режимы работы DES

Кевролетин В.В.

27 декабря 2011 г.

Задание5.2

Условие

Продемонстрировать различные режимы использования DES (Mathematica, Scheme, Sage, ...).

Решение

Блочные шифры допускают использование в разных режимах:

режимом простой замены

Каждый блок открытого текста заменяется блоком шифротекста.
Код ниже реализует разбиени входной строки на блоки(по 16 16-ричных цифр) и заменяет каждый блок открытого текста блоком шифротекста:

```
...
package Des;
...

sub _process_blocks_stream {
    my ($self, $blocks, $funct) = @_;
    my @res;
    for (@$blocks) {
        push @res, $funct->($self, $_);
    }
    \@res
}

sub _process_hex_str {
    my ($self, $str, $funct) = @_;
    die "length is not multiple 16" if length($str) % 16;
    my @blocks;
    for (0 .. (int length($str)/16) - 1) {
        my $subs = substr($str, $_*16, 16);
        push @blocks, BitsArray::from_hex($subs);
    }
    $self->_process_blocks_stream(\@blocks, $funct)
}
```

```
sub encode_hex_str {
    my ($self, $str) = @_;
    my $res = $self->_process_hex_str($str, \&encode_block);
    join '', map { BitsArray::to_hex($_) } @$res
}

sub decode_hex_str {
    my ($self, $str) = @_;
    my $res = $self->_process_hex_str($str, \&decode_block);
    join '', map { BitsArray::to_hex($_) } @$res
}
```

Пример использования:

```
my $encoder = Des->new(key => '9474B8E8C73BCA7D');

my $pretty_print = sub {
    my ($str) = @_;
    die "length is not multiple 16" if length($str) % 16;
    for (0 .. (int length($str)/16) - 1) {
        my $subs = substr($str, $_*16, 16);
        print $subs, " ";
    }
    print "\n";
};

my $open = 'abcdabcdabcdabcdabcdabcdabcdabcdabcdabcdabcdabcd';
my $cipher = $encoder->encode_hex_str($open);
$pretty_print->($cipher);
my $res = $encoder->decode_hex_str($cipher);
$pretty_print->($res);
```

Результат

```
50dc14fa03fb808c 50dc14fa03fb808c 50dc14fa03fb808c
abcdabcdabcdabcd abcdabcdabcdabcd abcdabcdabcdabcd
```

Режим сцепления блоков шифротекста

Каждый блок открытого текста (кроме первого) побитово складывается по модулю 2 (операция XOR) с предыдущим результатом шифрования. Шифрование:

Реализация

```
package Des::CBC;
use Moose;

extends 'Des';

sub _process_blocks_stream {
```

```
    my ($self, $blocks, $encode) = @_;
    my @res;
    my $a;
    if ($encode) {
        for (@$blocks) {
            my $b = $self->encode_block($_);
            push @res, $a ? BitsArray::map_xor($a, $b) : $b;
            $a = $_;
        }
    } else {
        for (@$blocks) {
            my $b = $self->decode_block($_);
            $b = BitsArray::map_xor($a, $b) if $a;
            push @res, $b;
            $a = $b;
        }
    }
    \@res
}
```

Результат для того же примера(шифротекст/расшифрованный открытый текст):

```
50dc14fa03fb808c fb11bf37a8362b41 fb11bf37a8362b41
abcdabcdabcdabcd abcdabcdabcdabcd abcdabcdabcdabcd
```

Режим обратной связи по шифротексту

Для шифрования следующего блока открытого текста он складывается по модулю 2 с перешифрованным (блочным шифром) результатом шифрования предыдущего блока.

Реализация

```
package Des::CFB;
use Moose;

extends 'Des';

sub _process_blocks_stream {
    my ($self, $blocks, $encode) = @_;
    my @res;
    if ($encode) {
        for (@$blocks) {
            my $b = $self->encode_block($_);
            push @res,
                (@res ? BitsArray::map_xor($res[-1]   , $b) : $b);
        }
    } else {
        for my $i (0 .. $#{$blocks}) {
            $_ = $blocks->[$i];
            $_ = BitsArray::map_xor($blocks->[$i - 1], $_) if $i;
```

```
            push @res, $self->decode_block($_)
        }
    }
    \@res
}

1;
```

Результат для того же примера(шифротекст/расшифрованный открытый текст):

```
50dc14fa03fb808c 0000000000000000 50dc14fa03fb808c
abcdabcdabcdabcd abcdabcdabcdabcd abcdabcdabcdabcd
```

Режим обратной связи по выходу

Генерируются ключевые блоки, которые складываются с блоками открытого текста

Реализация

```
package Des::OFB;
use Moose;

extends 'Des';

has 'init_vector' => ( isa => 'Key|Str',
                       is => 'ro',
                       required => 1 );

before 'BUILD' => sub {
    my ($self) = @_;
    unless (ref($self->{init_vector})) {
        $self->{init_vector} =
            BitsArray::from_hex($self->{init_vector});
    }
};


sub _process_blocks_stream {
    my ($self, $blocks, $encode) = @_;
    my @res;
    my $z = $self->encode_block($self->init_vector());
    for (@$blocks) {
        push @res, BitsArray::map_xor($_, $z);
        $z = $self->encode_block($z);
    }
    \@res
}
```

Результат для того же примера(шифротекст/расшифрованный открытый текст):

266aef2d6283f5da ff274d4d4a60d943 adc6221895611d4d
abcdabcdabcdabcd abcdabcdabcdabcd abcdabcdabcdabcd