Kryptografie – dokumentace k projektu 1

Vysoké učení technické v Brně

Petr Stehlík <xstehl14@stud.fit.vutbr.cz> 11. dubna 2018

1 Zadání

Cílem projektu bylo zjistit tajemství z několika souborů, které jsou šifrovány neznámou synchronní proudovou šifrou.

2 Získání klíče manuální metodou

V obdržených souborech se nachází dvojice souborů `bis.txt` a `bis.txt.enc`, kde první z nich je nezašifrovaný textový soubor (plaintext) a druhý je šifrovaný zatím neznámou synchronní šifrou (ciphertext).

2.1 Získání keystream

Po aplikovaní binární operace `XOR` na soubory `bis.txt` a `bis.txt.enc` jsme schopni získat prvních 512 B keystreamu, díky tomu, že platí vztah:

$$ciphertext = plaintext \ \texttt{XOR} \ keystream$$

Kde po úpravě dostaneme:

$$keystream = plaintext \ \texttt{XOR} \ ciphertext$$

2.2 Dešifrování `super_cipher.py.enc`

S pomocí získáného keystreamu jsme schopni dešifrovat část zašifrovaného souboru `super_cipher.py.enc` (prvních 512 B) tím, že na ciphertext a keystream aplikujeme opět operaci `XOR`.

Tato operace odkryje globální proměnné a funkci `step`.

Listing 1: Část dešifrovaného `super_cipher.py`

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

# Next keystream
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
```

2.3 Analýza `super_cipher.py`

Funkce step přijímá klíč x o délce `N` bitů, tento klíč rozšíří o další 2 bity, původní klíč posune o 1 bit směrem k MSB a původní MSB se zduplikuje na LSB. Tím vznikne x', které je následně použito pro vygenerování keystreamu y. Pro generování y je použito pole `SUB` a 3 bitů z x', které jsou použity pro indexování v poli `SUB`, ze kterého se získá bit, který je uložen do y. Takto vygenerovaný keystream je poté vrácen.

Všechny operace ve funkci `step` jsou reverzovatelné a tudíž bude možné získat počáteční klíč.

2.4 Dešifrování dalších souborů

Pro dešifrování souborů delších než 512 B je potřeba keystream vygenerovat ve stejné velikosti jako je daný plaintext. Po experimentování s různými způsoby bylo odhaleno, že pokračování keystreamu lze vytvořit aplikováním funkce `step` na první blok keystreamu velikosti `N` bitů získáného v předchozích krocích.

Díky tomuto postupu bylo možné úspěšně dešifrovat kompletní obsah souborů `super_cipher.py.enc` a `hint.gif.enc`. Po dešifrovaní Python skriptu a obrázku vidíme jak je šifra aplikovaná na celý plaintext a jak je použitý klíč.

```
for file in *.*
do ./super_cipher.py "KRY(
done
```

```
)" < $file > $file.enc
```

Obrázek 1: Dešifrovaný soubor `hint.gif`

2.5 Získání tajemství

Pro získání původního klíče je nutné reverzovat funkci `step`. Pro určení bitu na dané pozici musíme určit 3 bity původního keystreamu. To celkově vede ke 4 možnostem jak daný bit mohl být určený indexací do pole `SUB`. Nicméně každý bit klíče sdílí 2 bity původního keystreamu se svými okolními bity. To značně usnadňuje vyhledání správného bitu.

Reverzování algoritmu ve funkci `step` vyzkouší všechny možné kombinace, které k danému bitu mohly vést a pokud první dva a poslední dva bity jsou stejné, jde o kandidáta na získání původního klíče, kde k existujícím kandidátům přidáme danou trojici bitů, které indexovaly zkoušený bit.

Jakmile máme všechny možné kandidáty využijeme znova této vlastnosti a najdeme klíč, který má dva LSB a dva MSB stejné.

Na tento klíč následně aplikujeme invertované operace pro modifikaci x z funkce `step` a tím získáváme námi hledaný klíč.

Je zde možnost nalezení kolizního klíče, která je ale v případě 32 B klíče velmi malá a klíč získaný z obdržených souborů není kolizní, proto tuto situaci dešifrovací skript nezohledňuje.

3 Ovládání `solution.py`

Program `solution.py` disponuje přepínačem `-d`, který dešifruje soubory ve složce `in/` a uloží je do této složky.

4 Závěr

Manuální řešení získávání počatečního klíče bylo úspěšně implementováno a bylo získáno tajemství, které je pro poskytnuté soubory následující: `KRY{xstehl14-51f610e23c2b4e9}`.

Řešení pomocí SAT nebylo implementováno.