RC5

Outline

- Introduction (Feistel Networks)
- What is RC5
- Parameterization
- Algorithm
- The security of RC5
- Conclusion

Feistel Network

- **block cipher** is a symmetric key cipher operating on fixed-length groups of bits, called blocks.
- One of the most structures used in construction block ciphers is **Feistel Network** Structure

Feistel Network

- Feistel networks were first seen commercially in IBM's Lucifer cipher, designed by Horst Feistel and Don Coppersmith.
- Feistel networks gained respectability when the U.S. Federal Government adopted the DES (a cipher based on Lucifer, with some changes NSA).
- **RC5** is like a **Feistel Network** structure.

Recap

- Introduction (Feistel Networks)
- What is RC5
- Parameterization
- Algorithm
- The security of RC5
- Conclustion

What is RC5

What is RC5

- **RC5** is a block cipher notable for its simplicity. Designed by Ronald Rivest in 1994.
- *RC* stands for "Rivest Cipher", or alternatively, "Ron's Code.
- Rivest announced also RC2 and RC4 and now there is RC6 which is The Advanced Encryption Standard (AES) candidate (RC6 was based on RC5).

Features

- Symmetric block cipher (Like Feistel Network Structure)
  - the same secret cryptographic key is used for encryption and decryption
- Suitable for hardware and software
  - It uses only computational primitive operations commonly found on typical microprocessors
- Fast
  - Cause it uses Word-Oriented operations

Features count.

- 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 RC5, different choices of this parameter results different algorithms.
- 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

Features count.

- Variable length cryptographic key
  - The user can choose the level of security appropriate for his application the key length *b* in bytes is thus a third parameter of RC5
- Simple
  - It is simple to implement, This simplicity makes it more interesting to analyze and evaluate, so that the cryptographic strength can be more rapidly determined
- Low memory requirements
  - So it is easily implemented on devices with restricted memory

Features count.

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

Features - Highlight

- Data-dependent rotations
- Variable block size
- Variable number of rounds
- Variable key size

Recap

- Introduction (Feistel Networks)
- What is RC5
- Parameterization
- Algorithm
- The security of RC5
- Conclusion

Parameterization

Parameterization

RC5 is a parameterized algorithm, and a particular RC5 algorithm is designated as RC5-$w/r/b$. We summarize these parameters below:

- w The *word size*, in bits. The standard value is 32 bits; allowable values are 16, 32, and 64. RC5 encrypts two-word blocks so that the plaintext and ciphertext blocks are each $2w$ bits long.
- r The number of rounds. Allowable values are 0, 1, ..., 255.
- b The number of bytes in the secret key K. Allowable values of b are 0, 1, ..., 255.

Parameterization count.

- RC5 algorithm example: RC5-32/16/7
  - similar to DES
  - Two 32-bit word inputs and outputs
  - 16 rounds
  - 7-byte(56-bit) secret key
- Choices for *w* and *r*
  - speed vs. security
- Choosing larger number of rounds provides an increased level of security

Notations and Primitive operations

We use $\lg(x)$ to denote the base-two logarithm of x.

RC5 uses only the following three primitive operations (and their inverses).

1. Two's complement addition of words, denoted by "+". This is modulo-2^w addition. The inverse operation, subtraction, is denoted "$-$".
2. Bit-wise exclusive-OR of words, denoted by $\oplus$.
3. A left-rotation (or "left-spin") of words: the cyclic rotation of word x left by y bits is denoted $x \lll y$. Here y is interpreted modulo w, so that when w is a power of two, only the $\lg(w)$ low-order bits of y are used to determine the rotation amount. The inverse operation, right-rotation, is denoted $x \ggg y$.

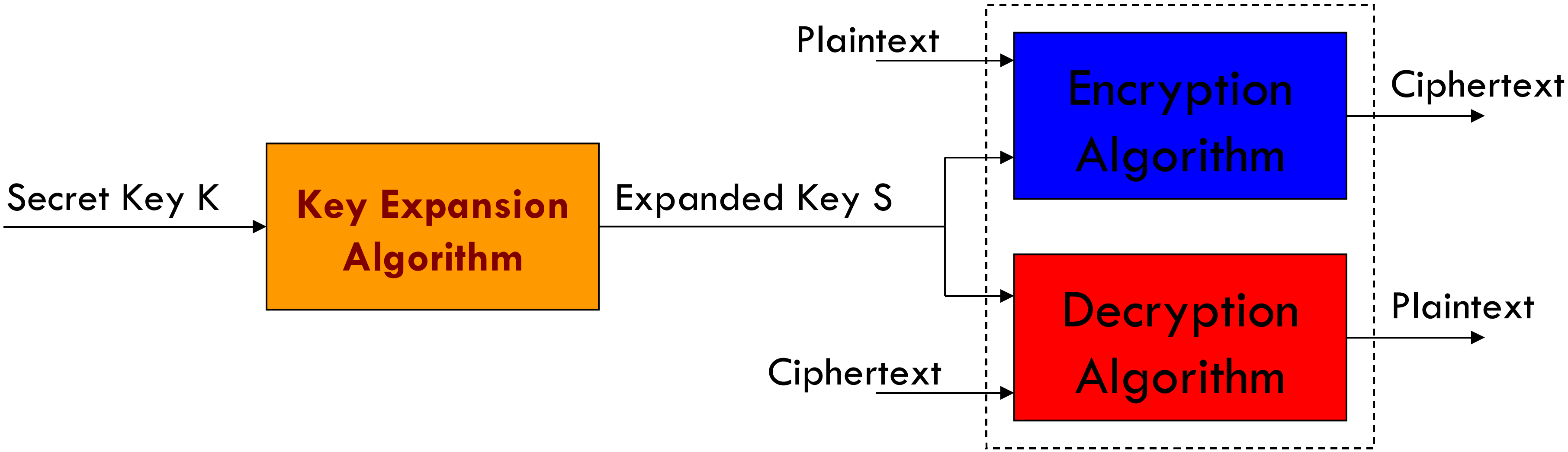
Recap

- Introduction (Feistel Networks)
- What is RC5
- Parameterization
- Algorithm
- The security of RC5
- Conclusion

Algorithm

Algorithm

- The are three components of RC5
  - Key expansion algorithm
  - Encryption algorithm
  - Decryption algorithm



Encryption

The description of the encryption algorithm is given in the pseudo-code below. We assume that the input block is given in two w-bit registers A and B, and that the output is also placed in the registers A and B.

$$
\begin{aligned}
&A = A + S[0] \\
&B = B + S[1] \\
&\textbf{for } i = 1 \textbf{ to } r \textbf{ do} \\
&\qquad A = ((A \oplus B) \lll B) + S[2i] \\
&\qquad B = ((B \oplus A) \lll A) + S[2i+1]
\end{aligned}
$$

The decryption routine is easily derived from the encryption routine.

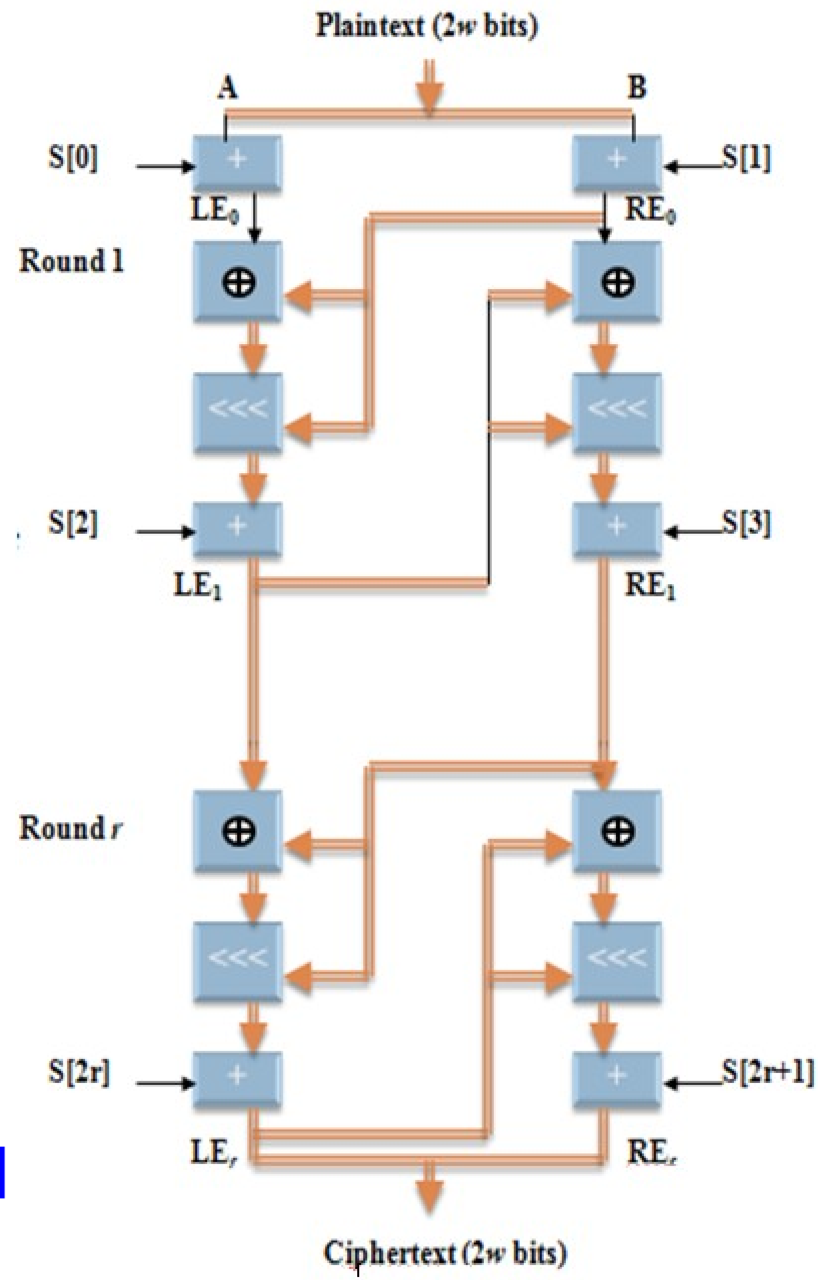
Encryption

```
A = A + S[0];
B = B + S[1];

for i = 1 to r do
    A = ((A ⊕ B) <<< B) + S[2*i];
    B = ((B ⊕ A) <<< A) + S[2*i + 1];
```

A <<< B

Bits in A are rotated to left by the amount specified by lower log2(*w*) *bits in B*

A				B			
03	02	01	00	07	06	05	04



Decryption

The decryption routine is easily derived from the encryption routine.

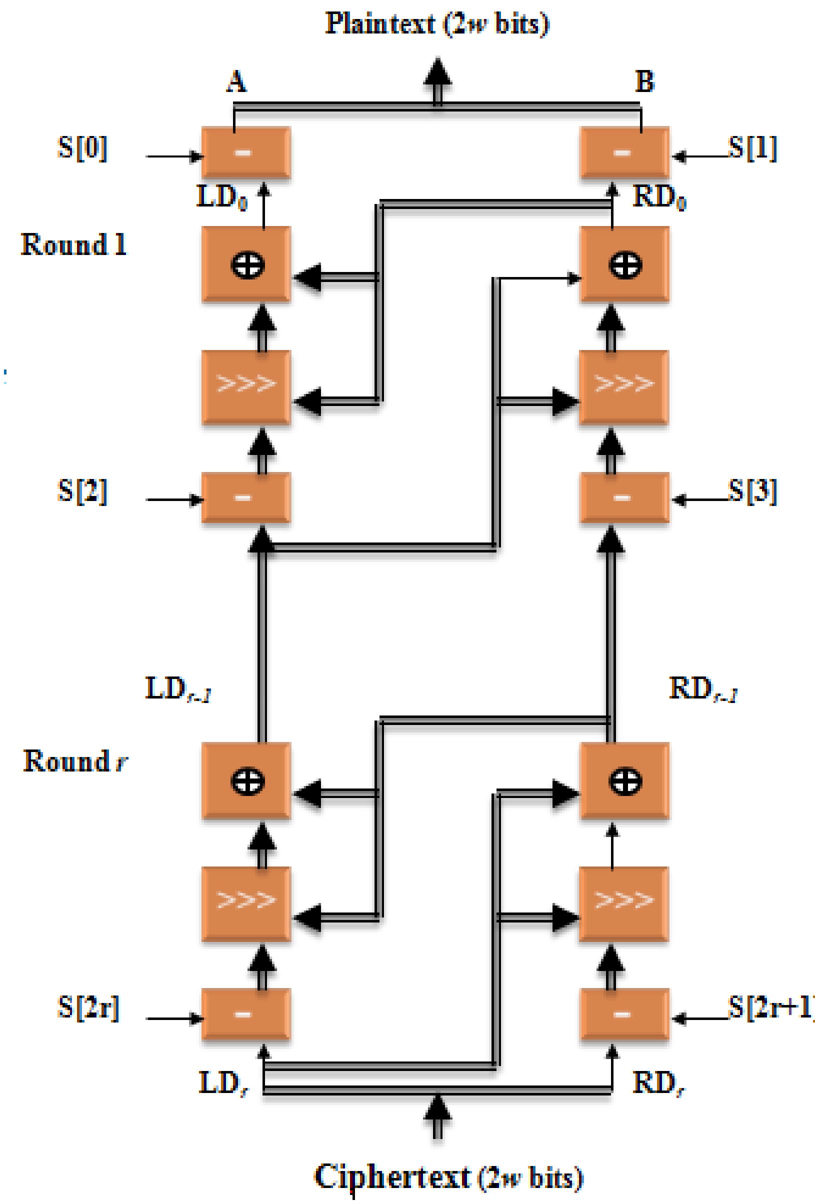
for $i = r$ **downto** 1 **do**

$$B = ((B - S[2 * i + 1]) \ggg A) \oplus A;$$
$$A = ((A - S[2 * i]) \ggg B) \oplus B;$$

$$B = B - S[1];$$
$$A = A - S[0];$$

Decryption

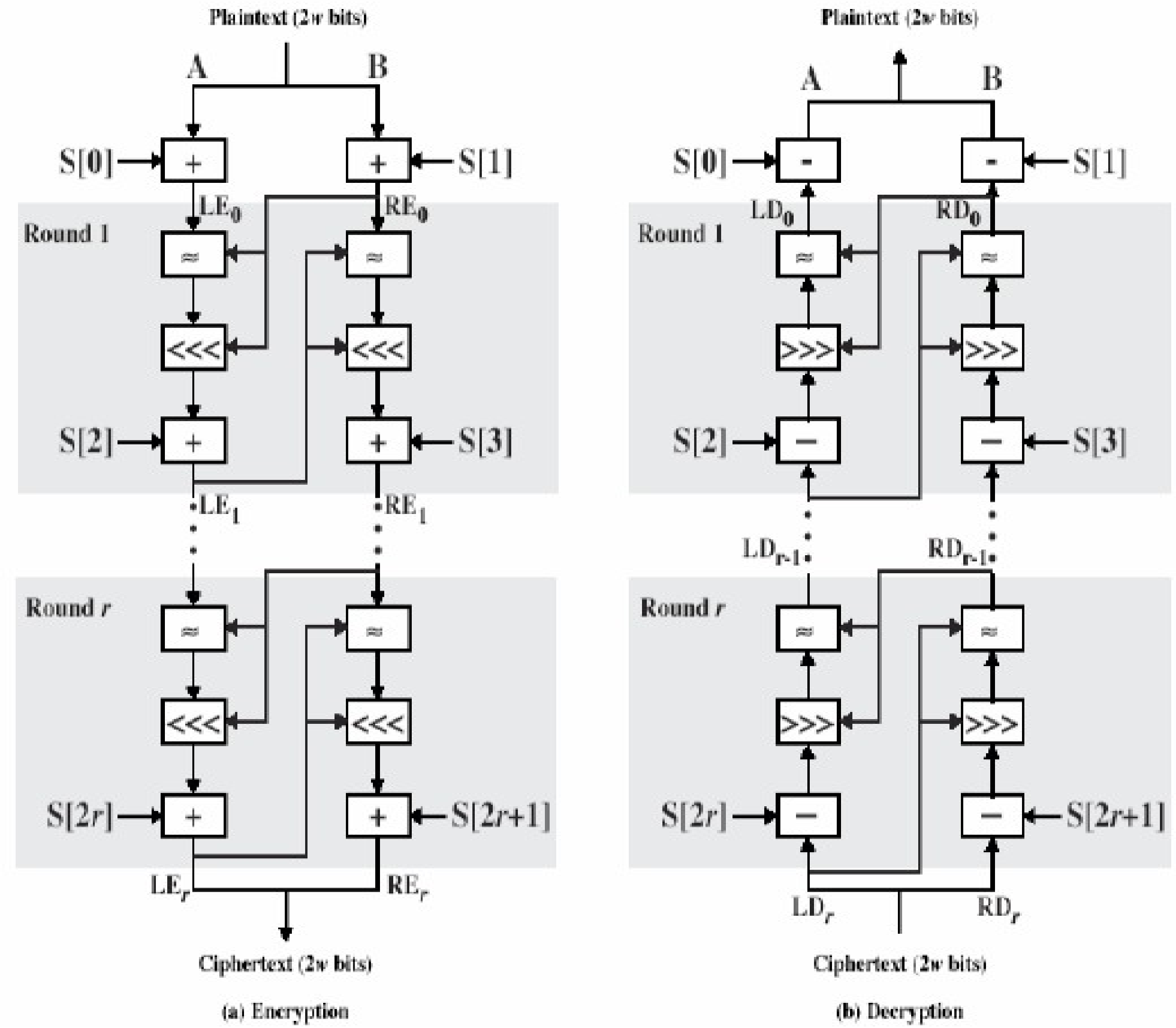
```
for i = r downto 1 do
    B = ((B - S[2*i +1]) >>> A) ⊕ A;
    A = ((A - S[2*i]) >>> B) ⊕ B;

B = B - S[1];
A = A - S[0];
```



A >>> B

Bits in A are rotated to right by the amount specified by lower log2(*w*) *bits in B*

Encryption and Decryption



Key Expansion

- RC5 performs some operations on the secret key to generate a total of t sub keys, which are stored in S array, S[0],S[1], …, S[t-1]
- The key expansion algorithm consists of two constants (Magic numbers) and three simple algorithm parts
  - Step-1: Convert secret key bytes to words
  - Step-2: Initialize sub key array S (S[0], S[1], …, S[t-1])
  - Step-3: Mix the secret key into sub key array S

Key Expansion

The key-expansion algorithm expands the user's key K to fill the expanded key table S, so that S resembles an array of $t = 2(r+1)$ random binary words determined by K. It uses two "magic constants" and consists of three simple algorithmic parts.

The two word-size magic constants P_w and Q_w are defined for arbitrary w as follows:

$$
\begin{aligned}
P_w &= \mathrm{Odd}((e-2)2^w) \\
Q_w &= \mathrm{Odd}((\phi-1)2^w)
\end{aligned}
$$

where

$$
\begin{aligned}
e &= 2.718281828459\ldots \text{ (base of natural logarithms)} \\
\phi &= 1.618033988749\ldots \text{ (golden ratio)} ,
\end{aligned}
$$

and where $\mathrm{Odd}(x)$ is the odd integer nearest to x (rounded up if x is an even integer, although this won't happen here).

The magic constants

- In key expansion, magic constants are used
  - Pw = Odd((e - 2)2w); e=2.718281828…. (base of natural logarithms)
  - Qw = Odd((ϕ - 1)2w);ϕ=1.618033988…. (golden ratio = (1+sqr(5))/2)
    - Odd(x): odd integer nearest to x
  - Example

w	16	32	64
P_w	B7E1	B7E15163	B7E151628AED2A6B
Q_w	9E37	9E3779B9	9E3779B97F4A7C15

Step-1: Convert secret key bytes to words

Generate L from the secret key K

K

00	01	02	03	04	05	06	07	08

L

03020100
07060504
00000008

Number of words in L $c = \lceil b/4 \rceil$ for 32-bit words

Copy the Key into new array L of Words with size equal c
Any unfilled byte positions of L are zeroed
In case $b = c = 0$ we reset $c = 1$ and set $L[0] = 0$

Step-2: Initialize sub key array S

- create an *expanded key table, S[0...t-1]*
  - has t entries, $t = 2(r + 1)$ w-bit words
- Initialize array *S*

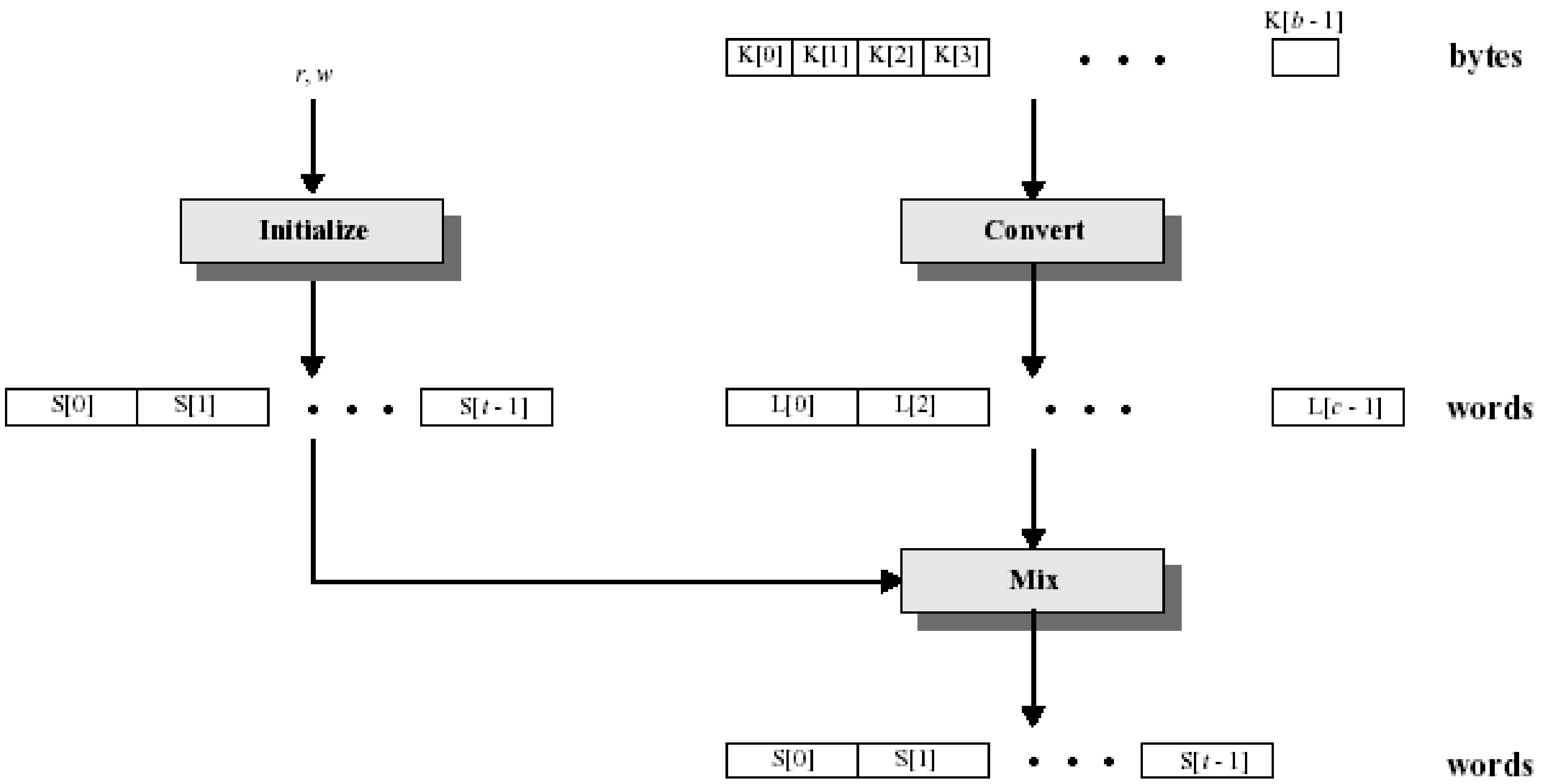
```
S[0] = P_w;
for i = 1 to t - 1 do
  S[i] = S[i - 1] + Q_w;
```

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

- Mix the secret key into table, *S*

```
i = j = 0;  A = B = 0;
do 3 * max(t, c) times:
      A = S[i] = (S[i] + A + B) <<< 3;
      B = L[j] = (L[j] + A + B) <<< (A + B);
      i = (i + 1) mod(t);
      j = (j + 1) mod(c);
```

Key Expansion Algorithm



Recap

- Introduction (Feistel Networks)
- What is RC5
- Parameterization
- Algorithm
- The security of RC5
- Conclusion

The security of RC5

The security of RC5

- Exhaustive Search
- Differential cryptanalysis
- Linear cryptanalysis
- Timing Attacks

Exhaustive Search

- RC5-32/r/b allows
  - a maximum of 2040 secret key bits
  - a maximum of 25(2r + 2) expanded key table bits
- Choosing large values for r and b can prevent exhaustive attacks

Differential cryptanalysis

- Pioneered by Biham and Shamir
- It has a quite evolutionary effect on the design and analysis of block ciphers
- The basic Idea
  - Two plaint text are chose with a certain difference P` (The difference here is measured by xor but for other cipher alternative measure may be applied)
  - The two plaintexts are enciphered to give two cipher texts such that their difference C`
  - Such a pair (P` , C`) is called a characteristic
  - Depending on the cipher and the analysis the behavior of this characteristics can be useful in deriving certain bit of the key

Linear cryptanalysis

- Introduced By Matsui.
- The basic idea is
  - to find relations among certain bits of plaintext, cipher text and key
  - Such as relation is called linear approximation which can be used to obtain information about the key
- Becomes impractical for $r > 6$

Differential and Linear attack

Number of plaintext/ciphertext pairs to break RC5

Rounds	4	6	8	10	12	14	16	18
RC5, Differential attacks	2^7	2^{16}	2^{28}	2^{36}	2^{44}	2^{52}	2^{61}	>
RC5, Linear attacks	2^{37}	2^{57}	>	>	>	>	>	>

Note: > means more than 2^{64} trials (brute-force) are required to break cipher

Timing Attacks

- Developed by Kocher
- The opponent can obtain some information about the secret key by recording and analyzing the time used for cryptographic operations that involve the key.
- Kocher found that RC5 may be subject to Timing attack if RC5 is implemented on platforms for which the time for computing a single rotation is proportional to the rotation amount
- RC5 can easily implemented to make the total time is data-independent (ex by computing the rotation of t bits using left-shift of t bits and right shift of w-t bits)

Conclusion

- Provides good security against the four main attacks
- Simple encryption/decryption algorithms
- RC5 is relatively is still under scrutiny by other cryptanalysis attack

Thank you for your attention