KRIPTOGRAFIJA
Zadaća 4.132

Filip Nikšić

14. svibnja 2007.

1. Odredite skupove $test_1(E_1, E_1^*, C_1')$ i $test_2(E_2, E_2^*, C_2')$ ako je

$$E_1 = 101110, \qquad E_1^* = 111010, \qquad C_1' = 0001$$
$$E_2 = 101111, \qquad E_2^* = 000001, \qquad C_2' = 1001\,.$$

Rj. Zadatak rješavamo programom *des.pl* napisanim u Perlu. Glavna (zapravo, i jedina) petlja u programu je sljedeća:

```
for (my $B=0; $B<64; $B++) {
    if ((S($j,$B)^S($j,$B^$EE))==$CC) {
        printf "%06b ", $B^$E;
    }
}
```

Varijabla `$j` predstavlja broj S-kutije, `$E`, `$EE` i `$CC` predstavljaju naše E_j, E_j' i C_j', a učitavaju se s komandne linije. (Točnije, umjesto E_j' učita se E_j^* pa se $E_j' = E_j \oplus E_j^*$ izračuna.) Za one `$B` za koje je zadovoljen definicijski uvjet skupa $IN_j(E_j', C_j')$ ispisuje se `$B^$E`. Ostaje razjasniti funkciju `S`:

```
sub S {
    my ($j,$B)=@_; # broj S-kutije i 6-bitni niz
    my $v=($B & 0b100000)>>4 | ($B & 1); # prvi i sesti bit
    my $c=($B & 0b11110)>>1; # srednji bitovi
    return $Sbox[$j][16*$v+$c];
}
```

Mislim da je i tu sve jasno. Iz niza `$B` ekstrahiraju se bitovi na odgovarajući način i vrati se broj zapisan na traženom mjestu u tablici S_j.

Izlazi programa za ulaze zadane u zadatku:

```
$ ./des.pl 1 0b101110 0b111010 0b0001
100000 110100 001010 000010 000000 011110 010110 010100

$ ./des.pl 2 0b101111 0b000001 0b1001
101100 000010
```

Prema tome:

$$test_1(E_1, E_1^*, C_1') = \{100000, 110100, 001010, 000010,\\ 000000, 011110, 010110, 010100\},$$
$$test_2(E_2, E_2^*, C_2') = \{101100, 000010\}\,.$$

2. Izračunajte:

$$(\texttt{0xfa}, \texttt{0xa1}, \texttt{0x1a}, \texttt{0x2b}) \otimes (\texttt{0x3f}, \texttt{0x7a}, \texttt{0x3c}, \texttt{0x1a})\,.$$

Rj. Zadatak možemo riješiti Perl programom *aes.pl*. Vektori

$$\texttt{@a} = (\texttt{0xfa}, \texttt{0xa1}, \texttt{0x1a}, \texttt{0x2b})$$
$$\texttt{@b} = (\texttt{0x3f}, \texttt{0x7a}, \texttt{0x3c}, \texttt{0x1a})$$

unose se preko komandne linije.

Označimo rezultat s `@d`. Komponente od `@d` računamo na sljedeći način:

```
my @d=(mnozi($a[0],$b[3])^mnozi($a[1],$b[2])^
            mnozi($a[2],$b[1])^mnozi($a[3],$b[0]),
       mnozi($a[1],$b[3])^mnozi($a[2],$b[2])^
            mnozi($a[3],$b[1])^mnozi($a[0],$b[0]),
       mnozi($a[2],$b[3])^mnozi($a[3],$b[2])^
            mnozi($a[0],$b[1])^mnozi($a[1],$b[0]),
       mnozi($a[3],$b[3])^mnozi($a[0],$b[2])^
            mnozi($a[1],$b[1])^mnozi($a[2],$b[0]));
```

pri čemu je indeksiranje komponenti obrnuto u Perlu od onog kojeg smo mi koristili.

Operacija `^` je standardni `XOR`, tj. zbrajanje u prstenu $\mathbb{Z}_2[x]$. Pogledajmo na primjeru kako se jednostavno realizira množenje polinoma kodiranih pomoću binarnih brojeva. Pomnožit ćemo `0xfa` i `0x1a`:

```
11111010 x 11010
----------------
11111010
 11111010
  00000000
   11111010
    00000000 XOR
----------------
100110000100 = 0x984
```

Potpisivanje radimo tako da za svaku znamenku 1 u desnom broju potpišemo lijevi broj, a za svaku znamenku 0 potpišemo niz 0, pri čemu se u svakom koraku pomičemo za jedno mjesto udesno. Na kraju dobivene retke `XOR`-amo. Evidentno je da je tako izvedeni zapis množenja ekvivalentan standardnom načinu zapisivanja. Pogledajmo kako se to lako implementira u Perlu korištenjem *bit-shifting* operatora:

```
sub mnozi {
    my ($a, $b)=@_; # prenosimo parametre
    my $result=0;
    while ($b) {
        # Ako je posljednji bit od $b jedinica, xoramo s $a:
        if ($b & 1) {$result^=$a}
        $a<<=1; # U svakom slucaju
        $b>>=1; # shiftamo brojeve.
```

```
    }
    return $result;
}
```

Ostaje reducirati svaku komponentu od `@d` polinomom `0x11b`. Pogledajmo opet na primjeru kako se to lako napravi iskorištavanjem zapisa polinoma pomoću binarnih brojeva. Reducirat ćemo u prethodnom primjeru dobiveni polinom `0x984`:

```
100110000100 : 100011011
100011011
------------
000101011100
   100011011
   ---------
   001000111
```

Rezultat dijeljenja nas uopće ne zanima, tražimo samo ostatak. U svakom koraku potpisujemo `0x11b` tako da mu prvi koeficijent dođe ispod prvog koeficijenta trenutnog polinoma i `XOR`-amo. Postupak ponavljamo dok ne dobijemo polinom stupnja najviše 7. Pogledajmo sad kako to izgleda u Perlu:

```
foreach my $component (@d) {
    my $p=16; # potencija
    while ($component>=0x100) {
        while (!($component & 1<<$p)) {$p--}
        $component^=0x11b<<($p-8);
    }
}
```

Varijabla `$p` označava nam potenciju uz koju se nalazi najstariji koeficijent polinoma `$component`. To koristimo kako bismo mogli poravnati naš polinom `0x11b` ispod `$component`.

Pogledajmo sad izlaz programa za vektore zadane u zadatku:

```
$ ./aes.pl 0xfa 0xa1 0x1a 0x2b 0x3f 0x7a 0x3c 0x1a
(0x78,0x16,0xdc,0x26)
```

Prema tome, imamo:

$$(\texttt{0xfa}, \texttt{0xa1}, \texttt{0x1a}, \texttt{0x2b}) \otimes (\texttt{0x3f}, \texttt{0x7a}, \texttt{0x3c}, \texttt{0x1a}) = (\texttt{0x78}, \texttt{0x16}, \texttt{0xdc}, \texttt{0x26})\,.$$