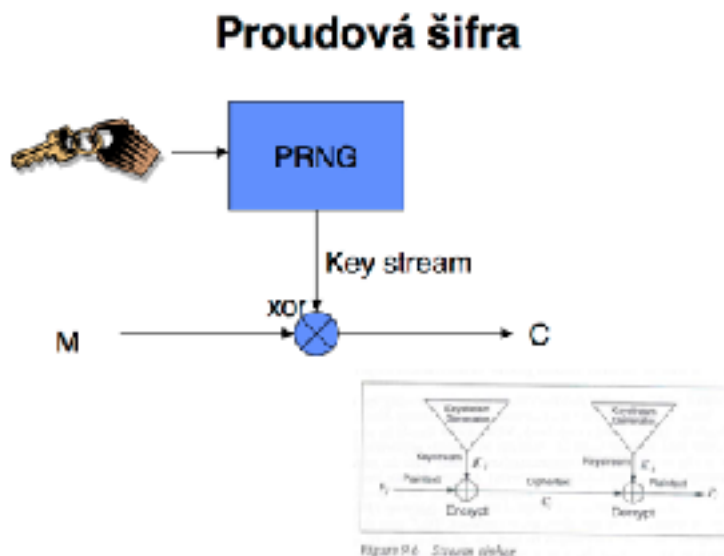


1. projekt KRY 2017/2018

Jordán Jarolím, xjarol03

Cílem 1. projektu KRY bylo získat klíč, kterým byly zašifrované soubory ze zadání. Názvy souborů napověděly, že máme k dispozici otevřenou a šifrovanou verzi minimálně jednoho souboru. Ze 3. přednášky, slide 94, se dá odvodit, že šifrovaný soubor vznikl operací XOR s výstupem generátoru



pseudonáhodných čísel, který je však závislý na vstupním (námi hledaném) klíči. Z vlastností operace XOR se dá odvodit (<https://crypto.stackexchange.com/questions/13263/cracking-stream-cipher>):

$$plain1 (XOR) cipher1 (XOR) cipher2 = plain2$$

Po provedení takové operace se soubory *bis.txt*, *bis.txt.enc* a *super_cipher.py.enc* se na výstupu objeví část pythonovského skriptu. Pro další část skriptu je možné použít XOR na dva zašifrované soubory, a to opět díky vlastnostem operace xor a neměnnosti vstupního klíče (<https://cryptosmith.com/2008/05/31/stream-reuse/>):

$$\begin{aligned} plain1 (XOR) key &= cipher1 \\ plain2 (XOR) key &= cipher2 \\ plain1 (XOR) plain2 &= cipher1 (XOR) cipher2 \end{aligned}$$

Tímto dostaneme druhou část skriptu. Lze vyčíst, že hlavním tělem skriptu je funkce `step`, která slouží jako již zmiňovaný generátor pseudonáhodných čísel se vstupním klíčem `x` a která se volá v několika iteracích:

```
def step(x):  
    x = (x & 1) << N + 1 | x << 1 | x >> N - 1  
    y = 0  
    for i in range(N):  
        y |= SUB[(x >> i) & 7] << i  
    return y
```

Dá se jednoduše získat výstupní hodnota `y` této funkce (keystream), a sice použitím XOR na *bis.txt* a *bis.txt.enc*. Dalším krokem pro získání vstupního klíče `x` je napsání reverzní funkce k funkci `step`.

Při bližším ohledání funkce vidíme, že na začátku probíhá rozšíření o jeden bit vlevo a jeden bit vpravo - hodnoty těchto bitů jsou určeny hodnotami nejlevějšího a nejpravějšího bitu vstupního bitového vektoru. Ze vstupu *1001 1001* udělá výstup *1 1001 1001 1*. Pro reverzi tedy stačí provést operaci shift doprava o jeden bit a poté, pokud je délka vektoru větší než 256 bitů, zahodit i ten nejlevější bit. Tato podmínka je důležitá, neboť pokud by byla na začátek vektoru přidána 0, python ji defaultně zahodí a 256-bitový vektor bychom tak zkrátili na 255, což je nežádoucí.

Nejprve je však nutné vůbec tento rozšířený vektor získat. Při důkladné analýze se dá zjistit, že *For* cyklus provádí následující operace:

1. na operaci (*x >> i*) lze vidět, že se posouvá ve vstupním vektoru doleva
2. získá 3 nejpravější bity vektoru, ukazujícího do SUB
3. výběr elementu z vektoru SUB
4. posun elementu SUB o 1 doleva
5. OR s existujícím vektorem *y*

Dle výše popsáného jde vidět, že v kroku *i=0*, bude vektor *y* mít hodnotu adresovaného prvku SUB. Ve 2. kroku se vstupní vektor posune o 1 doprava, získá se nový ukazatel do pole SUB, výsledek se posune o *i* pozic doleva a provede se OR. Mějme tedy *x = 1001*, potom tedy:

$$0001 \text{ (OR) } 0010 = 0011$$

Při reverzi tedy stačí jít zprava, podívat se na aktuální bit, ten je roven 1. Hodnota 1 se ve vstupním poli SUB vyskytuje na pozicích 1, 2, 4, 6 - získáme tedy 4 možnosti, ty si uložíme jako:

list1 = [001, 010, 100, 110]

Posuneme se doleva, tam narazíme opět na 1, ta nám dává opět možnosti

list2 = [001, 010, 100, 110]

Nyní uděláme kartézský součin těchto dvou listů, přičemž validní jsou hodnoty takové, kde nejpravější 2 bity *list2* se překrývají s nejlevějšími 2 bity *list1*. Pokud najdeme takovou shodu, nalezený prvek *list1* rozšíříme zleva o nejlevější prvek *list2*. Dostáváme tedy

list3 = [1001, 0010, 0100, 1100]

Tím získáváme 4 vektory, které se rozšiřují v dalších krocích podle stejného principu. Nakonec stačí provést reverzi shiftování z prvního řádku funkce a tím získat vstupní klíč *x*. Je také nutné zavolat reverzní funkci *step* ve stejném počtu jako původní funkci *step*. Poté lze získat vstupní tajemství které má tvar:

KRY{xjarolo3-ca5ddc0bc64761c}