1. projekt do předmětu KRY

Sára Škutová, xskuto00

5. dubna 2019

1 Analýza souborů z archívu

Soubory v archívu jsou zašifrované synchronní proudovou šifrou, kdy se otevřený text **xoruje** s posloupností nazvanou **keystream**. Výsledkem jsou tedy soubory s nečitelným textem. Ovšem při šifrování těchto souborů byla porušena jedna z podmínek kdy se pro generování keystremu pro jednotlivé soubory nesmí použít stejný inicializační vektor.

To, jakým keystreamem K byl text zašifrovaný lze zjistit, když zxorujeme zašifrovaný text souboru *bis.txt.enc.* C_1 s odpovídajícím otevřeným textem ze *bis.txt* P_1 (rovnice 1).

$$C_1 \oplus P_1 = K \tag{1}$$

Díky takto získanému klíči lze dešifrovat část souboru *super_ cipher.py.enc* (použije se transformována rovnice 1, kdy se zašifrovaný text zxoruje s klíčem, klíč ovšem není dostatečně dlouhý, a proto získáme pouze část otevřeného textu, zbytek bude stále zašifrovaný). Tímto zjistíme, že se jedná o pythonovský skript ve kterém se pomocí funkce `step(x)` a pole `SUB` v $N//2$ iteracích generuje první část keystreamu.

```
SUB = [0, 1, 1, 0, 1, 0, 1, 0]
N_B = 32
N = 8 * N_B

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
```

V první částí funkce `step` bitově rozšíří vstupní číslo, na začátek bitové posloupnosti zkopíruje bit co se nachází na konci a následně ke konci posloupnosti přidá – z čísla 1110 se tedy stane 11100 (normálně by posloupnost byla dlouhá 265 bitů, ale jelikož se na začátek zkopírovala 0, tak je zápis zkrácený). V druhé částí pak N iteracích určí výsledná hodnota, která se určuje pomocí pole `SUB`, vypočítané posloupnosti z první částí a hodnoty z předcházející iterace. Zkráceně lze říci, že v každé iteraci se posouvá bitová posloupnost proměnné `x`, pomocí jejíchž posledních tří bitů se určí index do pole `SUB`, hodnota z tohoto pole se pak připojí na začátek k bitové posloupnosti hodnoty, která se vypočítala v předcházející iteraci.

2 Reverze funkce step

Aby se získal inicializační vektor, tak se musí napsat funkce, která bude provádět reverzní práci funkce `step(x)`. Na začátku se z dříve vypočítaného keystreamu převezme první 32 bytů, které se transformují tak aby nevyšší bit se nacházel na nejnižší pozici (celá posloupnost bitů se otočí). Následně se tato posloupnost pošle do funkce `reverseStep()`, která se bude provádět $N//2$ krát.

V `reverseStep()` se vstupní hodnota `y` čte od konce po jednotlivých bitech, přičemž každý bit je ekvivalentní s hodnotou získanou z pole `SUB`. Binární číslo indexu pak určuje hodnotu proměnné `x`. Při čtení prvního bitu ovšem nevíme, která ze čtyř možností je správná, a proto se pracuje se všemi čtyřmi. Při čtení následujících již víme, protože poslední dva bity předcházející posloupnosti odpovídají prvním dvěma bitům indexů, které vybíráme. Z vybraného indexu pak převezmeme

poslední bit a připojíme ho na konec předcházející posloupnosti proměnné `x` (tato oprace – výběr indexu a připojení bitu se provádí pro všechny čtyři možnosti). Takto se zpracuje celá vstupní hodnota `y`.

Následně se ještě provede reverzace prvního řádku funkce, kdy se odstraní poslední nejméně významný bit a pokud je posloupnost delší než 256 bitů, tak se zkrátí právě na těchto 256. Tato operace se opět provede pro všechny čtyři možnosti.

Jako poslední operací zbývá vybrat, která z možností je ta správná. To lze jednoduše tím, že se pro každou vypočítá hodnota pomocí funkce `step(x)`. Následně stačí jenom porovnat s původní hodnotou co vstupovala do `reverseStep()`. Pouze jedna možnost vytvoří hodnotu, která je stejná jako původní hodnota. Tato možnost se následně pošle do další iterace `reverseStep()`.

3 Inicializační vektor

Po $N//2$ iteracích funkce `reverseStep()` a úpravě výsledné hodnoty získáme původní inicializační vektor: **KRY{xskuto00-d67d7b42e0a3c96}**