Kryptografie - 1. projekt
Prolomení symetrické proudové šifry

Petr Nodžák

3. dubna 2019

1 Získání klíče

Ze znalosti plain textu *bis.txt* a cipher textu *bis.txt.enc* pomocí logické oprace XOR bylo možné získat 32B klíč, kterým byl soubor *bis.txt* zašifrovaný. Ostatní soubory byly šifrované stejným klíčem, tudíž bylo možné dešifrovat i jejich část. Bylo tedy nutné přijít na způsob, jak byl tento klíč prodloužen na stejnou délku, jakou má šifrovaný soubor, aby bylo možné ho celý dešifrovat. Šifravací algoritmus byl skrytý v souboru *super_cipher.py*, ten bylo možné dešifrovat už s dřív získaným 32B klíčem. Se znalostí tohoto algoritmu jsem rozšířil klíč na délku souboru *hint.git.enc* a dešifroval ho, v něm se skrýval obrázek, který popisoval, kde vůbec požadovaný secret hledat.

2 Ruční řešení

Bylo nutné reverzovat algoritmus, který vytvořil prvních 32B klíče a získat jeho vstup. Prochází se binární řetězec od MSB (most significant bit), pokud je $1.bit = 0$, (resp. $1.bit = 1$) vezme se jeden z řetězců $zeroes$ = [”000”, ”011”, ”101”, ”111”], (resp. $ones$ = [”001”, ”010”, ”100”, ”110”]) a uloží do proměnné *revert*, následně se čte bit $i+1$ a provede to stejné jen s rozdílem, že poslední dva znaky z proměnné *revert* se musí rovnat prvním dvoum znakům z ones, (resp. zeroes) podle toho, zdá byl bit $i+1 = 1$ (resp. $i+1 = 0$), pokud se rovnají, přidá se do proměnné *revert* poslední bit z porovnávaného řetězce, takto se sestaví celých 32*8 bitů, které se vloží do funkce step() a porovnají se s bity vstupu reverzní funkce, pokud se shodují, tak máme správný předchozí stav algoritmu, pokud ne, vezme se jiný počáteční řetězec ze *zeroes* nebo *ones*, podle toho zda byl první bit 0 nebo 1. Tohle se opakuje $(32 * 8)//2$ krát, získá se původní vstup a vypíše se.

3 SAT solver

Druhou část zadání tvořil SAT solver, kterým se dá dostat ke vstupu algoritmu se znalostí výstupu a algoritmu. Samotná imlementace je hodně podobná

ručnímu řešení. Trik je v tom, vytvořit formule a ptát se SAT solveru, jestli jsou splnitelné. K tomuto jsem využil z3 solver, který umí říct, zda je formule splnitelná a při jakém ohodnocení proměnných. Bylo potřeba vytvořit pro každý bit 32B klíče ($key = x_0x_1...x_{255}$) proměnnou $Bool('x_i')$. K získání předchozího kroku, se čtou všechnt bity aktuálního klíče, bit po bitu, pro každý bit se vytvoří formule podle následujících dvou pravidel, pokud je bit i nulový použije se formule $(\neg x_i \wedge \neg x_{(i+1)\%N} \wedge \neg x_{(i+2)\%N}) \vee (x_i \wedge x_{(i+1)\%N} \wedge \neg x_{(i+2)\%N}) \vee (x_i \wedge \neg x_{(i+1)\%N} \wedge x_{(i+2)\%N}) \vee (x_i \wedge x_{(i+1)\%N} \wedge x_{(i+2)\%N})$, pokud 1 tak $(x_i \wedge \neg x_{(i+1)\%N} \wedge \neg x_{(i+2)\%N}) \vee (\neg x_i \wedge x_{(i+1)\%N} \wedge \neg x_{(i+2)\%N}) \vee (\neg x_i \wedge \neg x_{(i+1)\%N} \wedge x_{(i+2)\%N}) \vee (\neg x_i \wedge x_{(i+1)\%N} \wedge x_{(i+2)\%N})$, tyto formule se ve výsledné formuli spojí logickou spojkou $\wedge$ a vloží do SAT solveru, ten vrátí ohodonocení ve kterém je formule splnitelná. Po přečtení všech proměnných, pokud proměnná x_i je true (resp. false) zapíše se do klíče na pozici i bit 1 (resp. 0), takto se vytvoří předchozí stav algoritmu, po provedení $(32 * 8)//2$ takovýchto kroků dostávám původní vstup a vypisuji.

4 Závěr

Obě metody jsou funkční a získají požadované řešení v rozumném čase, SAT solveru to trvá znatelně déle.