Kryptografie
Projekt č. 1

Nikola Valešová
xvales02@stud.fit.vutbr.cz

7. dubna 2018

1 Úvod

Cílem projektu je získat tajemství, které bylo použito pro vygenerování klíče k zašifrování několika souborů nám neznámou synchronní proudovou šifrou. Jedná se o provedení tzv. „known plaintext attack“, jelikož je kromě tří zašifrovaných souborů k dispozici také nezašifrovaný obsah jednoho z nich.

2 Ruční řešení

Ruční způsob řešení problému sestával ze dvou částí. Cílem první části bylo získání keystreamu, který byl použit pro šifrování souboru *bis.txt*. Bylo tedy třeba načíst soubory *bis.txt* a *bis.txt.enc* jako bitová pole a na bitové úrovni provést jejich xor. Tímto postupem jsme získali prvních 512 bytů keystreamu, který byl použit pro zašifrování souboru *bis.txt*. Zkoumání získaného keystreamu nám ale žádné další informace neposkytlo.

Ve druhé části řešení bylo cílem ze sekvence keystreamu získat tajemství – klíč, který byl použit pro inicializaci algoritmu na vytvoření šifrovacího klíče. Pomocí duplikace již získaných bytů keystreamu bylo možné provést xor se souborem *super_cipher.py.enc* a získat tak prvních 512 bytů dešifrovaného algoritmu pro tvorbu keystreamu. Zbylé znaky byly také úspěšně dešifrovány, až na pár vyjímek, jejichž pravou hodnotu bylo možné odhadnout. Tímto postupem bylo možné získat dešifrovaný soubor *super_cipher.py*, ze kterého se dozvídáme postup, jakým byl vstupní klíč modifikován pro získání keystreamu. Po úpravě získáváme skript, který umí šifrovat a dešifrovat soubory danou šifrou, vyžaduje k tomu však správný počáteční klíč.

Při analýze algoritmu pro generování keystreamu zjistíme, že je inicializován nějakým klíčem, poté je vygenerováno 128 subklíčů, z nich každý je závislý na tom předchozím, a poté je použit výsledný keystream. Generování následujícího subklíče probíhá ve funkci `step`. Na prvním řádku je vstupní subklíč upraven tak, že jsou zduplikovány hodnoty nejvyššího a nejnižšího bitu a subklíč je tak rozšířen o 2 bity tak, že subklíč $(n-1, \ldots, 1, 0)$ je převeden na sekvenci $(0, n-1, \ldots, 1, 0, n-1)$. Ze získané sevence je následně generován nový subklíč, a to tak, že hodnota každého n-tého bitu nového subklíče závisí na hodnotě odpovídajícího n-tého bitu a jeho dvou sousedních bitů $n+1$ a $n+2$ v předchozím subklíči.

Získání tajemství pomocí ručního řešení je tedy založeno na reverzaci výše popsaného principu, kdy na základě hodnoty n-tého bitu nového subklíče a hodnot dvou sousedních bitů $n+1$ a $n+2$ původního subklíče získáváme hodnotu n-tého bitu původního subklíče. Po 128 iteracích reverzovaného algoritmu funkce `step` získáváme 32 bitů inicializačního klíče, jehož prvních 29 bitů obsahuje hledané tajemství.

3 SAT solver

Skript implementující řešení s využitím SAT solveru se od ručního řešení liší pouze ve způsobu řešení reverzace kroků při vytváření keystreamu. Opět jsou tedy nejprve načteny soubory *bis.txt* a *bis.txt.enc* a je proveden jejich xor pro získání finálního keystreamu. Poté jsou aplikovány kroky samotné reverzace keystreamu, která je v tomto případě řešena s využitím SAT solveru. Na základě již vypozorovaných bitových závislostí mezi subklíči je vytvořena sada logických výrazů, ve kterém je pro každý bit předcházejícího subklíče vyhrazena jedna proměnná. Ke každému bitu předchozího

subklíče je sestaven odpovídající logický výraz, přičemž pro správné řešení musí platit všechny dílčí výrazy současně. Pokud je hodnota n-tého bitu následujícího subklíče rovna 1, odpovídá n-tému bitu předchozího subklíče výraz 1, v opačném případě je n-tému bitu přiřazen výraz 2.

$$\left(x_i \wedge \neg x_{(i+1)\%N} \wedge \neg x_{(i+2)\%N}\right) \vee \left(\neg x_i \wedge x_{(i+1)\%N}\right) \vee \left(\neg x_i \wedge x_{(i+2)\%N}\right) \quad (1)$$

$$\left(\neg x_i \wedge \neg x_{(i+1)\%N} \wedge \neg x_{(i+2)\%N}\right) \vee \left(x_i \wedge x_{(i+1)\%N}\right) \vee \left(x_i \wedge x_{(i+2)\%N}\right) \quad (2)$$

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

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

Obrázek 1: Dešifrovaný soubor *hint.gif.enc*, který nám říká, jakým způsobem byly soubory šifrovány a že hledané tajemství spočívá v klíči, kterým byl generátor keystreamu inicializován

4 Implementace

Implementace ručního řešení využívá pouze standardních funkcí Pythonu 3. Pro implementaci SAT solveru byla využita knihovna satispy a SAT solver MiniSAT, pro který knihovna satispy poskytuje rozhraní. Všechny potřebné závislosti jsou doinstalovány pomocí přiloženého skriptu *install.sh*.

5 Závěr

V rámci tohoto projektu byla prolomena synchronní proudová šifra útokem „known plaintext attack“, a to jak ručním řešením, tak i prostřednictvím SAT solveru. Výsledné tajemství, tedy klíč použitý pro inicializaci algoritmu na generování keystreamu, je „KRY{xvales02-7323baa84d548a0}“.