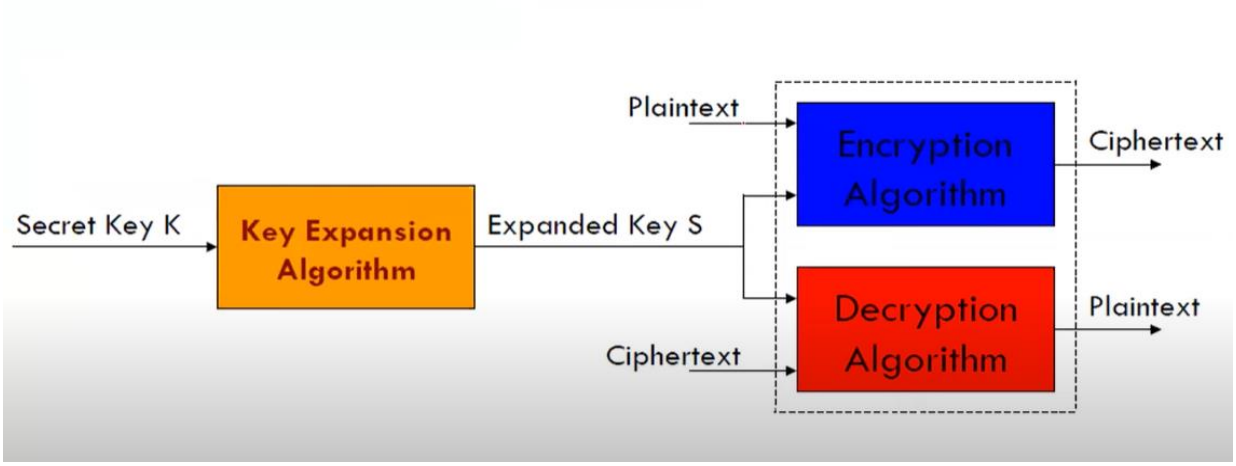
RC5 Algorithm Working:

Three components of RC5

- Key Expansion
- Encryption Algorithm
- Decryption Algorithm

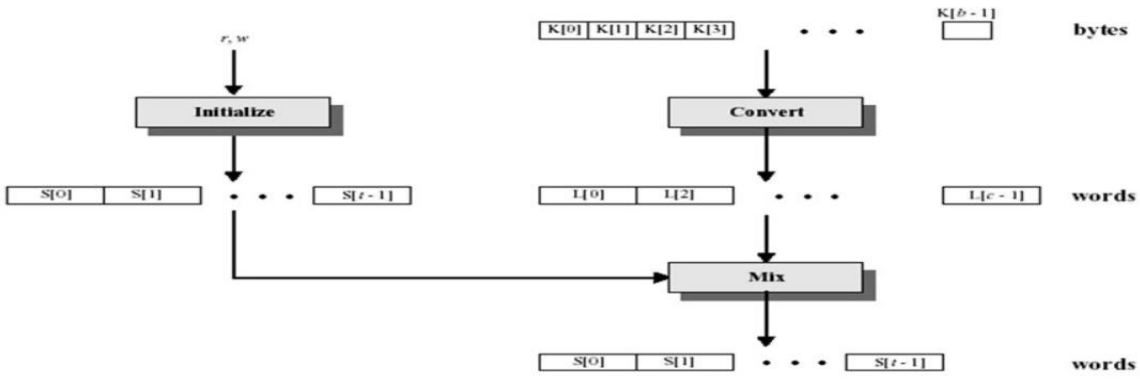
Advantages of Block Cipher:

- Fast symmetric block cipher
- Same key for encryption and decryption
- Plaintext and ciphertext are fixed-length bit sequences (blocks)



Key Expansion Algorithm:

- RC5 performs a complex set of operations on the secret key to produce a total of t subkeys.
- Two subkeys are used in each round, and two subkeys are used on an additional operation that is not part of any round, so t = 2r + 2.
- Each subkey is one word (w bits) in length.
- In RCS, the plain text message is divided into two blocks A and B each of 32 bits.
- Then two subkeys are generated S[0] and S[1].
- These two subkeys are added into A and B respectively.
- The subkeys are stored in a t-word array labeled S[O], S[I],..., S[t- 1].
- Using the parameters r and w as inputs, this array is initialized to a particular fixed pseudorandom bit pattern.
- Then the b-byte key, K[0...b-1] is converted into a c-word array L[o...c-1].
- Finally, a mixing operation is performed that applies the contents of L to the Initialized value of S to produce a final value for the array S.



After Key Schedule:

- Then the process of the round begins.
- In each round, the following operation is performed.
  - Bitwise XOR.
  - Left Circular Shift.
  - Addition to the next subkey, for both C and D.
    - This, is the addition operation, and then the result of the addition mod 2^w is performed.

Key Expansion Algorithm: Operations in detail

- Step-1: Convert secret key bytes to words

  b byte key K, (K [0], K[I], ..., K[b-1]) is converted to word array L[0], L[1], ..., L[c-1]

- Step-2: Initialize subkey array S (S [0], S[I], ... S[t-I])

  S[0] = P_w

  for i=I to t-1 do

  s[i] = s[i-1] + Q_w*;*

- Step-3: Mix the secret key into subkey array S

  i=j=X=Y=0;

  *Do 3*max (t, c) times:*

  X=S[i] = (S[i] + X + Y) <<< 3;

  Y=L[j] = (L[j] + X + Y) <<< (X + Y);

  i = (i+1) mod t;

  j = (j+1) mod c;

Encryption Algorithm

- RC5 uses 3 primitive operations
  - Addition, Subtraction (of words): modulo 2^w
  - Bitwise XOR
  - Left, right circular rotation
- Not a classical Feistel Structure.
- The Plaintext is assumed to reside in the two w-bit re as A and B.

- LE_i, and RE_i , refer to the left and right half of the data a round i has completed.

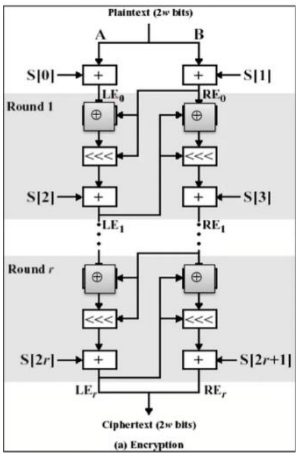
```
LE0 = A + S[0];
RE0 = B + S[1];
for i = 1 to r do
    LEi = ((LEi-1 ⊕ REi-1) <<< REi-1) + S[2i];
    REi = ((REi-1 ⊕ LEi) <<< LEi) + S[2i+1];
```

- The ciphertext is contained in the two variables LE_r and RE_r
- Each of the r rounds consists of
  - a substitution using both words of data
  - a permutation using both words of data
  - a substitution that depends on the key

* The simplicity of the operation which can be defined in 5 lines of code.

* Both halves of the data are updated in each round.

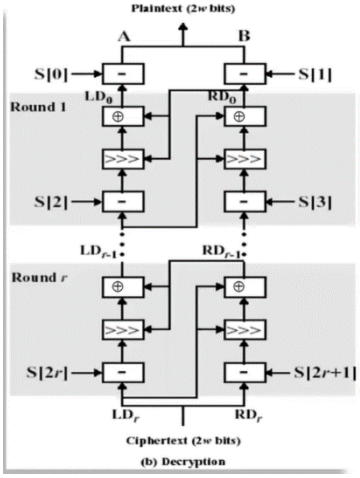
* Thus one round of RC5 is equivalent to 2 rounds of DES.



Decryption Algorithm

- Two w-bits of ciphertext are initially assigned to the two one-word variables LD_r, and RD_r,
- LD_r, and RD_r , refer to the left and right half of the data before round i has begun.

```
for i = r downto 1 do
    RDi-1 = ((RDi – S[2i+1] >>> LDi) ⊕ LDi) ;
    LDi-1 = ((LDi – S[2i] >>> Rdi-1) ⊕ RDi-1) ;
B = RD0 - S[1];
A = LD0 - S[0];
```



RC5 Modes

- RFC2040 defines 4 modes used by RC5:
    - RC5 Block Cipher, is ECB mode
    - RC5-CBC, input length is a multiples of 2w
    - RC5-CBC-PAD, any length CBC with padding
        - Output can be longer than input
    - RC5-CTS, CBC with padding
        - Output has same length than input

Uygulamanın Örnek Çalıştırma Ekran:

The application has been coded in Python3, the exe has been generated with pyInstaller. The official paper has been used as reference for the algorithm. 32 bit key length and block length has been used, of course the entered string is broken into blocks and then encrypted. String is taken for encryption and a base64 is outputted. The user is also asked for the number of rounds he has used to encrypt the plain text. For the decryption a user has to provide the base64 encrypted text and the number of rounds, the decrypted text is then printed on the screen. Similar thing is done for the web interface of the application.

Screenshots:

Example encryption:

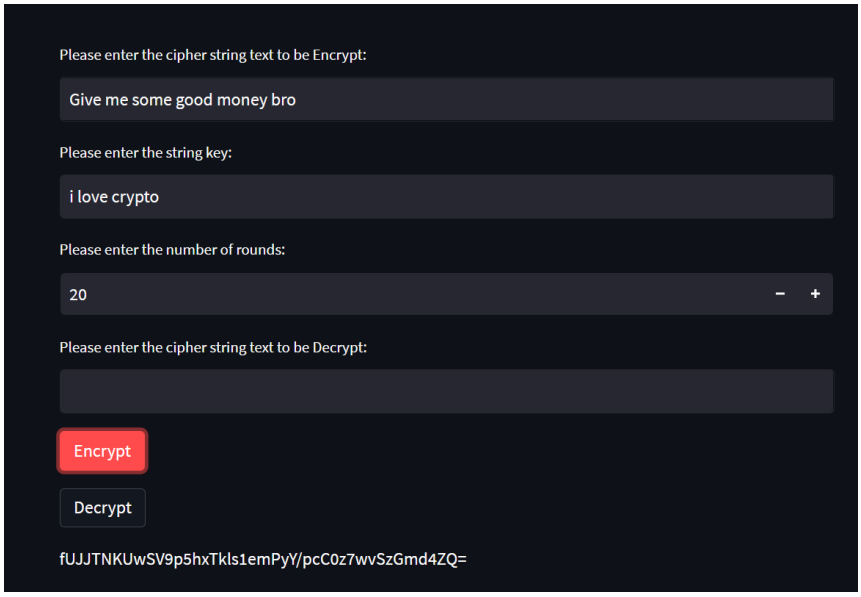
```
Please enter 1 for encryption and 2 for decryption and 3 to quit: 1
Please enter the text to be encrypted: waasiq loves cyberpunk
Please enter the key: kafi
Please enter the number of rounds: 15
Encrypted text:  GefSQfLC9gtTDKtTCV5zPZEDOmH/s6z4
```

Decryption of the encrypted text:

```
Please enter 1 for encryption and 2 for decryption and 3 to quit: 2
Please enter the cipher text to be decrypted: GefSQfLC9gtTDKtTCV5zPZEDOmH/s6z4
Please enter the key: kafi
Please enter the number of rounds: 15
Decrypted text:  waasiq loves cyberpunk
```

The program doesn’t work if the number of rounds are same or the key. Web interface can be run using the command ``` python -m streamlit run app.py``` in web folder.

Encryption done on web interface:



Decryption done on the web interface: