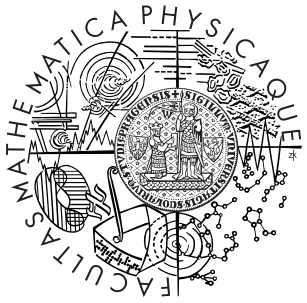
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Lukáš Fuchs

Generátory pseudonáhodných čísel a útoky na ně

Katedra algebry

Vedoucí bakalářské práce: Doc. RNDr. Jiří Tůma DrSc.

Studijní program: Matematické metody informační bezpečnosti

2007

Poďakovanie: Na tomto mieste by som rád poďakoval svojmu vedúcemu za usmerňovanie a rady pri písaní tejto práce, svojim rodičom a známym, ktorí ma podporovali a verili mi, že to nakoniec dobre dopadne a ešte všetkým mojim kamarátom a kamarátkam, ktorí ma vtedy keď som mal v pláne túto prácu písať zavolali na pivo, na ihrisko, prípadne hrať počítačové hry po sieti (a ja som nevedel na žiadnu z týchto činnosti povedať nie), pretože bezomňa by to asi nezvládli :-)

Prehlasujem, že som svoju bakalársku prácu napísal samostatne a výhradne s použitím citovaných prameňov. Súhlasím so zapožičiavaním práce a jej zverejňovaním.

V Prahe dňa Lukáš Fuchs

Obsah

Názov práce: Generátory pseudonáhodných čísel a útoky na ně
Autor: Lukáš Fuchs
Katedra (ústav): Katedra algebry
Vedúci bakalárskej práce: Doc. RNDr. Jiří Tůma DrSc.
e-mail vedúceho: tuma@karlin.mff.cuni.cz

Abstrakt: Táto práca je venovaná pseudonáhodným generátorom a to hlavne takým, ktoré sú založené na lineárnych posuvných registroch so spätnou väzbou a kryptoanalytickým útokom, ktoré sú proti nim namierené. V úvode upriamujeme pozornosť na vlastnosti, ktoré by mali postupnosti splňovať, aby boli štatistickými testami nerozlíšiteľné od náhodných. Prvá kapitola je venovaná generátorom, ktoré pozostávajú z lineárnych posuvných registrov. Na jej začiatku je ich všeobecný popis a vlastnosti, v jej hlavnej časti sa zameriavame na útoky rôzneho typu - od jednoduchého hádania počiatočného stavu po zostavovanie sústav lineárnych rovníc. Druhá kapitola popisuje algebraické útoky spolu s ich matematickým pozadím.

Kľúčové slová: pseudonáhodný generátor, lineárny posuvný register, boolovská funkcia

Title: Generators of pseudorandom numbers and attacks on them
Author: Lukáš Fuchs
Department: Department of algebra
Supervisor: Doc. RNDr. Jiří Tůma DrSc.
Supervisor's e-mail address: tuma@karlin.mff.cuni.cz

Abstract: In this thesis we present pseudorandom generators mainly based on linear feedback shift registers and cryptanalytical attacks aimed against them. Prologue is applied to features and criteria which should sequences meet to be undistinguishable from random ones. The first chapter is dedicated to generators based upon linear feedback shift registers - its general description and properties. Its main part is intent on attacks of various types - from simple guessing of initial state to building of sets of linear equations. The second chapter describes algebraic attacks along with their mathematical background.

Keywords: pseudorandom generator, linear shift register, boolean function

0.1 Úvod

V kryptografii (a nie len tam) dnes často potrebujeme náhodné postupnosti znakov (čísel, bitov...). Ideálnou šifrou je totiž taká, kde zo znalosti šifrového textu nevieme odvodiť nič o texte otvorenom a ani o kľúči. To znamená, že musíme použiť kľúč, ktorý je čo možno najnáhodnejší a neobsahuje žiadne regularity. Typickým príkladom takéhoto šifrovacieho systému je one-time pad alebo tiež Vernamova šifra. Podstatou tejto šifry je použitie náhodného kľúča, ktorý sa bitovou operáciou XOR pričíta k otvorenému textu a vznikne tak šifrový text. Kde však taký kľúč zohnať? Spôsobov ako dosiahnuť skutočne náhodnú postupnosť je málo, typicky nie sú ľahko dostupné (majú hardwarovú alebo fyzikálnu povahu) a v kryptografii ich nemôžeme použiť, pretože generátor náhodných čísel produkuje rozdielne výstupy pri rovnakých vstupoch. Tým pádom sa teda nikdy nedostaneme k pôvodnemu výstupu generátora. Keby sme niečo pomocou takého reťazca zašifrovali, nikdy by sme sa nedostali k otvorenému textu. Preto často prichádza na rad pseudonáhodnosť. Pseudonáhodné postupnosti sú výstupmi algoritmov, nemôžeme ich teda považovať za náhodné. Na druhej strane ale majú vlastnosti náhodných postupností a tým pádom sú od nich nerozlíšiteľné (takto je definovaná pseudonáhodnosť v teórii výpočtovej zložitosti). Podmienkami pre tieto algoritmy býva ľahká (polynomiálna) časová a priestorová zložitosť a už spomínaná nerozlíšiteľnosť od náhodnej distribúcie. Táto nerozlíšiteľnosť sa stanovuje pomocou štatistických testov, o ktorých sa v krátkosti zmienime. Práve vďaka týmto vlastnostiam je pseudonáhodný generátor vhodný pre tvorbu kľúčov prúdových (ale aj iných) šifier.

Pseudonáhodný generátor je zariadenie (algoritmus), ktoré z počiatočného stavu označovaného ako inicializačný vektor (v ďalšom texte ho budeme označovať IV) alebo semienko (anglicky - seed) vygeneruje nekonečnú postupnosť. V každom kroku výpočtu je na výstupe 1 alebo viacero bitov, ktoré závisia iba na aktuálnom stave. Ten sa následne nejak zmení a nasledujúca časť výstupu už závisí iba na novom aktuálnom stave. Výstupné postupnosti pseudonáhodných generátorov sú teda periodické, pretože generátor má iba konečný počet vnútorných stavov. Pri designe pseudonáhodného generátora musíme dbať na to, aby tieto periódy boli dostatočne dlhé. Útočník nesmie mať k dispozícii viacero períód výstupu generátora. Jeho úlohou býva spravidla určenie inicializačného vektora z odchyteného výstupného úseku. O možných postupoch útočníka bude táto práca pojednávať. V ďalšom texte budeme pre pseudonáhodný generátor používať skratkou PRG.

0.2 Trocha štatistiky

Na úvod si zhrnieme štatistické vlastnosti, ktoré musia spĺňať dobré pseudonáhodné postupnosti. Je dôležité, aby útočník nemohol vyčítať zo zachyteného úseku výstupu nič o bitoch, ktoré mu predchádzali alebo ho budú nasledovať. V súčastnosti existuje veľké množstvo štatistických testov alebo celých ich balíkov napr. NIST alebo DIEHARD, spomenieme si tie najzákladnejšie.

Definícia: Náhodný jav je výsledok náhodného pokusu, ktorý nastáva s istou pravdepodobnosťou. Môže pozostávať z viacerých elementárnych javov ω. Tie tvoria pravdepodobnostný priestor Ω.

Príklad: Pri hode kockou je pravdepodobnostný priestor tvorený elementárnymi javmi:

$$\Omega = \{1\} \cup \{2\} \cup \{3\} \cup \{4\} \cup \{5\} \cup \{6\}$$

Náhodným javom môže byť padnutie párneho čísla, ktorého pravdepodobnosť je 1/2.

Definícia: Náhodná veličina je zobrazenie

$$X : \Omega \to \mathcal{R}$$

ktoré javom ω z pravdepodobnostného priestoru Ω priradzuje isté reálne alebo prirodzené hodnoty. Ak sú hodnoty, ktoré náhodná veličina nadobúda ľubovoľné reálne čísla, hovoríme o spojitej náhodnej veličine, ak sú prirodzené, náhodná veličina je diskrétna.

Definícia: Rozdelenie náhodnej veličiny je pravidlo, ktoré každému náhodnému javu priradí určitú pravdepodobnosť.
Môžeme ho popísať pomocou distribučnej funkcie, ktorá každému reálnemu číslu x priradí pravdepodobnosť, že náhodná veličina nadobudne hodnotu menšiu než x.

Definícia: Distribučná funkcia spojitej náhodnej veličiny je

$$F(x) = \int_{-\infty}^{x} f(t)dt$$

funkcia $f(t)$ sa nazýva hustota pravdepodobnosti a platí $\int_{-\infty}^{\infty} f(x)dx = 1$.

Nakoniec si ešte zadefinujeme rozdelenia, s ktorými sa pri štatistických testoch stretneme:

Definícia: O náhodnej veličine X s hustotou pravdepodobnosti

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot e^{-(x-\mu)^2/2\sigma^2}$$

hovoríme, že má normálne rozdelenie $N(\mu, \sigma^2)$, kde μ a σ^2 sú jeho parametre. Ak $\mu = 0$ a $\sigma^2 = 1$, potom má X normované normálne rozdelenie.

Definícia: Náhodná veličina má rozdelenie χ^2 s k stupňami voľnosti, ak jej hustota je:

$$f(x) = \frac{1}{\Gamma\left(\frac{k}{2}\right) \cdot 2^{k/2}} \cdot e^{-x/2} \cdot x^{k/2-1}$$

pre $x > 0$, parameter $k = 1, 2, \ldots$ a funkcia Γ je definovaná nasledovne:

$$\Gamma(x) = \int_0^\infty t^{x-1} \cdot e^{-t} dt$$

Súvislosť týchto 2 rozdelení popisuje nasledujúca veta, ktorej dôkaz je obsiahly a pre túto prácu nepodstatný:

Veta: Nech $X_1 \ldots X_k$ sú nezávislé náhodné veličiny s normovaným normálnym rozdelením. Potom náhodná veličina

$$Y = \sum_{i=1}^{k} X_i^2$$

má rozdelenie χ^2 s k stupňami voľnosti.

Testovanie hypotéz a štatistické testy

Majme pseudonáhodný generátor. O postupnostiach, ktoré produkuje chceme rozhodnúť, či sa dajú rozlíšiť od náhodných. Napriek tomu, že neexistuje matematický dôkaz o tom, že by postupnosť bola náhodná, dajú sa takéto postupnosti jednoducho popísať - napríklad by mali mať približne rovnaký počet núl a jedničiek, nemali by sa v nich žiadne úseky opakovať a pod. Prvými nutnými podmienkami pre náhodnosť postupností boli Golombove postuláty, ktoré môžme nájsť napríklad v knihe [3].

Pomocou nami skúmaného generátora si teda vygenerujeme nejakú množinu postupností. Túto vzorku následne podrobíme štatistickým testom.
V týchto testoch overujeme platnosť štatistických hypotéz. To sú určité predpoklady o rozdeleniach skúmaných náhodných veličín. Vždy si stanovíme hypotézu, že daná náhodná veličina má nejaké rozdelenie a tú potom na základe našich dát prijímeme alebo odmietneme. Náhodné veličiny, ktoré budeme skúmať sú zvolené tak, aby mali normálne náhodné rozdelenie, prípadne rozdelenie χ^2. Ak nami vygenerované postupnosti prejdú viacerými testami, budú nerozlíšiteľné od náhodných a geberátor je tým pádom vhodný na kryptografické účely.

V závere si uvedieme niektoré z testov na náhodnosť, ktoré som čerpal z knihy [3] a taktiež sme ich preberali na prednáške.
Do konca tejto časti bude n označovať dĺžku skúmanej postupnosti.

Frekvenčný test

Frekvenčný test je asi najjednoduchším testom na náhodnosť bitovej postupnosti. Skúmame v ňom rozdiel medzi počtom núl a jednotiek. Dobrá pseudonáhodná postupnosť by mala mať tieto počty samozrejme vyrovnané. Označíme n_0 počet núl a n_1 počet jednotiek v testovanom úseku. Následne zistíme, či má náhodná veličina

$$X = \frac{(n_0 - n_1)^2}{n}$$

rozdelenie χ^2 s 1 stupňom voľnosti.

Sériový test

V tomto teste skúmame mohutnosti množín skupín núl a jednotiek. Môžeme si zvoliť dvoj-, troj- ale aj viacmiestne skupiny, štandardne sa však volia dvojice. Označíme si n_{ij} $i, j \in \{0, 1\}$ počet dvojíc ij v postupnosti. Tieto počty by sa pre všetky 4 možnosti dvojíc mali vyskytovať rovnomerne. Testujeme náhodnú veličinu

$$X = \frac{4}{n-1} \cdot \sum_{i,j=0}^{1} n_{ij}^2 - \frac{2}{n}(n_0^2 + n_1^2) + 1$$

na rozdelenie χ^2 s 2 stupňami voľnosti. Pri úspešnom pokuse môžeme zvýšiť mohutnosť skupín na 3 atď.

Poker test

Testovanú postupnosť si rozdelíme na neprekrývajúce sa úseky dĺžky $k \ll n$. Získame $m = \lfloor n/k \rfloor$ bitových reťazcov, z ktorých každý reprezentuje nejaké k-bitové číslo r. Pre $\forall r \in \{0..2^k-1\}$ určíme číslo n_r ako počet reťazcov v postupnosti reprezentujúcich číslo r. Očakávame, že náhodná veličina

$$X = \frac{2^k}{m} \cdot \left(\sum_{r=0}^{2^k-1} n_r^2 \right) - k$$

bude mať rozdelenie χ^2 s $2^k - 1$ stupňami voľnosti.

Runs test

Run je podpostupnosť dĺžky i, ktorá obsahuje len samé nuly, ich počet označujeme G_i(z angl. gap) alebo samé jednotky, ktorých počet je B_i (z angl. block). Očakávaný počet runov dĺžky i je $e_i = \frac{n-i+3}{2^{i+2}}$. Následne by mala náhodná veličina

$$X = \sum_{i=1}^{k} \frac{(B_i - e_i)^2}{e_i} + \sum_{i=1}^{k} \frac{(G_i - e_i)^2}{e_i}$$

mať rozdelenie χ^2 s $2k - 2$ stupňami voľnosti.

Autokorelačný test

Posledý test, o ktorom sa zmienime je zameraný na podobnosti v podpostupnostiach testovaného úseku. Konkrétne vyhľadáva podobnosti pôvodného úseku a jeho posunutých podpostupností s použitím Hammingovej vzdialenosti. Spočítame: $A_l = \sum_{i=1}^{n-l} x_i \oplus x_{i+l}$ a očakávame, že náhodná veličina

$$X = \frac{2(A_l - (n-l)/2)}{\sqrt{n-l}}$$

bude mať normálne rozdelenie N(0,1).

Kapitola 1

Lineárne posuvné registre

1.1 Popis

Táto kapitola bude venovaná lineárnym posuvným registrom so spätnou väzbou (v ďalšom texte budeme používať skratku LFSR z anglického spojenia Linear Feedback Shift Register), ktoré sa v kryptografii hojne využívajú. Sú ľahko implementovateľné, rýchle a matematicky veľmi dobre popísateľné. Ich jedinou nevýhodou, ktorá je bohužiaľ pre kryptografiu dosť podstatná je linearita, ktorú musíme pri implementácii do kryptosystému nejako odstrániť. Ich výsledné postupnosti majú však aj napriek tomu dobré štatistické vlastnosti a dlhé periódy.

LFSR je stavové zariadenie, ktoré pozostáva z posuvného registra a funkcie, ktorá býva označovaná ako feedback function. Posuvný register má l buniek, ktoré sú naplnené bitmi. V každom kroku sa tieto bity posunú o 1 doprava. Bit, ktorý sa nachádza najviac napravo je buď výstupným alebo sa skartuje a výstup určí iná (nelineárna) funkcia. Nový najľavejší bit sa spočíta pomocou feedback funkcie. U LFSR je táto funkcia jednoduchým XOR-om istej skupiny bitov registra, ktorá sa nazýva tap sequence. Tieto bity by sme mohli označiť ako riadiace a sú veľmi dôležité - od nich totiž závisí dĺžka periódy registra. Matematický popis registra dlžky l spolu s jeho riadiacimi bitmi je možný pomocou polynómu $p(x) \in Z_2[x]$, ktorý má tvar

$$p(x) = c_0 + c_1 x + c_2 x^2 + \cdots + c_l x^l$$

kde pre konštanty $c_i \in \{0, 1\}$, $i = 1 \ldots l-1$ platí $c_i = 1$ ak sa $(l-i+1)$-ty bit nachádza medzi riadicimi a $c_i = 0$ inak (c_l a c_0 sú vždy rovné 1). Tento polynóm sa v angličtine nazýva rôzne: connection, characteristic, feedback

polynomial. Budeme používať termín charakteristický polynóm. Tento polynóm je pre každý register veľmi dôležitý - ak je primitívny:

Definícia: Polynóm $p(x)$ stupňa d nad q-prvkovým telesom je primitívny, ak je irreducibilný (nedá sa faktorizovať) a najmenšie n, pre ktoré $p(x)$ delí $x^n - 1$ je $n = q^m - 1$.

potom register generuje postupnosť s maximálnou periódou, a to $2^l - 1$. Toto číslo predstavuje najvyšší počet stavov, ktorý môže register dĺžky l dosiahnuť. Z celkového množstva 2^l stavov je treba vylúčiť nulový, pretože z neho sa nemožno dostať do žiadneho iného a ani naopak - nikdy sa nedostaneme do nulového stavu z nenulového. Po prejdení všetkých stavov sa zákonite musí výstupná postupnosť registra začať opakovať a teda jej maximálna perióda je $2^l - 1$. Dôkaz tohto tvrdenia sa mi bohužiaľ nepodarilo nikde nájsť a ani vymyslieť.

S charakteristickým polynómom súvisí aj jednotlivé vyjadrenie bitov výstupu registra - ak poznáme l po sebe idúcich bitov výstupu môžeme si nasledujúci vyjadriť nasledovne:

$$z_i = c_1 z_{i-1} + c_2 z_{i-2} + \ldots + c_l z_{i-l} \bmod 2 \text{ pre } i > l$$

Tento vzťah nám môže slúžiť aj opačným smerom - z výstupných bitov môžeme zostaviť charakteristický polynóm:

Veta: *Ak poznáme $2l + 1$ po sebe idúcich bitov výstupu registra, môžeme v lineárnom čase určiť jeho charakteristický polynóm.*

Dôkaz: Keďže je z_{l+1} lineárnou kombináciou bitov $z_1, \ldots, z_l$, $2l + 1$ bitov nám dáva l lineárnych rovníc s l neznámymi koeficientami charakteristického polynómu. Príslušnú sústavu, ktorá má tvar:

$$\left(\begin{array}{ccccc|c} z_1 & z_2 & \cdots & \cdots & z_l & z_{l+1} \\ z_2 & z_3 & \cdots & \cdots & z_{l+1} & z_{l+2} \\ \vdots & \vdots & \ddots & & \vdots & \vdots \\ \vdots & \vdots & & \ddots & \vdots & \vdots \\ z_l & z_{l+1} & \cdots & \cdots & z_{2l} & z_{2l+1} \end{array}\right)$$

už poľahky vyriešime. ♡

Inou možnosťou ako určiť charakteristický polynóm je použitie Berlekamp-Masseyho algoritmu, ktorý okrem neho určí aj lineárnu zložitosť pseudonáhodnej potupnosti bitov.

Definícia: Linear complexity (lineárna zložitosť, ozn. LC) postupnosti $\{z_i\}^n$ dĺžky n je dĺžka najkratšieho LFSR, ktorý danú postupnosť vygeneruje.

Ďalej pre LC($\{z_i\}$) platí:
(i) lineárna zložitosť nulovej postupnosti je rovná 0.
(ii) ak neexistuje LFSR, ktorý by generoval $\{z_i\}$, potom je $LC(\{z_i\})=\infty$.

Berlekamp-Masseyho algoritmus je možné nájsť v knihe [3].
Niektoré vlastnosti lineárnej zložitosti popisuje nasledujúce tvrdenie:

Veta: Nech $\{z_i\}$ je binárna postupnosť. Potom
(i) Pre každé $n \geq 1$ je $0 \leq \mathrm{LC}(\{z_i\}^n) \leq n$.
(ii) $\mathrm{LC}(\{z_i\}) = 0$ práve vtedy, keď $\{z_i\}$ je nulová postupnosť.
(iii) $\mathrm{LC}(\{z_i\}) = n$ práve vtedy, keď $\{z_i\} = 0, 0, 0 \ldots, 0, 1$.
(iv) Ak má $\{z_i\}$ periódu N, potom je $\mathrm{LC}(\{z_i\}) \leq N$.

Dôkaz: Tvrdenie je zrejmé, väčšina vyplýva z definície lineárnej zložitosti.

Vďaka linearite si môžeme prechod LFSR zo stavu i do $i+1$ takisto reprezentovať pomocou matice (musíme ale poznať charakteristický polynóm).

Veta: *Majme LFSR s charakteristickým polynómom* $p(x) = 1 + c_1 x + \ldots + c_l x^l$, ktorý je v čase i naplnený hodnotami $K^i = (a_1^i, \ldots, a_l^i)$. Potom pre nasledujúci stav K^{i+1} platí:

$$(K^{i+1})^T = L \cdot (K^i)^T$$

kde matica L má tvar:

$$\begin{pmatrix} 0 & 1 & 0 & \ldots & \ldots & 0 \\ 0 & 0 & 1 & 0 & \ldots & 0 \\ \vdots & 0 & 0 & 1 & \ldots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ldots & \ldots & \ldots & 0 & 1 \\ c_l & c_{l-1} & \ldots & \ldots & c_2 & c_1 \end{pmatrix}$$

Dôkaz: Môžeme si všimnúť, že matica je takmer diagonálna, len jej diagonála je posunutá o 1 doprava - to pri násobení vektorom aktuálneho stavu odpovedá posunu jeho bitov doprava. V poslednom riadku sú riadiace bity, ktorými sa spočíta nový najľavejší, a teda posledný bit nového stavu registra. Násobenie matice teda odpovedá prechodu registra do nasledujúceho stavu. ♡

Nelinearita v registroch

Zavedením nelinearity do feedback function nám vznikne trošku zložitejší model posuvného registra, ktorého popis budeme čerpať z článku [2]. Majme LFSR dĺžky l s funkciou stupňa d, ktorá určuje jeho výstup. Potom každému stavu registra, ktorý si môžeme reprezentovať riadkovým vektorom $K^t = (k_l^t, ..., k_1^t)$ (k_i^t teda označuje $i-ty$ bit stavu registra v čase t) odpovedá stĺpcový vektor $M_d(t)$ monómov stupňa nanajvýš d, ktorý je dimenzie $D = \sum_{i=0}^{d} \binom{l}{i}$. Číslo D odpovedá počtu monómov l premených stupňa nanajvýš d (okrem monómu 0).
Napríklad pre $l = 5$ a $d = 2$ máme $D = 16$ a $K^t = (k_5^t, k_4^t, k_3^t, k_2^t, k_1^t)$.
Monómy 5 premenných do stupňa 2 sú:

$$M_d(t) \;=\; (m_1^t, m_2^t, m_3^t, \cdots, m_{15}^t, m_{16}^t)^T \;=\; (1, x_1, x_2, x_3, \cdots, x_3x_5, x_4x_5)^T$$

Ich usporiadanie nie je samozrejme pevne dané, my sa budeme pridržiavať lexikografického. Pre konkrétny stav napr. $K^t = (0, 1, 0, 1, 1)$ po stotožnení $k_i^t = x_i$ dostaneme:

$$M_d(t) = (1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1)^T$$

Ľubovoľnú boolovskú funkciu stavu registra môžeme vyjadriť ako súčin stĺpcového vektora $M_d(t)$ s riadkovým vektorom, ktorý tejto funkcii odpovedá. Nech napríklad $f(K^t) = k_2^t \oplus k_5^t \oplus k_1^t k_3^t \oplus k_3^t k_5^t$.
Túto funkciu môžeme rozpísať nasledovne:

$$f(K^t) \;=\; 0 \cdot 1 \oplus 0 \cdot k_1^t \oplus 1 \cdot k_2^t \oplus \ldots \oplus 1 \cdot k_5^t \oplus \ldots \oplus 1 \cdot k_1^t k_3^t \oplus \ldots \oplus 1 \cdot k_3^t k_5^t \oplus 0 \cdot k_4^t k_5^t$$

Jej prislúchajúcim vektorom teda je:

$$f = (0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0)$$

Výstup z aktuálneho stavu registra tým pádom spočítame ako súčin vektorov

$$f(K_t) = f \cdot M_d(t)$$

Prechod registra zo stavu t do stavu $t+1$ môžeme vyjadriť ako súčin vektora aktuálneho stavu a odpovedajúcej matice:

$$M_d(t+1) = R_d \cdot M_d(t)$$

Matica R_d sa označuje ako monomial state update matrix. Ako ale túto maticu nájdeme?

Každý bit nového stavu registra závisí na bitoch stavu predošlého. Tieto závislosti popisuje lineárna feedback function. Analogicky môžeme za pomoci tejto funkcie nájsť závislosti medzi vektormi monómov $M_d(t)$ a $M_d(t+1)$, ktoré môžeme znova popísať charakteristickými vektormi dimenzie D. Zoradením týchto vektorov do matice (vodorovne) dostaneme hľadanú maticu R_d.

Príklad: Uvážme opäť päťbitový register z predošlého príkladu s feedbackovou funkciou f:

$$k_5^{t+1} = k_4^t \oplus k_2^t \oplus k_1^t$$

kde k_5^{t+1} je nový "najľavejší" bit stavu $t+1$ a k_i^t sú bity stavu t. Ostatné bity stavu $t+1$ sú klasicky určené rovnicami $k_i^{t+1} = k_{i+1}^t$ kde $i \in \{1,2,3,4\}$. Tieto rovnice odpovedajú posunu registra o 1 miesto doprava. Pre monómy z $M_d(t+1) = (m_0^{t+1}, m_1^{t+1}, ..., m_1 0^{t+1})$ platí:

$m_0^{t+1} = 1 = (1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0) \cdot M_d(t)$
$m_1^{t+1} = x_1^{t+1} = x_2^t = (0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0) \cdot M_d(t)$
$\vdots$
$m_6^{t+1} = x_5^{t+1} = x_1^t \oplus x_2^t \oplus x_4^t = (0,1,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0) \cdot M_d(t)$
$m_7^{t+1} = x_1^{t+1} x_2^{t+1} = x_2^t x_3^t = (0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0) \cdot M_d(t)$
$\vdots$
$m_{10}^{t+1} = x_1^{t+1} x_5^{t+1} = x_2^t \cdot (x_1^t \oplus x_2^t \oplus x_4^t) = x_1^t x_2^t \oplus x_2^t \oplus x_2^t x_4^t =$
$(0,0,1,0,0,0,1,0,0,0,0,1,0,0,0,0) \cdot M_d(t)$
$\vdots$

Vektory, ktorými násobíme $M_d(t)$ sú spomínanými charakteristickými vektormi závislostí medzi $M_d(t)$ a $M_d(t+1)$ a tvoria maticu R_d, ktorá závisí iba na registri a stupni d - to znamená, že pre každý LFSR a stupeň existuje

matica s vlastnosťami matice R_d. Navyše môžme v každom kroku t registra vyjadriť jeho aktuálny vektor monómov pomocou matice a inicializačného stavu:

$$M_d(t) = R_d^t \cdot M_d(0)$$

$M_d(0)$ odpovedá vektoru monómov v čase inicializácie ($t = 0$).
Dosadením tejto rovnosti do vzťahu pre $f(K_t)$ dostávame:

$$f(K_t) = f \cdot M_d(t) = (f \cdot R_d^t) \cdot M_d = f(t) \cdot M_d$$

kde $f(t) := f \cdot R_d^t$ závisí iba na funkcii f, matici R_d a čase t (všetky tieto údaje má útočník zväčša k dispozícii).

Ďalšími možnosťami zavedenia nelinearity do generátora, ktoré už ale nebudeme tak podrobne popisovať sú:
Nelineárna kombinácia - Použijeme viacero LFSR (ideálne s nesúdeiteľnými dĺžkami) a ich výstupné bity dáme opäť na vstup nelineárnej funkcii, ktorá z nich generuje výstup. Príkladom takéhoto PRG je Geffeho generátor.
Nepravidelné krokovanie - Inou možnosťou je využtie špeciálnej funkcie, ktorá kontroluje pohyb registra. Ten je závislý na jej výstupe, prípadne na aktuálnom stave registra. LFSR sa teda môže posunúť o 1 ale aj 2 kroky, prípadne môže zostať stáť. Takéto registre sa označujú ako clock-controlled, my si predstavíme 1-2-clocked genarátor.

Geffeho generátor

Na záver sekcie si uvedieme 2 konkrétne príklady pseudonáhodných generátorov, ktoré sú založené na lineárnych posuvných registroch. Geffeho generátor pozostáva z 3 LFSR, ktoré si označíme A, B, C. Dĺžky príslušných registrov budú l_A, l_B, l_C. V každom kroku sa pohnú všetky 3 registre a prúd kľúča $\{z_i\}$ sa generuje pomocou nasledujúcej boolovskej funkcie:

$$z_i = (c_i \wedge a_i) \vee (\neg c_i \wedge b_i) \quad = c_i \cdot a_i \oplus c_i \cdot b_i \oplus b_i$$

To znamená, že výstupom generátora je bit registra A práve vtedy, keď na výstupe registra C je 1 a bit registra B, ak na výstupe C je 0. Výstupný bit druhého z registrov nemá v danom okamihu žiaden význam.

{1,2}-clocked generátor

Tento generátor je typickým príkladom clock-controlled generátora. Obsahuje 2 registre, ktoré sa označujú A a C. Jeho výstupom môže byť iba bit z

registra A v závislosti na registri C. Ten označujeme ako kontrolný register. Ak $c_i = 0$ potom sa register A pohne o 1 krok, inak o 2 kroky. Inými slovami sa skartuje c_i bitov registra A a nasledujúci je výstupom generátora.

1.2 Útoky na generátory založené na LFSR

Pri útokoch na pseudonáhodné generátory útočník pozná design generátora a má k dispozícii istú časť výstupného prúdu kľúča. Z týchto informácií sa snaží zistiť počiatočné nastavenie - inicializačný vektor. Využívať k tomu môže rôzne techniky - od prostého hádania až po zostavovanie lineárnych, či nelineárnych rovníc. Kombinovanie viacerých techník samozrejme vždy útok zefektivňuje. Útočník pritom vždy musí dávať pozor na svoje vlastné časové a pamäťové možnosti. V tejto sekcii som čerpal z [4].

Útok hrubou silou

Popis - Útok hrubou silou (Brute Force attack) znamená tipovanie všetkých l bitov inicializačného vektora. Pre každý z týchto tipov pustíme generátor a vygenerujeme aspoň istú časť bitov výstupu, ktoré následne porovnáme s bitmi, ktoré máme k dispozícii. Ak sa tieto dve postupnosti líšia (hoci len v 1 bite), náš tip bol nesprávny a musíme skúsiť ďalší. Ak sa zhodujú, náš tip je zaradený medzi kandidátov na kľúč. Ak sme našli viacero takýchto kandidátov, spustíme generátor ďalej a opäť porovnáme s odchyteným prúdom kľúča. V praxi je takýto útok veľmi zriedkavý, pretože útočník typicky nemá možnosť prejsť všetky možnosti inicializačného vektora a pre každý spustiť generátor – pre IV dĺžky l to znamená v najhoršom prípade 2^l spustení, takže nádej na úspech má len pri slabších generátoroch s krátkym IV. Pri útokoch na väčšinu generátorov sa zväčša tipuje iba časť IV a využívajú sa konkrétne vlastnosti generátora.

Modifikácie - Najprv hádame nejakú časť bitov IV a následne spustíme generátor pre každý z našich tipov. Ďalšia časť útoku sa už odvíja od konkrétneho generátora a jeho vlastostí. Musíme tam nájsť nejaké slabiny, prípadne nejaké zákonitosti, ktoré nám umožnia o niektorých z našich tipov rozhodnúť, že sú nesprávne. Ak sa nám podarí vyradiť dostatočný počet pokusov, máme veľkú šancu na úspech. V poslednej fáze opäť tipujeme, tentokrát už zvyšné bity, a skúšame ďalej. Pri niektorých slabších generátoroch môžeme dosiahnuť realistickú časovú zložitosť cca $O(2^{40})$ krokov výpočtu. Krokom výpočtu spravidla rozumieme spustenie generátora a vygenerova-

nie rozumne dlhej postupnosti. V ďalšej časti si popíšeme takéto útoky a ukážeme ich aplikácie na Geffeho a {1,2}-clocked generátore.

Veta: *Nech pre Geffeho generátor platí* $l_A = l_B = l_C = l/3$ *a nech má útočník k dispozícii lineárne mnoho (vo vzťahu k* l*) bitov prúdu kľúča. Potom je možné generátor prelomiť v čase* $O(2^{2l/3})$ *krokov výpočtu.*

Dôkaz č.1: V každom kroku útoku si najprv tipneme počiatočné nastavenie registrov A a B. Následne môžeme zostaviť úplné výstupné postupnosti $\{a_i\}$ a $\{b_i\}$ týchto 2 registrov. Z popisu Geffeho generátora vieme, že jeho výstupným bitom je vždy buď výstup registra A alebo B. Označíme si $\{z_i\}$ bity odchyteného prúdu výstupu generátora, ktorý máme v role útočníka k dispozícii. Ak sa v postupnosti $\{z_i\}$ nájde taký index i, pre ktorý bude platiť $a_i = b_i \neq z_i$, znamená to, že náš tip bol nesprávny. Čím viac bitov $\{z_i\}$ máme, tým väčšia je šanca zamietnutia nášho tipu. Ak sme takýto bit nenašli, tipujeme následne semienko registra C a opäť porovnávame s odchytenými bitmi. V prípade, že sa nám podarí nájsť také počiatočné nastavenie registra C, že všetky bity súhlasia s výstupnými, je náš tip zaradený medzi kandidátov na kľúč a pokračujeme v tipovaní ďalej. Tipovanie registrov A a B nám zaberie čas $O(2^{l_A+l_B})$ (môže sa stať, že budeme musieť prejsť takmer všetky možnosti počiatočných stavov týchto registrov). Tento krok je spomedzi všetkých v útoku najnáročnejší a preto je aj výsledná zložitosť útoku $O(2^{l_A+l_B})$. ♡

Dôkaz č.2: Tentokrát budeme hádať semienko registra C. Počiatočné nastavenie registra C určuje kedy je na výstupe bit z registra A ($c_i = 1$) a kedy z registra B ($c_i = 0$). Keďže je dĺžka registra C $l/3$, vieme o prvých $l/3$ bitoch výstupu rozhodnúť, z ktorého registra pochádzajú a môžeme ich tam zapísať. Pritom však zapisujeme práve 1 bit - buď do registra A alebo B. Pri každom tipe semienka registra C teda máme zapísaných $l/3$ bitov IV, zvyšných $l/3$ už musíme znova tipovať. Celý útok teda prebehne v čase $O(2^{2l/3})$ krokov výpočtu. ♡

Veta: *Majme {1,2}-clocked generátor s* $l_A = l_C = l/2$*. Potom útočník potrebuje výpočetný čas* $O(2^{2l/3})$ *na jeho zlomenie.*

Dôkaz: Budeme znova hádať obsah registra C. Zo semienka registra C sa dozvieme, na ktorých miestach v prúde bitov výstupu z_i sa nachádzajú bity

registra A. Inicializačný vektor registra C by mal byť zvolený náhodne, mal by teda obsahovať približne rovnaký počet núl a jedničiek. Ak je na výstupe C 0, potom vieme bit aktuálny bit výstupu A, ak je na výstupe 1 - nepoznáme ho, ale vieme nasledujúci - v priemere teda poznáme 2 z 3 bitov semienka A, ostatné musíme tipovať. Celkovo hádame $l_C + l_A/3 = l/2 + l/6 = 2l/3$ bitov IV. Nato potrebujeme výpočetný čas $O(2^{2l/3})$. ♡

Time-Memory-Data Tradeoff

Narodeninový paradox - Útok typu Time-Memory-Data Tradeoff (TMDTO), ktorý súvisí s narodeninovým paradoxom a hľadaním kolízií sa dosť často využíva pri útokoch na rôzne kryptografické primitívy, napr. blokové šifry alebo hashovacie funkcie. O čo teda ide v narodeninovom paradoxe? Za predpokladu, že sú všetky dátumy narodenia rovnako pravdepodobné sa v skupine 23 ľudí s pravdepodobnosťou viac ako 50% nachádzajú dvaja, ktorí majú narodeniny v ten istý deň.
Prečo je to tak? Pozrime sa na to z druhej strany - aká je pravdepodobnosť, že žiadni dvaja ľudia z našej (náhodne vybranej) skupiny nemajú narodeniny v ten istý deň? Prvá osoba má narodeniy v istý deň, druhá už môže mať narodeniny v jednom zo zvyšných 364 dňoch, ďalšia v 363 dňoch atď. Narodeniny jednotlivých osôb sú nezávislé javy, ktoré nastávajú súčasne - ich pravdepodobnosti sa preto násobia. Spočítame pravdepodobnosť javu, že žiadni dvaja ľudia v skupine n ľudí nemajú narodeniny v ten istý deň:

$$\tilde{p}(n) = 1(1 - 1/365)(1 - 2/365)...(1 - n/365) = ... = \frac{365!}{365^n \cdot (365 - n)!}$$

Práve sme spočítali pravdepodobnosť javu, ktorý je opačný k tomu, ktorý sme pôvodne hľadali. Jeho pravdepodobnosť teda bude $p(n) = 1 - \tilde{p}(n)$.
Po dosadení $n = 23$ dostávame $p(23) = 0.507$. Pravdepodobnosť javu, že medzi 23 ľudmi sa nachádzajú dvaja s rovnakým dátumom narodenín je teda vyššia ako 50%. My z toho môžeme odpozorovať, že pravdepodobnosť nájdenia kolízií (nájdenie $x_i \neq x_j$ takých, že $f(x_i) = f(x_j)$) je relatívne vysoká aj keď je počet obrazov x_n, ktoré máme k dispozícii nízky. To nám dáva možnosť k nasledujúcemu útoku.

Popis útoku - TMDTO pozostáva z dvoch častí. V prvej (precomputional phase) si náhodne vyberieme spomedzi všetkých možných kľúčov n hodnôt a pre tieto si spočítame prvých l bitov výstupu (potrebujeme n spustení generátora). Príslušné dvojice (kľúč,výstup) uložíme do tabuľky (budú

dokopy zaberať $2 \cdot l \cdot n$ bitov pamäte). Táto fáza je náročná na pamäť. V druhej časti útoku (realtime phase) odchytíme $m+l$–1 bitov výstupu. Máme teda m prekrývajúcich sa úsekov dĺžky l. Každý z nich vyhľadáme v tabuľke (musíme ju teda v najhoršom prípade m-krát prejsť, čo je zase náročné na čas), pre $n = m = 2^{l/2}$ máme dosť vysokú šancu na nájdenie nejakej kolízie a teda prelomenie generátora v čase $2^{l/2+1}$ pri spotrebe $2 \cdot l \cdot 2^{l/2}$ bitov pamäte. Tento útok sa dá samozrejme rôzne upravovať vzhľadom k časovým a priestorovým možnostiam útočníka.

Využitie linearity v kombinácii s Brute Force

Linear consistency test je kombináciou útoku hrubou silou a využívania linearity v generátore. Útočník háda nejakú časť bitov IV tak, aby mohol zo vzťahov medzi zostávajúcimi bitmi, uhádnutými bitmi a prúdom kľúča vytvoriť sústavu lineárnych rovníc. Musí byť ale možné vyjadriť výstupné bity pomocou bitov semienka lineárnymi rovnicami. Ak sa mu podarí vytvoriť sústavu, ktorá je schopný vyriešiť, generátor poľahky zlomí. Ak pri jej budovaní narazí na lineárne závislé rovnosti, musí doplniť ďalšie rovnice tak, aby získal sústavu s jednoznačným riešením. Tú potom vyrieši Gaussovou elimináciou. Ak sa mu však takúto sústavu nepodarí zložiť, háda semienko a zostavuje rovnice znova.

Veta: *Ak máme k dispozícii $O(l_A + l_B)$ bitov prúdu kľúča, môžeme Geffeho generátor týmto útokom zlomiť v čase $O(2^{l_C} \cdot (l_A^3 + l_B^3))$.*

Dôkaz: Budeme opäť tipovať počiatočné nastavenie registra C (potrebujeme nanajvýš 2^{l_C} pokusov), takže pre každý bit výstupu vieme určiť, z ktorého registra pochádza. Vezmeme si teda l_A výstupných bitov registra A a každý z nich vyjadríme ako lineárnu kombináciu bitov jeho semienka. Tým získame sústavu rovníc, musíme však dať pozor na jej hodnosť, môžu sa tam nachádzať lineárne závislé rovnice, vtedy musíme pridať ďalšie, na ktoré potrebujeme nové bity výstupu. Vyriešime ju a získame bity IV pre register A (riešenie sústavy zvládneme v čase $O(L_A^3)$). Analogicky postupujeme pri registri B – čas $O(L_3^B)$. Celkovo máme teda časovú zložitosť útoku $O(2^{l_C} \cdot (l_A^3 + l_B^3))$. ♡

Veta: *Ak počet bitov kľúča, ktoré má útočník k dispozícii je $O(l)$, potom je možné prelomiť $\{1-2\}$-krokovací generátor týmto útokom v čase $O(l_A^3 \cdot 2^{l_C})$.*

Dôkaz: Tipujeme stav registra C. Potom pre každý bit výstupu z_i vie útočník určiť príslušný index j taký, že $z_i = a_j$. Keďže je každý bit a_j jednoznačne určený lineárnou kombináciou počiatočného stavu, l_A bitov výstupu nám postačí na vybudovanie sústavy rovníc. Postupne zostavujeme rovnice až dokým nedostaneme sústavu, ktorú môžeme vyriešiť. Ak takú zostavíme a vyriešime, bude náš útok úspešný a odhalíme počiatočné nastavenie oboch registrov. V inom prípade hádame obsah registra C znova. Útok teda pozostáva z hádania obsahu registra C (čas $O(2^{l_C})$) a zostavenia a vyriešenia sústavy lineárnych rovníc (v čase $O(l_A^3)$). Celkový čas útoku je teda $O(l_A^3 \cdot 2^{l_C})$. ♡

Kapitola 2

Boolovské funkcie a anihilátory

Túto poslednú časť som napísal za pomoci článkov [1], [2] a [4]. Doteraz sme pri snahe zlomiť generátor zostavovali iba lineárne rovnice a zameriavali sme sa na také, ktoré popisujú LFSR. Existujú však aj útoky, ktoré sú zamerané proti kombinačnej (filtrujúcej) funkcii. Tá z výstupov niekoľkých registrov spočíta výstupný bit generátora. Túto boolovskú funkciu

$$f : GF(2^n) \rightarrow GF(2)$$

kde GF(k) označuje k-prvkové teleso (Galois Field) a n je počet registrov, môžeme popísať niekoľkými spôsobmi:

1. Tabuľkou pravdivostných hodnôt:

$$T(f) := (f(0), ..., f(2^n - 1)) \in GF(2^n),$$

ktorá obsahuje hodnoty funkcie vo všetkých bodoch $GF(2^n)$.

2. Algebraickou normálnou formou ANF(f):

$$f(x) := \bigoplus_\alpha f_\alpha x^\alpha$$

s koeficientami $f_\alpha \in GF(2)$ a indexmi $\alpha \in GF(n)$.
Algebraická normálna forma je v podstate len inak pomenovaný binárny polynóm.

Veta: *Ku každej boolovskej funkcii* $f : GF(2^n) \rightarrow GF(2)$ *existuje polynóm* $p \in GF(2^n)[x_1, ..., x_n]$ *taký, že* $f(x) = p(x)\ \forall x \in GF(2^n)$.

Dôkaz: Najprv uvážme funkciu $h_{i_1...i_n}$, ktorá má hodnotu 1 iba v bode $(i_1, ..., i_n)$, inde je nulová. Pre každé k od 1 do n vezmeme polynóm p_k premennej x_k, ktorý definujeme nasledovne:

$$p_k(x_k) = x_k \text{ pre } i_k = 1$$

$$p_k(x_k) = 1 \oplus x_k \text{ pre } i_k = 0$$

Teraz položíme $f_{i_1...i_n}(x_1, ..., x_n) = p_1(x_1) \cdot ... \cdot p_n(x_n)$ a vidíme, že platí $f_{i_1...i_n} = h_{i_1...i_n}$.
Pre obecnú boolovskú funkciu h definujeme polynóm

$$f = \sum_{h(i_1,...,i_n)=1} f_{i_1...i_n}$$

a opäť vidíme, že $f(i_1, ..., i_n) = 1$ práve vtedy, keď $h(i_1, ..., i_n) = 1$ a teda $f = h$. ♡

Definícia: Anihilátor funkcie f je každá funkcia g, pre ktorú platí:

$$f(x) \cdot g(x) = 0 \ \vee \ (f + 1)(x) \cdot g(x) = 0 \ \ \forall x \in GF(2^n)$$

Množina anihilátorov f sa označuje *AN(f)*.

Definícia: Algebraická imunita funkcie f je:

$$AI(f) := \min\{deg(g) \mid g \in AN(f)\}$$

AI(f) určuje odolnosť funkcie voči algebraickým útokom - čím je vyššia, tým je útok náročnejší. Pri našom postupe totiž riešime nelineárne rovnice, ktoré charakterizujú vzťahy medzi bitmi vstupu a výstupu kombinačnej funkcie a ich stupeň je rovný nanjavýš AI(f). Nelineárne rovnice majú oproti lineárnym niektoré nevýhody:

1. Lineárna rovnica s l premennými obsahuje maximálne l členov. Nelineárna rovnica stupňa d ich má až

$$N_d = \sum_{k=1}^{d} \binom{l}{k} \in O(l^d)$$

V najhoršom prípade ak sa $l = d$ môže mať rovnica až 2^l členov, takže sa dobre pracuje iba s rovnicami, ktorých členy majú nízke stupne. Preto sa kryptografovia snažia rovnice čo najviac zozložitiť a ich riešenie tým pádom nie je jednoduché.

2. Riešenie nelineárnych rovníc je NP-úplny problém a dosiaľ naň neexistujú efektívne algoritmy. Naštastie to nie je ani útočníkovou úlohou, ten musí zápasiť iba s konkrétnymi rovnicami, ku ktorým sa snaží nájsť aspoň množinu kandidátov na riešenie, prípadne hľadá len čiastočné riešenie, ktoré by mu napomohlo v ďalšom postupe.

Kvôli týmto dôvodom by sa nám mohlo zdať, že sú tieto útoky príliš náročné až nerealizovateľné. Kryptoanalytici ale vyvinuli niekoľko spôsobov ako si ich uľahčiť:

Linearizácia: Útočník má k dispozícii sústavu nelineárnych rovníc, kde každá rovnica vyjadruje vzťah medzi bitom výstupu a bitmi semienka. Z tejto nelineárnej sústavy môžeme urobiť jednoducho lineárnu a to tak, že členy v nej nahradíme novými premennými. Dostaneme sústavu lineárnych rovníc, ktorá ale bude mať podstatne viac premenných (ich počet môže vzrásť až exponenciálne). Preto potrebujeme veľa bitov prúdu kľúča, aby sme mohli pridať ďalšie rovnice a tým dosiahli jednoznačné riešenie. Treba tiež ešte poznamenať, že táto metóda má isté nevýhody - nahradením zložitých členov jednoduchými premennými strácame užitočné informácie.

Znižovanie stupňov: Inou možnosťou ako zlepšiť útok je znížiť stupne algebraických rovníc, pretože od nich sa vyvíja náročnosť riešnia. Práve na znižovanie stupňov využívame spomínané anihilátory boolovskych funkcií. Ak znížime stupne rovníc, klesne nám maximálny počet monómov, ktoré môžu obsahovať, a tým aj počet premenných v linearizácii, ktorá typicky nasleduje. Takéto zlúčenie týchto dvoch techník sa nazýva XL-attack (eXtension & Linearisation). Algebraické útoky sa v poslednej dobe stali mocným nástrojom kryptoanalýzy. Im venujeme záver práce.

Algebraické a rýchle algebraické útoky

Pripomenieme, že i-ty bit výstupu generátora môžeme vyjadriť nasledovne:

$$z_i = f(L^i(K_0))$$

kde f je nelineárna funkcia určujúca výstup, L je funkcia, ktorá určuje prechod registra do nasledujúceho stavu a K_0 je inicializačný vektor.

Priebeh algebraického útoku - V prvej fáze útoku sa snažíme nájsť funkciu g nízkeho stupňa d, tak aby pre všetky bity výstupu generátora, ktoré máme k dispozícii platilo $f(x) \cdot g(x) = 0$. Ak je výstupným bitom 1 ($f(x) = 1$) potom z $f(x) \cdot g(x) = 1$ vyplýva $g(x) = 0$ a g je anihilátorom f. V druhom prípade ($f(x) = 0$) použijeme rovnosť $f(x) \cdot g(x) = h(x)$. Odtiaľ dostávame $h(x) = 0$. Využitím vlastnosti GF(2): $x^2 = x$ máme

$$h(x) \cdot f(x) = f^2(x) \cdot g(x) = f(x) \cdot g(x) = h(x)$$

odtiaľ $h(x) \cdot f(x) = h(x)$ a teda $h(x) = (f + 1)(x) \cdot h(x)$. Tým pádom je h anihilátorom funkcie $f + 1$ (a podľa definície aj funkcie f). Našou úlohou v tejto časti je teda nájsť anihilátor funkcie f. Pre takú funkciu platí, že jej stupeň je vždy maximálne $n/2$.

Veta: *Nech $f : GF(2^k) \rightarrow GF(2)$. Potom pre každú dvojicu prirodzených čísel (e, d) takú, že $d + e \geq k$ existuje funkcia $g \neq 0$ stupňa nanajvýš e taká, že $f(x) \cdot g(x) = h(x)$ je stupňa nanajvýš d.*

Dôkaz: Označíme si A množinu všetkých monómov, ktoré sú stupňa $\leq d$.

$$A = \{1, x_1, x_2, ..., x_1x_2, ...\}$$

Označíme ďalej B množinu násobkov f a všetkých monómov stupňa $\leq e$.

$$B = \{f(x), x_1 \cdot f(x), ..., x_1x_2 \cdot f(x), ...\}.$$

Keďže si ľubovoľnú boolovskú funkciu môžeme vyjadriť ako polynóm, množiny A aj B teda obsahujú polynómy zo $Z_2[x_1, \ldots, x_k]$.
Ich veľkosti sú $|A| = \sum_{i=0}^{d} \binom{k}{i}$ a $|B| = \sum_{i=0}^{e} \binom{k}{i}$. Ďalej platí, že množiny boolovských polynómov k premenných môžu mať nanajvýš 2^k prvkov. Nech $C = A \cup B$, keďže podľa predpokladu máme $d + e \geq k$ potom

$$|C| = |A| + |B| = \sum_{i=0}^{d} \binom{k}{i} + \sum_{i=0}^{e} \binom{k}{i} = \sum_{i=0}^{d} \binom{k}{i} + \sum_{i=0}^{e} \binom{k}{k-i} > 2^k$$

Mnnožina C ale obsahuje monómy k premenných, a preto jej veľkosť nemôže presiahnuť 2^k. Preto sa medzi hodnotami v množinách A a B musia nachádzať

lineárne závislosti, ktoré nám umožňujú funkcie odpovedajúcich stupňov e a d nájsť. ♡

Nájsť však anihilátor funkcie f, ktorá je stupňa n, nie je jednoduché - v článku [1] hovoria o zložitosti $O(D^2)$ kde $D = \sum_{i=0}^{d} \binom{n}{i}$. Taktiež je tam uvedený aj algoritmus, ktorý využíva interpoláciu polynómov viacerých premenných. Aby bolo vôbec možné nájsť anihilátor f, musí mať táto funkcia nízky stupeň.
Ak sa nám to podarí, môžeme pristúpiť k druhému kroku, ktorým nie je nič iné ako vyriešenie sústavy rovníc $g(L^i(K_0)) = 0$ pre rôzne indexy i, v závislosti na odchytenom prúde výstupu. Je to sústava nelineárnych rovníc stupňa nanajvýš d, ktorú vyriešime linearizáciou v čase D_l^3, kde $D_l = \sum_{i=0}^{d} \binom{l}{i}$. Nato potrebujeme zhruba D_l bitov prúdu kľúča.

Vylepšenia v rýchlom algebraickom útoku - V prvom kroku nehľadáme anihilátory f, ale funkcie g ($deg(g) = e$) a h ($deg(h) = d$) také, že platí $f(x) \cdot g(x) = h(x)$ a $e < d$. Pri ich hľadaní je výhodné si f vyjadriť jej pravdivostnou tabuľkou a g a h pomocou príslušných algebraických foriem

$$g(x) = \bigoplus_{\beta} g_\beta \cdot z^\beta \quad h(x) = \bigoplus_{\gamma} h_\gamma \cdot x^\gamma$$

Následne zostavíme sústavu rovníc medzi monómami g, h a hodnotami $f(x)$:

$$f(z) \cdot \bigoplus_{\beta} g_\beta \cdot z^\beta = \bigoplus_{\gamma} h_\gamma \cdot x^\gamma$$

do ktorej za z dosadíme známe bity z výstupného prúdu. Tú následne vyriešime linearizáciou v čase $O((D + E)^3)$ kde hodnoty D a E sú klasicky $D = \sum_{i=0}^{d} \binom{n}{i}$ a $E = \sum_{i=0}^{e} \binom{n}{i}$ a predstavujú maximálny počet monómov, ktoré môžu príslušné funkcie mať a teda aj počet neznámych v sústave. Po úspešnom zostavení a vyriešení rovníc nastáva hlavná zmena oproti algebraickému útoku, a to je eliminácia členov stupňa väčšieho ako je e tak, že si vyjadríme koeficienty h_γ ako lineárne kombinácie koeficientov g_β. Týmto krokom si výrazne pomôžeme, keďže je $E < D$, pretože na záver riešime znova sústavu, tentokrát už ale s E neznámymi.

Slovo na záver

V tejto práci som sa oboznámil s popisom a vlastnosťami pseudonáhodných postupností, lineárnych posuvných registrov so spätnou väzbou, ďalej so základnými tipmi a schémami útokov, ktoré sú použiteľné proti pseudonáhodným generátorom ale aj prúdovým šifrám a iným kryptografickým primitívom.

Literatura

[1] Armknecht F. a kolektív (2006): *Efficient Computation of Algebraic Immunity for Algebraic and Fast Algebraic Attack*

[2] Hawkes P., Rose G.G. (2004): *Rewriting Variables: The Complexity of Fast Algebraic Attacks on Stream Ciphers*

[3] Venezes A., van Oorschot P., Vanstone S. (1996): *Handbook of Applied Cryptography*, CRC Press 169-201

[4] Zenner E. (2004): *Cryptoanalysis of LFSR-based Pseudorandom Generators - a Survey*

[5] http:*en.wikipedia.org*