UNIVERZA V LJUBLJANI
FAKULTETA ZA MATEMATIKO IN FIZIKO

Matematika - uporabna smer (UNI)

Matjaž Praprotnik

KRIPTOANALIZA URNO-KONTROLIRANEGA POMIČNEGA REGISTRA

Diplomsko delo

Ljubljana, 2001

Zahvaljujem se mojemu mentorju
doc. dr. Aleksandru Jurišiću
za vso pomoč pri pisanju tega dela.

Zahvaljujem se tudi mojim staršem
in bratu, ki mi stojijo ob strani.

Kazalo

PROGRAM DIPLOMSKEGA DELA

Delo naj predstavi matematične osnove, potrebne za razumevanje tokovnih šifer, ki se uporabljajo v kriptografiji (simetrični, generatorji psevdo-naključnih števil) in Linear Feedback Shift Register - LFSR (opišite zvezo med periodo in karakterističnim polinomom). Glavna cilja sta predstavitev urno-kontroliranih pomičnih šifer (angl. Clock-Controled Shift Register) in napad z omejenimi vložitvami (angl. Constrained Embedding Attack). Preučite njihove konstrukcije, varnost ter učinkovite implementacije.

Literatura:

A. Menezes, P. van Oorschot and S. Vanstone, *Handbook of Applied Cryptography*, CRC Press (Series on Discrete Mathematics and its Applications), 4th ed, 1999.

J. DJ. Golić, Constrained embedding probability for two binary strings, *Siam. J. Discr. Math.* **9**/3, (1996), 360-364.

J. DJ. Golić, L. O'Connor, Embedding and probabilistic Correlation Attacks on Clock-Controled Shift Register, *Advances in cryptology, EUROCRYPT'94*, LNCS **950** (1995), 230-243,

J. DJ. Golić, L. O'Connor, A cryptanalysis of clock-controlled shift registers with multiple steps, Cryptography: policy and algorithms (Brisbane, 1995), LNCS **1029** (1996), 174–185,

M. V. Živković, An algorithm for the Initial State Reconstruction of the Clock-Controled Shift Register, *IEEE Trans. Inform. Theory* **37**/6 (1991), 1707-1716.

Povzetek

Proučevali bomo simetrične šifre, natančneje tokovne šifre, ter njihovo varnost. Podali bomo njihovo klasifikacijo. Temu sledijo opisi osnovnih kriptografskih lastnosti toka ključev. Podrobneje sta predstavljena dva generatorja toka ključev, linearni povratni pomični register (LFSR) in urno-kontroliran pomični register (CCSR). Glavni poudarek je na napadu z vložitvami na CCSR in analizi varnosti.

Ključne besede: kriptografija, kriptoanaliza, tokovni tajnopisi, LFSR, CCSR, napad z vložitvami, vložitvena verjetnost.

Abstract

We study symmetric cryptology, in particulary stream ciphers. We clasify stream ciphers, describe basic properties of keystream generators. Detailed descriptions of linear feedback shift register (LFSR) and clock controlled shift register (CCSR) are given. Our focus are constrained embedding attack and cryptoanalysis of CCSR.

Key words: cryptology, cryptanalysis, stream ciphers, LFSR, CCSR, embedding attack, embedding probability.

Math. Subj. Class. (2000): 03D80, 11T06, 11T71, 68P25, 60C05, 11B83, 11B85

Poglavje 1

UVOD

To delo govori o simetrični kriptografiji, tokovnih šifrah, generatorjih toka ključev in varnosti le-teh. Simetrične šifre predstavljajo osnovo šifrirnim sistemom in so v uporabi že zelo dolgo, pomične šifre naj bi uporabljal že Juilius Caesar. Najbolj znan simetrični kriptosistem trenutno v uporabi je DES (angl. Data Encryption Standard), ki pa ga bo kmalu zamenjal AES (angl. Advanced Encryption Standard). Pomemben razred šifer predstavljajo tokovne šifre, ki so se pojavile s telegrafom in so prisotne in pomembne še danes. Ključni del tokovnih šifer je generator toka ključev.

Delo podrobneje obravnava posebne tipe generatorjev toka ključev, pomične registre, od najbolj osnovnega, linearnega povratnega pomičnega registra (angl. Linear Feedback Shift Register - LFSR), do bolj kompliciranih urno-kontroliranih pomičnih registrov (angl. Clock Contolled Shift Register - CCSR). Pomični registri so na področju šifriranja in kodiranja prisotni že dolgo. V praksi se tokovni sistemi uporabljajo predvsem v telekomunikacijah za šifriranje signalov. Zaradi svoje strukture so namreč zelo primerni za hardversko implementacijo, imajo pa tudi druge lastnosti, ki so pomembne v dobi informatike. Z naglim razvojem telekomunikacij je povpraševanje po hitrih in varnih kriptosistemih vse večje, zato se tokovne šifre, ki so najbolj uporabne v te namene, stalno razvijajo, prav tako pa se pojavljajo novi napadi nanje.

Naša glavna naloga je analiza varnosti sistemov s pomičnimi registri. Poudarek je na CCSR-generatorju in vprašanju kako se varnost CCSR-generatorja poveča, če povečamo število dovoljenih premikov v urno kontroliranem registru. Predstavljeni so osnovni napadi na take sisteme in njihova uspešnost.

V slovenski literaturi ni nobenega dela, ki bi se podrobneje ukvarjal s tokovnimi šiframi, razen redkih izjem o LFSR-generatorjih. Zato so uvodna poglavja daljša. Tako bo vsak bralec, z osnovnim matematičnim znanjem dobil občutek, za tokovne šifre.

Delo je razdeljeno na osem poglavij. Drugo poglavje predstavi koncept simetrične kriptografije, pojem varnosti, računske in brezpogojne, tokovne šifre in njihovo klasifikacijo ter primerjavo z bločnimi šiframi. Tretje poglavje je posvečeno generatorjem toka ključev in njihovim lastnostim, ki so pomembne v kriptografiji. Podrobneje opiše linearno in sferno zahtevnost zaporedij. Tu je vključen tudi razdelek, ki govori o končnih deterministčnih avtomatih, saj so le-ti pomembno orodje v simetrični kriptografiji. Četrto poglavje podrobno opiše linearni povratni pomični register (LFSR), saj je le-ta osnovni gradnik v večini generatorjev toka ključev, ki temeljijo na pomičnih registrih. Peto poglavje opiše poseben tip generatorjev toka ključev, urno kontroliran pomični register (CCSR). V njem je podana definicija takega generatorja, njegove lastnosti (linearna zahtevnost, perioda), ter primeri. Šesto poglavje opisuje napade na CCSR, natančneje napad z vložitvami, ki je uporaben za vse tipe CCSR-generatorjev. Sedmo poglavje je kriptoanaliza CCSR-generatorjev. Glavni poudarek tega poglavja je računanje vložitvene verjetnosti, kar je težak problem in danes v splošnem še ni rešen. Zato si pomagamo z oceno zgornje

meje le-te verjetnosti, pri čemer uporabimo teorijo končnih avtomatov. Na koncu v zaključku predstavimo še odprte probleme na področju tokovnih šifer in nakažemo možnosti razvoja tega področja v prihodnje.

Poglavje 2

SIMETRIČNA KRIPTOGRAFIJA

V tem poglavju bomo najprej predstavili princip simetrične kriptografije, nato pa vpeljali še njeno alternativo, asimetrično kriptografijo. Nadaljevali bomo z osnovnimi simetričnimi kriptosistemi, kjer bomo najprej opisali bločne šifre, nato pa podrobneje tokovne šifre. Poglavje zaključimo s primerjavo med bločnimi in tokovnimi šiframi.

Simetrični kriptosistemi predstavljajo osnovo kriptografije. Vključeni so v večino aplikacij, ki uporabljajo šifriranje. Čeprav je zadnje čase veliko govora o javni kriptografiji (asimetrična kriptografija), se le-ta uporablja predvsem za distribucijo ključev. Za ostalo pa v večini primerov uporabljamo simetrične šifre, tako da predstavljajo levji delež šifrirnih sistemov.

Definicija 2.1. Kriptosistem *je peterka* $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$*, ki zadošča naslednjim pogojem.*

1. $\mathcal{P}$ *je končna množica možnih čistopisov (angl. plaintext).*
2. $\mathcal{C}$ *je končna množica možnih tajnopisov (angl. ciphertext).*
3. $\mathcal{K}$, **prostor ključev**, *je končna množica možnih ključev.*
4. *Za vsak* $K \in \mathcal{K}$ *obstajata šifrirno pravilo* $e_K \in \mathcal{E}$ *in ustrezno dešifrirno pravilo* $d_K \in \mathcal{D}$*, kjer sta* $e_K : \mathcal{P} \to \mathcal{C}$ *in* $d_K : \mathcal{C} \to \mathcal{P}$ *taki funkciji, da je* $d_K(e_K(x)) = x$*, za vsak čistopis* $x \in \mathcal{P}$*.*

Predpostavimo, da je pošiljatelj šifriranega sporočila Ana, prejemnik pa Bob. Ana in Bob bosta sprejela naslednji protokol za uporabo specifičnega kriptosistema. Najprej bosta izbrala naključen ključ $K \in \mathcal{K}$. To storita na samem, kjer možni napadalec ne opazuje, ali preko varnega kanala. Kasneje, Ana želi imeti možnost poslati sporočilo tudi preko ne-varnega kanala Bobu. Naj bo sporočilo niz

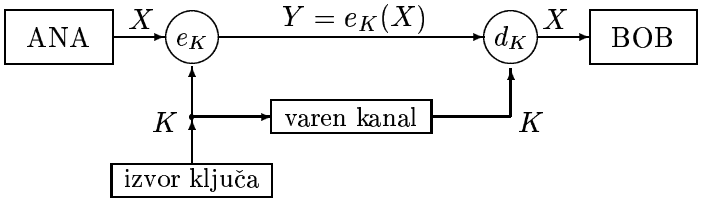
$$\boldsymbol{x} = x_1 x_2 \dots x_n,$$

za neko celo število $n \geq 1$, kjer je vsak simbol čistopisa $x_i \in \mathcal{P}$, za $1 \leq i \leq n$. Vsak x_i je zašifriran s šifrirnim pravilom e_K, ki je določeno z vnaprej izračunanim ključem K. Ana torej izračuna $y_i = e_K(x_i),\ 1 \leq i \leq n$. Rezultat, niz tajnopisa

$$\boldsymbol{y} = y_1 x_2 \dots y_n,$$

je nato poslan preko kanala Bobu. Ko Bob dobi niz $y_1 y_2 \dots y_n$, ga dešifrira z dešifrirno funkcijo d_K, ter tako dobi niz čistopisa $x_1 x_2 \dots x_n$. Komunikacijski kanal in ilustracija protokola sta razvidna iz slike 2.1.

Pri zgoraj opisanem protokolu smo upoštevali lastnost 4 v definiciji 2.1, poleg tega pa mora biti šifrirna funkcija e_K injektivna, saj drugače dešifriranje ni mogoče. Če je recimo $y = e_K(x_1) = e_K(x_2)$, za $x_1 \neq x_2$, potem Bob ne more vedeti, ali se y odšifrira v x_1 ali v x_2.



Slika 2.1: Shema simetričnega kriptosistema.

Bistvo simetričnega kriptosistema je torej v tem, da se isti ključ $K \in \mathcal{K}$ uporabi v šifrirni in dešifrirni funkciji. Druga imena za take kriptosisteme sta še *one-key kriptosistem* ali *privat-key kriptosistem*. Simetričnih kriptosistemov je veliko, mednje štejemo tudi bločne in tokovne šifre. Najbolj znan in uporabljen simetrični kriptositem je DES (angl. Data Encryption Standard), glej [29, str.72-113]. National Institute of Standards and Technology (NIPS) je v letu 2000 za nov Federal Information Processing Standard (FIPS) izbral bločni kriptosistem imenovan tudi AES (angl. Advanced Encryption Standard), ki bo pri ameriški vladi nadomestil DES, glej [32].

Slaba stran takega kriptosistema je vsekakor potreba po varnem kanalu za izmenjavo ključa. Poleg tega je zaradi dejstva, da se isti ključ uporabi tako pri šifrirnem, kot pri dešifrirnem postopku, e_K ponavadi enak d_K, oziroma se iz njega da hitro izračunati.

Alternativa simetričnemu kriptosistemu, ki odpravi potrebo po varnem kanalu, so kriptosistemi, ki temeljijo na Diffie-Helmmanovem principu *javne kriptografije*, objavljenim leta 1976. Takim kriptosistemom pravimo tudi *asimetrični kriptosistemi*. Ideja, ki se skriva tu je, da je mogoče konstruirati kriptosistem, kjer je težko izračunati dešifrirno pravilo d iz šifrirnega pravila e. Tako je e lahko javen in znan vsem, d pa pozna samo prejemnik šifriranega sporočila. Prednost javnih kriptosistemov je, da Ana lahko pošlje Bobu šifrirano sporočilo, ne da bi se z njim prej dogovorila za ključ, z uporabo javne kriptografske funkcije e. Bob bo tako edina oseba, ki bo lahko dešifriral sporočilo z uporabo svoje tajne dešifrirne funkcije d. Javnih kriptosistemov je več. Najbolj znan in danes uporabljen je RSA, glej [29, str.116-161], ki temelji na problemu razcepa števila n na prafaktorje. Poleg RSA se uporabljajo še sistemi, ki temeljijo na problemu diskretnega logaritma, najbolj uporaben je kriptosistem na eliptičnih krivuljah, glej [29, str.177-190] in [20].

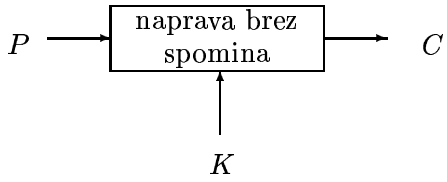
Slabost asimetričnih kriptosistemov je v njihovi počasnosti, zato so simetrični sistemi, ki so občutno hitrejši, zelo aktualni.

2.1 Bločne šifre

Kot smo že omenili, so primeri simetričnih kriptosistemov tudi *bločne šifre*. Te šifre so pomembne v kriptografiji in so osnovni gradniki v konstrukciji raznih generatorjev psevdo-naključnih števil, zgoščevalnih funkcij, tokovnih šifer, prav tako pa se uporabljajo pri digitalnih podpisih, identifikacijskih protokolih,... V tem razdelku bomo definirali bločne šifre, prikazali osnovne lastnosti in navedli najbolj značilne primere.

Definicija 2.2. Bločna šifra *je funkcija $E : U_n \times \mathcal{K} \to V_n$, ki preslika n-bitni blok $P = (p_1, p_2, \ldots, p_n)$ čistopisa $P \in \mathcal{P}$ v n-bitni blok $C = (c_1, c_2, \ldots, c_n)$ tajnopisa $C \in \mathcal{C}$, taka, da je za vsak ključ $K \in \mathcal{K}$ šifrirna funkcija $e_K(P) = e(K, P)$ obrnljiva preslikava, katere inverz je dešifrirna funkcija $d_K(C)$, in velja $d_K(e_K(P)) = P$ za vsak ključ K in n-bitni blok $P \in U_n$. Množica U_n se sestoji iz n-bitnih blokov čistopisa, množica V_n pa iz n-bitnih blokov tajnopisa.*

Bločna šifra razbije čistopis na zaporedne bloke dolžine n in vsak blok $P = (p_1, p_2, \ldots, p_n)$ sporočila s ključem K zašifrira v pripadajoč blok $C = (c_1, c_2, \ldots, c_n)$. V večini primerov je $\mathcal{P} = \mathcal{C}$. Za šifriranje zaporednih blokov se uporablja ista funkcija, torej so bločne šifre brez spomina (slika 2.2).



Slika 2.2: Shema bločne šifre.

Bločnih šifer je več vrst. Kot primer si bomo ogledali dve najbolj enostavni.

1. **Substitucijska šifra**
   Najenostavnejša je substitucijska šifra, kjer je dolžina bloka $n = 1$. Ključ $K \in \mathbb{Z}_{25}$ pa je fiksirano število oziroma črka abecede. Šifrirna funkcija $e_K(P)$, kjer $P \in \mathbb{Z}_{25}$, je seštevanje po modulu 25, dešifrirna funkcija $d_K(C)$, kjer $C \in Z_{25}$, pa odštevanje po modulu 25. Izbrani črki pravimo tudi *ključna črka*.

$$\mathcal{P} = \mathcal{C} = \mathbb{Z}_{25},\ K \in \mathbb{Z}_{25}$$

$$e_K(P) \equiv P + K \pmod{25},\quad d_K(C) \equiv C - K \pmod{25}.$$

   Za ilustracijo take šifre si oglejmo naslednji primer. Izberimo si naključno črko abecede, oziroma celo število med 0 in 24. Recimo, da smo si izbrali črko M. Ustrezno število je potem 13. Z njim želimo zašifrirati čistopis
   MOJEIMEJEMATJAŽ
   Najprej pretvorimo zgornji tekst v ustrezna števila med 0 in 24 (A=0, Ž=24), ki jim nato po modulu 25 prištejemo naš ključ $K = 13$. Dobljenim številom nato priredimo ustrezne črke.

13	15	10	5	9	13	5	10	5	13	0	20	10	0	24
13	13	13	13	13	13	13	13	13	13	13	13	13	13	13
1	3	23	18	22	1	18	23	18	1	13	7	23	13	12

(2.1)

   Tajnopis bo torej naslednji:
   BČZSVBSZSBMGZML.

2. **Vigenerjeva šifra**
   Pri Vigenerjevi šifri si za ključ K izberemo besedo dolžine m, kjer je m dolžina bloka. Izbrani besedi pravimo tudi *ključna beseda*.

$$\mathcal{P} = \mathcal{C} = (\mathbb{Z}_{25})^m,\ K \in (\mathbb{Z}_{25})^m,\quad K = (k_1, k_2, \ldots, k_m),$$

$$e_K(p_1, p_2, \ldots, p_m) = (p_1 + k_1 \pmod{25}, \ldots, p_m + k_m \pmod{25}),$$

$$d_K(c_1, c_2, \ldots, c_m) = (c_1 - k_1 \pmod{25}, \ldots, c_m + k_m \pmod{25}).$$

   Za ilustracijo Vigenerjeve šifre si oglejmo zgornji primer, kjer bomo za ključ K vzeli besedo IME. Zopet, kot v prvem primeru, črke čistopisa pretvorimo v števila med 0 in 24 (A=0,

Ž=24), nato pa šifriramo s funkcijo opisano zgoraj. Ključ je $K = (9, 13, 5)$.

13	15	10	5	9	13	5	10	5	13	0	20	10	0	24
9	13	5	9	13	5	9	13	5	9	13	5	9	13	5
22	3	15	14	22	18	14	23	10	22	13	0	19	13	4

(2.2)

Tajnopis bo torej naslednji:
VČONVSNZJVMAŠMD.

2.2 Tokovne šifre

Tokovne šifre (angl. Stream Ciphers) tvorijo pomemben razred šifrirnih algoritmov. Za razliko od bločnih šifer, ki simultano šifrirajo skupino znakov oziroma blok čistopisa s fiksno šifrirno transformacijo, tokovne šifre šifrirajo vsak bit čistopisa posebej, s šifrirno transformacijo, ki se s časom spreminja. Tokovne šifre so ponavadi hitrejše od bločnih in tudi bolj primerne, v nekaterih primerih celo nujne, npr. kadar je vmesni pomnilnik (angl. buffering) omejen ali kadar moramo vsak znak obdelati posebej. Pogosto se uporabljajo tudi v situacijah, kjer je verjetnost napake med prenosom velika.

Teoretično znanje o tokovnih šifrah je danes zelo obsežno in predlaganih je bilo že mnogo različnih principov in sistemov. Kljub temu pa je relativno malo sistemov s popolno specifikacijo in analizo, ki so dostopni v literaturi. Razloge za to lahko najdemo tudi v dejstvu, da večina v praksi uporabljenih tokovnih šifer teži k tajnosti. Po drugi strani pa je bilo veliko bločnih šifer javno objavljenih in sprejetih v standarde. Kljub temu, pa so zaradi že naštetih prednosti tokovne šifre danes v široki uporabi in v prihodnje lahko pričakujemo še več predlogov novih sistemov.

V nadaljevanju bomo najprej definirali tokovne šifre. Kot primer si bomo podrobneje ogledali Vernamovo šifro, znano tudi kot *enkratni ščit* (angl. one-time-pad), pri katerem bomo za analizo varnosti uporabili Shannonovo teorijo. Temu sledi klasifikacija tokovnih šifer in za konec nekaj osnovnih primerov.

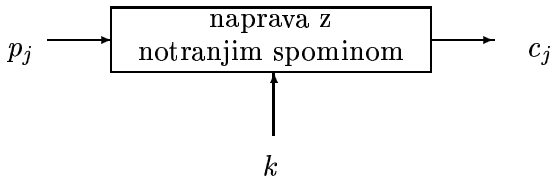
Osnovne lastnosti in definicije

Definicija 2.3. Tokovna šifra *je sedmerica* $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{L}, \mathcal{F}, \mathcal{E}, \mathcal{D})$, *ki zadošča naslednjim pogojem:*

1. $\mathcal{P}$ *je končna množica možnih čistopisov.*
2. $\mathcal{C}$ *je končna množica možnih tajnopisov.*
3. $\mathcal{K}$, *prostor ključev, je končna množica možnih ključev.*
4. $\mathcal{L}$ *je končna množica tokovne abecede.*
5. $\mathcal{F} = (f_1, f_2, \ldots)$ *je* **generator toka ključev***:*

$$f_i : \mathcal{K} \times \mathcal{P}^{i-1} \to \mathcal{L}, \quad i \geq 1.$$

6. *Za vsak ključ* $z \in \mathcal{L}$ *imamo šifrirni postopek* $e_z \in \mathcal{E}$ *in dešifrirni postopek* $d_z \in \mathcal{D}$, *tako da je* $d_z(e_z(p)) = x$ *za vsak* $p \in \mathcal{P}$.



Slika 2.3: Shema tokovne šifre.

Tokovna šifra je torej določena z napravo z notranjim spominom, ki j-ti člen čistopisa p_j zašifrira v j-ti člen tajnopisa c_j s ključem z_j in funkcijo, ki je odvisna od ključa in notranjega stanja tokovne šifre v koraku j. Z drugimi besedami, šifrirna funkcija se s časom spreminja (slika 2.3).

$$c_j = e_{z_j}(p_j), \quad z_j = f(k, p_{j-1}), \quad z_j = f(k, c_{j-1}).$$

Zaporedju $z^\infty = z_0 z_1 \ldots$ pravimo *tok ključev*, deterministočnemu avtomatu, ki proizvaja tok ključev pa *generator toka ključev*. Za šifriranje čistopisa $p_1 p_2 \ldots$ zaporedno računamo

$$z_1, c_1, z_2, c_2, \ldots,$$

za dešifriranje tajnopisa $c_1 c_2 \ldots$ pa

$$z_1, p_1, z_2, p_2, \ldots.$$

Definicija 2.4. *Tokovna šifra je* **periodična s periodo d**, *če se tok ključev ponovi po d korakih, to je, kadar velja $z_{i+d} = z_i$ za vsak $i \geq 1$.*

Če si še enkrat ogledamo Vigenerjevo šifro, ki smo jo definirali pri bločnih šifrah, potem lahko nanjo gledamo tudi kot na tokovno šifro s periodo m, kjer je m dolžina izbrane ključne besede.

Primer. Naj bodo $(k_1, k_2, \ldots k_m)$ ključi. Za vsak $i = 1, \ldots, m$ naj bo $z_i = k_i$. Zaporedje definirajmo z naslednjo linearno rekurzijo stopnje m:

$$z_{i+m} = z_i + \sum_{j=1}^{m-1} c_j z_{i+j} \pmod 2,$$

kjer so $c_1, c_2, \ldots, c_{m-1} \in \mathbb{Z}_2$ vnaprej določene konstante. Zgornja meja za periodo takšnega zaporedja je $2^m - 1$. S primerno izbiro vektorja $(k_1, \ldots, k_m)$ lahko dobimo tokovno šifro, katerega perioda doseže to zgornjo mejo. •

Tokovne šifre štejemo k simetrični kriptografiji, kar pa ne pomeni, da ne obstajajo tokovne šifre, ki so asimetrične. Primer takega je recimo Blum-Goldwasser-jeva verjetnostna šifra z javnimi ključi (angl. probabilistic public-key encryption), glej [29, str.379-382].

Shannonova teorija

Obstajata dva osnovna pristopa k varnosti kriptosistemov, *računska* in *brezpogojna* varnost.

Definicija 2.5. *Kriptosistem je* **računsko varen**, *če najboljši algoritem za razbijanje le-tega potrebuje vsaj N operacij, kjer je N določeno, zelo veliko število. Drugače povedano, kriptosistem je "računsko varen", če najboljši algoritem za razbijanje le-tega potrebuje veliko računskega časa, oziroma če se da problem varnosti prevesti na kakšen drug, bolj znan težak problem, kot so na primer faktorizacija celih števil ali diskretni logaritem.*

Tak pristop zagotavlja relativno varnost glede na drug problem in ne absolutne varnosti kriptosistema.

Definicija 2.6. *Kriptosistem je* **brezpogojno varen**, *če se ga ne da razbiti niti z neomejeno računsko močjo.*

Brezpogojne varnosti seveda ne študiramo s stališča računske zahtevnosti, ker privzamemo, da je računska moč neomejena. Zato se zatečemo k pomembnemu orodju, verjetnosti. Osnovni izreki verjetnosti, ki jih bomo tu uporabili, so na voljo v [17]. Poleg pojma brezpogojne varnosti vpeljemo še koncept *popolne varnosti.* Neformalno popolna varnost pomeni, da napadalec ne more dobiti nobene informacije o čistopisu, če ima na voljo samo tajnopis. Za natančno definicijo pa uporabimo verjetnost.

Definicija 2.7. *Naj bo $P_{\mathcal{P}}(x \mid y)$ verjetnost, da je x čistopis, ki ustreza danemu tajnopisu y, $P_{\mathcal{P}}(x)$ pa verjetnost, da se čistopis x pojavi v sporočilu. Kriptosistem ima* **popolno varnost**, *če je $P_{\mathcal{P}}(x \mid y) = P_{\mathcal{P}}(x)$ za vsak $x \in \mathcal{P}$, $y \in \mathcal{C}$. Drugače povedano, kriptosistem je popolnoma varen, če je verjetnost, da čistopis x ustreza danemu tajnopisu y, enaka verjetnosti pojavitve čistopisa x v sporočilu.*

Naslednji izrek nam da zadosten pogoj za popolno varnost kriptosistema.

Izrek 2.8. *Naj bo $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ kriptosistem, kjer $|\mathcal{K}| = |\mathcal{C}| = |\mathcal{P}|$. Potem je kriptosistem popolnoma varen natanko tedaj, ko je vsak ključ uporabljen z enako verjetnostjo $1/|\mathcal{K}|$ in za vsak $x \in \mathcal{P}$ in $y \in \mathcal{C}$ obstaja enoličen ključ K, tako da je $e_K(x) = y$.*

Dokaz. Predpostavimo, da je dani kriptosistem popolnom varen. Za vsak $x \in \mathcal{P}$ in $y \in \mathcal{C}$ mora potem biti vsaj en ključ K, tako da je $e_K(x) = y$. Imamo torej naslednjo neenakost:

$$|\mathcal{C}| = |\{e_K(x) : K \in \mathcal{K}\}| \ \leq \ |\mathcal{K}|. \tag{2.3}$$

Ker pa smo predpostavili, da je $|\mathcal{K}| = |\mathcal{C}|$, mora v neenakosti (2.3) veljati enačaj, iz česar sledi

$$|\{e_K(x) : K \in \mathcal{K}\}| = |\mathcal{K}|. \tag{2.4}$$

Torej ne obstajata dva različna ključa K_1 in K_2, tako da bi veljalo $e_{K_1}(x) = e_{K_2}(x) = y$. Pokazali smo torej, da za vsak $x \in \mathcal{P}$ in $y \in \mathcal{C}$ obstaja natanko en ključ, tako da je $e_K(x) = y$.

Naj bo $n = |\mathcal{K}|$. Naj bo $\mathcal{P} = \{x_i : 1 \leq i \leq n\}$. Fiksiramo $y \in \mathcal{C}$. Naj bodo ključi $K_1, \ldots, K_n$ izbrani tako, da velja $e_{K_i}(x_i) = y_i, \ 1 \leq i \leq n$. Z uporabo Bayes-ovega izreka [17, str.22] dobimo naslednjo enakost:

$$P_p(x_i \mid y) = \frac{P_{\mathcal{C}}(y \mid x_i) P_p(x_i)}{P_{\mathcal{C}}(y)} = \frac{P_{\mathcal{K}}(K_i) P_p(x_i)}{P_{\mathcal{C}}(y)}, \tag{2.5}$$

kjer je $P_{\mathcal{C}}(y \mid x)$ verjetnost, da dan tajnopis y ustreza čistopisu x, $P_{\mathcal{C}}(y)$ verjetnost, da se tajnopis y pojavi v sporočilu, $P_{\mathcal{K}}(K)$ verjetnost, da je bil uporabljen ključ K. Iz definicije popolne varnosti sledi $P_{\mathcal{P}}(x_i \mid y) = P_{\mathcal{P}}(x_i)$. Če to vstavimo v enakost (2.5), dobimo $P_{\mathcal{K}}(K_i) = P_{\mathcal{C}}(y)$, za $1 \leq i \leq n$. To nam pove, da so ključi uporabljeni z enako verjetnostjo ($P_{\mathcal{C}}(y)$). Ker pa je število ključev enako $|\mathcal{K}|$, sledi da je $P_{\mathcal{K}}(K) = 1/|\mathcal{K}|$ za vsak $K \in \mathcal{K}$.

Obratno, če sta izpolnjeni obe zahtevi, to je zahteva po enoličnem ključu in enaki verjetnosti za vse ključe, potem lahko hitro pokažemo popolno varnost kriptosistema. ■

Definicija 2.9. Vernamova šifra *nad binarno abecedo je definirana s pravilom:*

$$c_i = m_i \oplus k_i, \quad i = 1, 2, 3, \ldots,$$

kjer so $m_1, m_2, m_3, \ldots$ elementi čistopisa, $k_1, k_2, k_3 \ldots$ je tok ključa, $c_1, c_2, c_3, \ldots$ elementi tajnopisa in $\oplus$ bitni XOR (seštevanje po bitih $\pmod 2$). Dešifrirni postopek je definiran z

$$m_i = c_i \oplus k_i, \quad i = 1, 2, 3 \ldots.$$

Definicija 2.10. *Vernamova šifra je* **enkratni ščit** *(angl. one-time-pad), če so členi toka ključev generirani naključno in neodvisno eden od drugega.*

Iz definicije enkratnega ščita in izreka 2.8 sledi, da je le-ta popolnoma varen. Poleg tega pa je sistem zanimiv zaradi enostavnega šifriranja in dešifriranja. Žal pa je največji problem pogoj $|\mathcal{K}| \geq |\mathcal{P}|$, ki zahteva, da je velikost tajno dogovorjenega ključa vsaj tolikšna kot velikost čistopisa. Za šifriranje n bitov čistopisa zahtevamo n bitov ključa. To sicer ne bi predstavljalo prevelikega problema, če bi isti ključ lahko uporabili večkrat, vendar pa je varnost brezpogojno varnega tajnopisa odvisna od dejstva, da vsak ključ uporabimo samo za en čistopis (od tod tudi ime enkratni ščit). Tako je ob uporabi istega ključa za več sporočil, enkratni ščit občutljiv na napade z znanim čistopisom, saj ključ K lahko izračunamo kot ekskluzivni ali bitnega niza x in $e_K(x)$. Torej moramo za popolno varnost za vsako sporočilo generirati nov ključ, kar povzroča težave pri distribuciji ključev. To je tudi razlog za omejeno uporabo enkratnega ščita v komercialne namene. Zaradi brezpogojne varnosti pa je uporaben tam, kjer je to potrebno, na primer za vojaške ali diplomatske namene.

Problemi, ki nastopijo pri brezpogojno varnih kriptosistemih, motivirajo konstrukcijo tokovnih šifer, kjer bo tok ključev generiran psevdo-naključno iz danega semena, ki postane skrivni ključ. Taki sistemi ne zagotavljajo brezpogojne varnosti, obstaja pa upanje, da so računsko varni.

Klasifikacija tokovnih šifer

Običajno tokovne šifre razdelimo v dva razreda, *sinhrone* in *asinhrone*. Sinhrone tokovne šifre so tiste, pri katerih je tok ključa generiran neodvisno od čistopisa in tajnopisa sporočila. Asinhrone tokovne šifre so tiste, pri katerih je tok ključev generiran kot funkcija ključa k in fiksnega števila členov predhodnega tajnopisa.

Sinhrone tokovne šifre

Če si ogledamo definicijo 2.3 tokovnih šifer, oziroma komentar k tej definiciji, smo tokovne šifre definirali kot naprave z notranjim spominom. Pri sinhroni tokovni šifri je generator toka ključev neodvisen od čistopisa in tajnopisa, zato za te vrste tokovnih šifer velja, da so šifrirne transformacije brez spomina, kljub temu pa se s časom spreminjajo, saj je generator toka ključev odvisen od prejšnjega stanja.

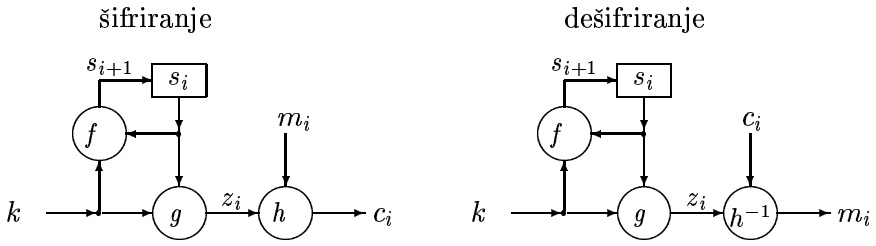
Naj bodo s_0 začetno stanje, ki ga lahko določimo iz ključa k, f funkcija naslednjega stanja, g generator toka ključev z_i, h izhodna funkcija, ki iz čistopisa m_i in ključa z_i generira tajnopis c_i. Šifrirni postopek sinhrone tokovne šifre lahko opišemo z enačbami:

$$\begin{aligned} s_{i+1} &= f(s_i, k), \\ z_i &= g(s_i, k), \\ c_i &= h(z_i, m_i). \end{aligned} \tag{2.6}$$

Postopka za šifriranje in dešifriranje sta razvidna tudi na sliki 2.4.

Lastnosti sinhronih tokovnih šifer

1. *Sinhronizacija*
   Pri sinhronih tokovnih šifrah morata biti oba, prejemnik in pošiljatelj sinhronizirana, to je, uporabljati morata isti ključ in operirati pri istem stanju znotraj tega ključa, da se tajnopis pravilno dešifrira. Če je sinhronizacija izgubljena zaradi brisanja ali vstavljanja členov tajnopisa med prenosom, potem dešifriranje odpove in ga lahko znova vzpostavimo le s



Slika 2.4: Shema sinhrone tokovne šifre.

posebnimi metodami, kot so re-inicializacija ali pa postavljanje posebnih znakov oziroma markerjev na koncu vsakega intervala z vnaprej dogovorjeno dolžino.

2. *Onemogočen prenos napak*
   Če med prenosom pride do modifikacije (ki pa ne sme biti brisanje) določenega člena tajnopisa, to ne vpliva na dešifriranje ostalih členov tajnopisa.

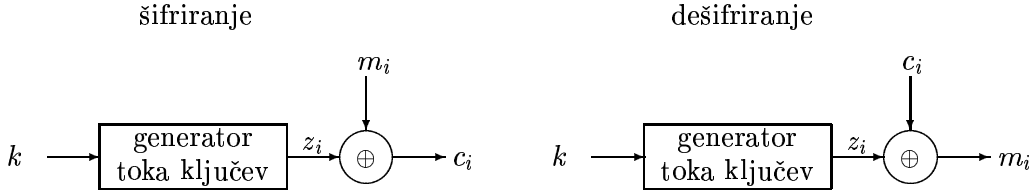
Posledica lastnosti 1, vstavljanja, brisanja ali ponovitve posameznega člena tajnopisa, povzroči takojšnjo izgubo sinhronizacije in posledično prejemnik oziroma dešifrant napad lahko odkrije. Posledica lastnosti 2 je, da s spremembo določenega člena tajnopisa med prenosom, napadalec lahko odkrije, kje se ta sprememba pozna pri čistopisu.

Aditivne sinhrone tokovne šifre

Večina predlaganih sinhronih tokovnih šifer je aditivnih.

Definicija 2.11. Aditivna sinhrona tokovna šifra *je sinhrona tokovna šifra, pri kateri je izhodna funkcija h, ki iz toka ključev z_i in čistopisa m_i generira tajnopis c_i, funkcija seštevanja. Pri tem morajo biti tokovna abeceda $\mathcal{L}$, množica čistopisev $\mathcal{P}$ in množica tajnopisov $\mathcal{C}$ Abelove grupe $(G, +)$.* **Binarna aditivna sinhrona šifra**, *je aditivna sinhrona šifra, kjer so tok ključev, čistopis in tajnopis binarna števila, izhodna funkcija h pa je ekskluzivni ali (XOR).*

Postopka za šifriranje in dešifriranje pri binarnih aditivnih sinhronih šifrah sta prikazana na sliki 2.5.



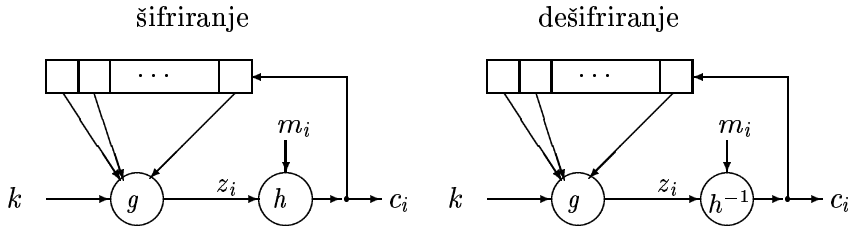
Slika 2.5: Shema binarne aditivne sinhrone tokovne šifre.

Asinhrone tokovne šifre

Naj bodo $(c_{-t}, c_{-t+1}, \ldots, c_{-1})$ začetno stanje (lahko tudi javno), k ključ, g generator toka ključev z_i, h izhodna funkcija, ki iz čistopisa m_i in toka ključev generira tajnopis c_i. Šifrirni postopek asinhrone tokovne šifre lahko opišemo z enačbami:

$$\begin{aligned} s_i &= (c_{i-t}, c_{i-t+1}, \ldots, c_{i-1}), \\ z_i &= g(s_i, k), \\ c_i &= h(z_i, m_i). \end{aligned} \tag{2.7}$$

Postopka šifriranja in dešifriranja sta razvidna tudi iz slike 2.6.



Slika 2.6: Shema asinhrone tokovne šifre.

Lastnosti asinhronih tokovnih šifer

1. *Samo-sinhronizacija*
   Če so posamezni biti tajnopisa vstavljeni ali zbrisani med prenosom, je možna samo-sinhronizacija, saj je dešifrirna funkcija odvisna samo od fiksnega števila členov predhodnega tajnopisa. Take šifre so zmožne avtomatične vzpostavitve sinhronizacije, če je bila ta izgubljena, pri čemer se bo narobe odšifriralo samo določeno število členov tajnopisa. Če na primer zbrišemo i-ti člen tajnopisa c_i, se bo narobe odšifriralo t členov tajnopisa, $c_{i+1}, \ldots, c_{i+t}$.

2. *Omejen prenos napak*
   Če med prenosom modificiramo določen člen tajnopisa, kjer zaradi samo-sinhronizacije za razliko od sinhronih dopuščamo tudi brisanje in vstavljanje, se bo narobe odšifriralo t zaporednih členov tajnopisa, do vzpostavitve sinhronizacije.

3. *Razpršenost statističnih lastnosti čistopisa*
   Ker vsak člen čistopisa vpliva na celoten tajnopis od tega člena dalje, so statistične lastnosti čistopisa razpršene čez tajnopis, kar naredi asinhrone tokovne šifre manj občutljive na napade, ki temeljijo na statističnih lastnostih čistopisa.

Zaradi lastnosti 2, modifikacija posameznega člena tajnopisa povzroči napačno dešifriranje t zaporednih členov tajnopisa. Verjetnost, da prejemnik modifikacijo opazi, je v primerjavi s sinhronimi tokovnimi šiframi večja, saj se odšifrira napačno več členov. Zaradi lastnosti 1 je težje opaziti vstavljanje, brisanje ali ponovitev posameznih bitov.

2.3 Primerjava med bločnimi in tokovnimi šiframi

Razlika med bločnimi in tokovnimi šiframi je v tem, da pri bločnih šifrah zaporedoma šifriramo bloke čistopisa z istim ključem in isto šifrirno funkcijo, medtem ko pri tokovnih šifrah šifriramo posamezne člene čistopisa s tokom ključev in funkcijo, ki se s časom spreminja. Prav tako smo omenili, da imajo tokovne šifre spomin, medtem ko ga bločne ne potrebujejo.

Vendar pa ta delitev ni stroga. Določene bločne šifre namreč lahko smatramo za tokovne. Primer takega je Vigenerjeva šifra, ki jo lahko smatramo za tokovno šifro s kratko periodo. Če poleg tega bločnim šifram dodamo spomin, dobimo tokovno šifro na velikih blokih (CBC način). V praksi so uporabljene mnoge tokovne šifre, ki so v tesni zvezi z bločnimi šiframi, oziroma načini šifriranje le-teh (EBC, CBC, CFB, OFB). Zato so v nadaljevanju ti načini podrobneje predstavljeni, prav tako tudi povezava s tokovnimi šiframi.

ECB - Electronic Codebook

Najbolj osnovni način bločnega šifriranja je ECB. Ima mnogo pomankljvosti in slabosti, vendar predstavlja osnovo ostalim, zato ga tu tudi omenjamo.

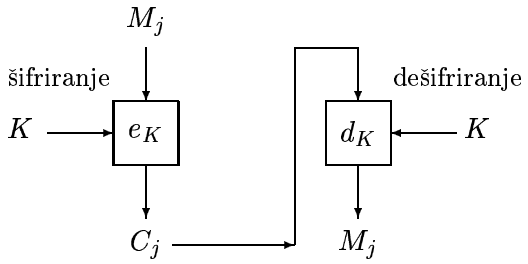
Definicija 2.12. *Naj bo K k-bitni ključ, M_i pa n-bitni blok čistopisa. Naj bosta e_K in d_K šifrirni in dešifrirni funkciji, ki delujeta na n-bitnih blokih. Čistopis M je sestavljen iz n-bitnih blokov M_i, tj. $M = M_1 M_2 \dots M_t$, tajnopis C pa iz n-bitnih blokov C_i, tj. $C = C_1 C_2 \dots C_t$. V ECB je šifriranje definirano z*

$$C_i = e_K(M_i),\ i = 1, 2, \dots, t,$$

dešifriranje pa z

$$M_i = d_K(M_i),\ i = 1, 2, \dots, t.$$

Postopka šifriranja in dešifriranja sta predstavljena na sliki 2.7.



Slika 2.7: ECB način.

Lastnosti načina ECB:

1. Identični bloki čistopisa se šifrirajo v identične bloke tajnopisa, kar napadalec lahko izkoristi.

2. Če permutiramo bloke tajnopisa, bo po dešifriranju rezultat permutacija istih blokov čistopisa.

3. Če spremenimo en ali več blokov tajnopisa, se bo to pri dešifriranju poznalo samo na spremenjenih blokih.

4. Bloki tajnopisa so neodvisni drug od drugega, torej dodajanje ali brisanje blokov ne vpliva na dešifriranje drugih blokov. Še več, ker identični bloki tajnopisa pripadajo identičnim blokom čistopisa, ta način ne skrije vzorca in statističnih lastnosti sporočila.

CBC - Cipher Block Chaning

Pri tem načinu šifriranja so bloki med sabo povezani z začetno vrednostjo oziroma začetnim blokom. Dodatna predpostavka pri temu načinu je, da sta prostora blokov tajnopisa in čistopisa identična in Abelovi grupi za operacijo seštevanja.

Definicija 2.13. *Naj bo K k-bitni ključ, M_i pa n-bitni blok čistopisa. Naj bosta e_K in d_K zaporedoma šifrirni in dešifrirni funkciji. Čistopis M je sestavljen iz n-bitnih blokov M_i, tj. $M = M_1 M_2 \dots M_t$, tajnopis C pa iz n-bitnih blokov C_i, tj. $C = C_1 C_2 \dots C_t$. Naj bo* IV *(angl. initial value) začetni, n-bitni blok. V CBC načinu je šifriranje definirano z:*

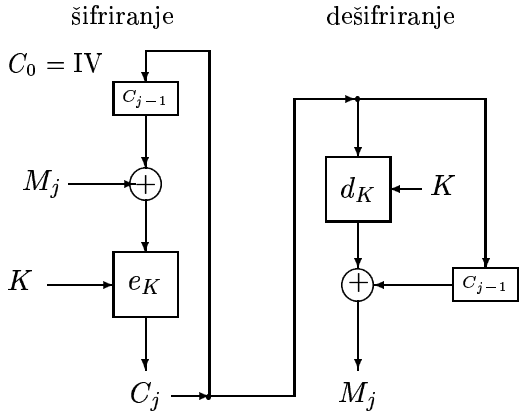
$$\begin{aligned} C_1 &= e_K(M_1 + \mathrm{IV}), \\ C_i &= e_K(M_i + C_{i-1}),\ i = 2, 3, \dots, t, \end{aligned}$$

dešifriranje pa z

$$\begin{aligned} M_1 &= d_K(C_1) - \mathrm{IV}, \\ M_i &= d_K(C_i) - C_{i-1},\ i = 2, 3, \dots, t, \end{aligned}$$

kjer je "$-$" inverzna operacija seštevanja "$+$" v grupi blokov.

CBC način je torej bločna šifra, ki pa smo mu dodali notranji spomin, torej bi ga lahko šteli tudi med tokovne šifre. Postopka šifriranja in dešifriranja sta razvidna iz slike 2.8.



Slika 2.8: CBC način

Lastnosti načina CBC:

1. Če uporabimo isti ključ K, potem identičnim blokom čistopisa pripadajo identični bloki tajnopisa.

2. Če se narobe odšifrira j-ti blok tajnopisa C_j, se bo narobe dešifriral tudi $j+1$ blok tajnopisa C_{j+1}.

3. Če v bloku C_j spremenimo en bit, se bo to poznalo pri dešifriranju blokov C_j in C_{j+1}. Blok C_j se dešifrira v naključen blok, pri dešifriranju bloka C_{j+1} pa bo napaka na tistem mestu, kjer smo spremenili blok C_j.

4. CBC se samo-sinhronizira, saj se pri modifikaciji bloka C_j narobe dešifrirata bloka C_j in C_{j+1}, blok C_{j+2} in ostali pa se dešifrirajo pravilno.

CFB - Cipher Feedback Chaining

CBC način šifrira n bitov čistopisa naenkrat z uporabo n-bitne bločne šifre. Nekatere aplikacije pa zahtevajo, da m bitov čistopisa zašifriramo in pošljemo brez čakanja. V tem primeru uporabimo način CFB.

Naj bo A^n prostor n-bitnih blokov tajnopisa in čistopisa in naj bo $(A^n, +)$ komutativna grupa, kjer je seštevanje elementov $B = (b_1, \ldots b_n), C = (c_1, \ldots, c_n)$ definirano z naslednjim pravilom:

$$B + C = (b_1 + c_1, \ldots, b_n + c_n).$$

Naj $e_K(X) : A^n \to A^n$ in $d_K(X) : A^n \to A^n$ označujeta izbrana postopka šifriranja in dešifriranja. Definiramo naslednji preslikavi:
rchop$_u$ - preslikava, ki odreže u bitov na desni strani,
lchop$_u$ - preslikava, ki odreže u bitov na levi strani.
Operacija $\|$ je stik (angl. concatenation) dveh blokov. CFB način je definiran z naslednjimi pravili:

Definicija 2.14. *Za ključ K, $m \in \{1, 2, \ldots, n\}$, prvi n-bitni blok X_1, i-ti blok čistopisa $M_i \in A^m$ definiramo:*

1. *šifriranje:* $C_i = M_i + \text{rchop}_{n-m}(e_K(X_i)); \quad X_{i+1} = C_i \,\|\, \text{lchop}_m(X_i)$,

2. *dešifriranje:* $M_i = C_i - \text{rchop}_{n-m}(e_K(X_i)); \quad X_{i+1} = C_i \,\|\, \text{lchop}_m(X_i)$.

Postopka šifriranja in dešifriranja sta razvidna tudi iz slike 2.9.

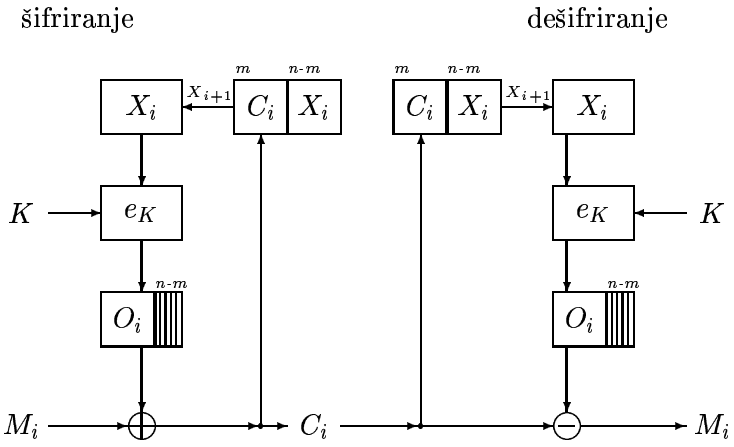
Lastnosti načina CFB:

1. Identični bloki tajnopisa pripadajo identičnim blokom čistopisa. Isti bloki se zašifrirajo (dešifrirajo) v iste bloke.

2. Če permutiramo bloke tajnopisa, se to pozna pri dešifriranju.

3. Ena ali več sprememb na poljubnem m-bitnem bloku tajnopisa C_j, se pozna pri dešifriranju naslednjih $\lceil n/m \rceil$ blokov tajnopisa. Pri tem se bo dešifrirani blok M_j' razlikoval od pravilno dešifriranega bloka M_j na tistem mestu, kjer smo napravili spremembo, medtem ko se bodo ostali bloki dešifrirali v naključen blok.

4. CFB je torej asinhrona šifra, saj je za ponovno sinhronizacijo potrebno $\lceil n/m \rceil$ blokov tajnopisa.



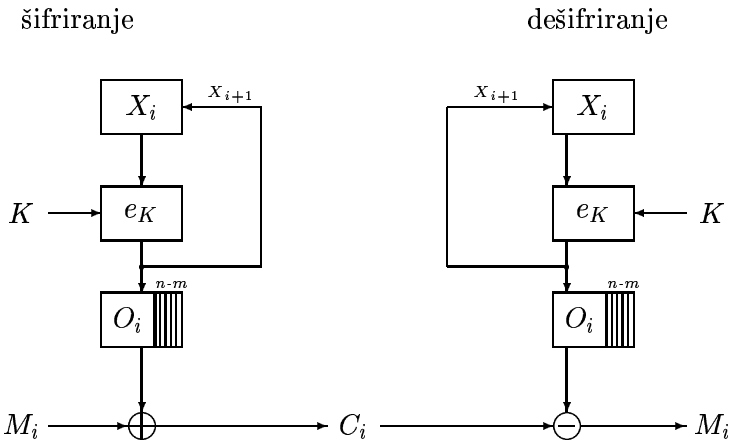
Slika 2.9: CFB način.

OFB - Output Feedback

Definicija 2.15. *Naj bodo* A^n, X_1, $e_K(X)$, $d_K(X)$, K, $\mathbf{rchop}_u$, $\mathbf{lchop}_u$, $(A^n, +)$ *in* m, $1 \leq m \leq n$ *definirani kot pri CFB. Za* i*-ti blok čistopisa* $M_i \in A^m$ *definiramo:*

1. *šifriranje:* $C_i = M_i + \mathrm{rchop}_{n-m}(e_K(X_i)); \quad X_{i+1} = e_K(X_i)$,
2. *dešifriranje:* $M_i = C_i - \mathrm{rchop}_{n-m}(e_K(X_i)); \quad X_{i+1} = e_K(X_i)$.

Postopka šifriranja in dešifriranja sta razvidna tudi iz slike 2.10.



Slika 2.10: OFB način.

Lastnosti načina OFB:

1. Identični bloki tajnopisa pripadajo identičnim blokom čistopisa. Isti bloki se zašifrirajo (dešifrirajo) v iste bloke.

2. Pri ponovni uporabi OFB moramo spremeniti začetno vrednost $X_1 = IV$, ker drugače dobimo identičen tok ključev.

3. Sprememba enega ali več bitov v poljubnem bloku tajnopisa se pozna samo pri dešifriranju spremenjenega bloka, natančneje, narobe se dešifrirajo natanko tisti biti bloka, ki smo jih spremenili.

4. OFB način je sinhrona šifra.

Poglavje 3

GENERATORJI TOKA KLJUČEV

Pomembno matematično orodje za modeliranje elektronskega hardverja so končni avtomati. Uporabni so tudi za generiranje neskončnih zaporedij nad končno abecedo in veliko generatorjev toka ključev v tokovnih šifrah lahko modeliramo z njimi. To je tudi razlog, da to poglavje najprej obravnava osnove teorije končnih avtomatov in jezikov. Temu sledi uporaba končnih avtomatov v generatorjih toka ključev, primeri generatorjev toka ključev, ki temeljijo na problemih iz teorije števil, poglavje se zaključi z osnovnimi lastnostmi, ki jih od generatorjev toka ključev pričakujemo, od katerih največ pozornosti namenimo linearni in sferni zahtevnosti.

3.1 Teorija avtomatov in jezikov

Teorija avtomatov in jezikov je precej obširna. Tu bo predstavljen samo del teorije, natančneje regularni izrazi, ki jih bomo potrebovali v poglavju 7. Navedeni izreki bodo brez dokazov, ki so na voljo v [19].

Preden se lotimo bistva tega razdelka, determinističnih končnih avtomatov, moramo definirati pojem jezika in abecede.

Definicija 3.1. Abeceda *je končna množica elementov, ki jim pravimo znaki ali simboli.* **Beseda** *W dolžine n nad abecedo Σ je element množice Σ^n. Besedo zapišemo kot: $W = w_1 w_2 \dots w_n$. Množico besed nad abecedo Σ označujemo s Σ^*, velja pa*

$$\Sigma^* = \bigcup_{n=0}^{\infty} \Sigma^n,$$

pri čemer je $\Sigma^0 = \{\epsilon\}$, kjer je ϵ prazna beseda, njena dolžina je 0.

Nad besedami dane abecede definiramo binarno operacijo, ki ji pravimo *stik* ali *kokatenacija* (angl. cocatenation), katere pomen je preprosto stikanje dveh besed v eno.

Definicija 3.2. *Naj bo Σ dana abeceda.* **Stik** *ali* **kokatenacija** *je binarna operacija nad besedami $X, Y \in \Sigma^*$, $X = x_1 x_1 \dots x_m, Y = y_1 y_2 \dots y_n$ abecede Σ definirana s pravilom:*

$$X \parallel Y = x_1 x_2 \dots x_m y_1 y_2 \dots y_n.$$

Očitno je prazna beseda ϵ enota za zgornjo operacijo in množica besed skupaj z ϵ tako postane monoid. Zdaj, ko smo definirali abecedo, besede in operacijo nad besedami, lahko definiramo pojem formalnega jezika.

Definicija 3.3. Formalni jezik *je pri neki izbrani abecedi Σ podmnožica Σ^*.*

Najbolj preprosti primeri jezikov so $\emptyset$, $\{\epsilon\}$, Σ^*. Zdaj se lahko lotimo deterministočnih končnih avtomatov (DKA) in njihovih lastnosti.

Definicija 3.4. Deterministični končni avtomat *(bf DKA) M je peterka* $(Q, \Sigma, \delta, q_0, F)$, *kjer je:*

1. Q *neka končna* **množica stanj**,
2. Σ *končna abeceda,*
3. **funkcija prehodov** $\delta : Q \times \Sigma \to Q$, *neka totalna funkcija (*$f : S \to T$ *je totalna, če je domena funkcije f,* $\mathcal{D}_f = S$*),*
4. $q_0 \in Q$ *neko posebno začetno stanje in*
5. $F \subseteq Q$ *neka množica končnih stanj.*

Preden prevedemo definicijo 3.3 jezika na deterministične končne avtomate, potrebujemo točno definicijo prehodne funkcije. Zato funkcijo δ v definiciji DKA razširimo tako, da bo definiran na besedah in ne le na simbolih abecede.

Definicija 3.5. *Naj bo dana funkcija* $\delta : Q \times \Sigma \to Q$. **Razširjeno funkcijo** $\tilde{\delta} : Q \times \Sigma \to Q$ *definiramo takole:*

1. $\forall q \in Q \Rightarrow \tilde{\delta}(q, \epsilon) = q$,
2. $\forall q \in Q, W \in \Sigma^*, a \in \Sigma \Rightarrow \tilde{\delta}(q, Wa) = \delta(\tilde{\delta}(q, W), q)$.

Torej če velja $\tilde{\delta}(q, W) = r$, potem to ustreza stavku: "vhodna beseda W nas pripelje iz stanja q v stanje r". Zdaj lahko definiramo jezik nekega deterministočnega avtomata.

Definicija 3.6. Jezik *nekega deterministočnega končnega avtomata M, ki ga označimo z L(M), je množica besed* $L(M) = \{W \ ; \ \tilde{\delta}(q_0, W) \in F\}$. *Drugače povedano, jezik je množica besed, ki nas pripeljejo iz začetnega stanja* q_0 *v neko končno stanje.*

Vpeljimo zdaj naslednje oznake. Naj bo Σ neka abeceda in $\boldsymbol{x} = x_1 x_2 \ldots x_n \in \Sigma^*$. Oznaka $\boldsymbol{x}^R$ je definirana kot $\boldsymbol{x}^R = x_n x_{n-1} \ldots x_1$. Če je L nek jezik nad Σ, potem je $L^R = \{w \ ; \ w^R \in L\}$ tudi jezik nekega končnega avtomata.

Doslej smo za opisovanje nekega jezika uporabljali teoretičen model, ki sloni na predstavi stroja (avtomata), ki sproti preverja, ali je neka beseda v jeziku. Vendar so jezike že od nekdaj opisovali tudi z gramatikami in sintaksami. Pri tem gre za pravila, ki nakazujejo, kako so zgrajene pravilne jezikovne oblike. V nadaljevanju bomo podali opis nekega razreda sintaks, ki opisujejo jezike, ki jih sprejemajo končni avtomati.

Definicija 3.7. *Naj bo* Σ *neka končna abeceda, ki ne vsebuje simbolov* $\{\bar{\emptyset}, \bar{\epsilon}, \|, +, *, (,)\}$. *Naj bo* $\bar{\Sigma} = \{\bar{a} \mid a \in \Sigma\}$ *prav tako končna abeceda.* **Regularni izrazi** *nad* Σ *so jezik, ki ga sestavljajo besede nad abecedo* $\bar{\Sigma} \cup \{\bar{\emptyset}, \bar{\epsilon}, \|, +, *, (,)\}$, *in ki so zgrajene na podlagi naslednje induktivne definicije.*

1. $\bar{\emptyset}, \bar{\epsilon}$ *in* $\bar{a}, a \in \Sigma$, *so regularni izrazi.*
2. *Naj bosta E in F regularna izraza. Potem so* $(E + F), (E \parallel F), (E)^*$ *regularni izrazi.*
3. *Vsi regularni izrazi so zgrajeni po zgornjih dveh pravilih.*

Vsakemu regularnemu izrazu nad Σ pripisujemo neki pomen, predstavlja namreč določen jezik nad Σ. Da bi natanko opisali ta pomen, bomo definirali nekaj operacij nad jeziki oziroma množicami besed.

Definicija 3.8. *Naj bosta L_1 in L_2 jezika nad abecedo Σ. Potem je* **stik** *jezikov definiran kot*

$$L_1 \parallel L_2 = \{X \ ; \ X = X_1 \parallel X_2, \ X_1 \in L_1, X_2 \in L_2\}.$$

Iteracija *jezika L_1 je definirana kot*

$$L_1^i = \underbrace{L_1 \parallel L_1 \parallel \ldots \parallel L_1}_{i}, \ L_1^0 = \{\epsilon\}.$$

Pomen nekega regularnega izraza zdaj lahko definiramo induktivno.

Definicija 3.9. *Regularen izraz E predstavlja jezik L(E), ki je določen na naslednji način:*

1. *V primeru $E = \bar{a}$, $a \in \Sigma$, velja $L(E) = \{a\}$, v primeru $E = \{\bar{\emptyset}\}$ velja $L(E) = \emptyset$ in v primeru $E = \bar{\epsilon}$ velja $L(E) = \{\epsilon\}$.*

2. *Naj bosta E_1 in E_2 regularna izraza, ki imata predpisana jezika $L(E_1)$ in $L(E_2)$. Potem velja*

$$\begin{aligned} L((E_1 + E_2)) &= L(E_1) \cup L(E_2), \\ L((E_1 \parallel E_2)) &= L(E_1) \parallel L(E_2) \end{aligned}$$

in

$$L((E_1)^*) = L(E_1)^*.$$

Oglejmo si zdaj primer regularnih izrazov in jezikov, ki jih ti izrazi predstavljajo.

Primer. Naj bo $\Sigma = \{0, 1\}$ abeceda in naj bosta E_1 in E_2 naslednja izraza:

$$\begin{aligned} E_1 &= (((\bar{0})^* \parallel \bar{1}) \parallel (\bar{0})^*), \\ E_2 &= ((\bar{0} + \bar{1}) \parallel \bar{1}). \end{aligned}$$

Da sta to res regularna izraza sledi iz definicije regularnih izrazov. Poglejmo si, kakšna jezika predstavljata ta dva izraza. Besede jezika, ki ga predstavlja E_1, so sestavljene iz poljubnega števila ničel, ki jim sledi ena enica, in zopet poljubno število ničel. Jezik, ki ga predstavlja E_2 pa je množica besed $\{01, 11\}$. •

Naslednji izrek bo pokazal ekvivalentnost regularnih izrazov in končnih avtomatov. Vsak regularen izraz predstavlja jezik, ki ga sprejema nek končen avtomat, in nasprotno, vsak končen avtomat sprejema jezik, ki ga je možno predstaviti z nekim regularnim izrazom.

Izrek 3.10. *Naj bo E regularen izraz nad Σ. Potem obstaja končen avtomat $M \in DKA$, ki sprejme jezik L(E).*

Izreka ne bomo dokazali, pokazali bomo samo idejo. Dokaze vseh korakov lahko bralec najde v [19]. Najprej vpeljimo naslednje oznake. NKA označuje razred nedeterminističnih končnih avtomatov, $\epsilon -$ NKA označuje razred nedeterminističnih končnih avtomatov z ϵ prehodi. Za formalne definicije obeh glej [19].

Skica dokaza. Najprej pokažemo, da za regularen izraz E nad abecedo Σ obstaja končen avtomat $M'' \in \epsilon$-NKA. Nato pokažemo, da je razred ϵ-NKA ekvivalenten razredu NKA. Dokaz zaključimo z ekvivalenco med razredom DKA in NKA. ■

Izrek 3.11. *Naj bo* $M = (Q, \Sigma, \delta, q_1, F) \in$ DKA *nek končni avtomat. Potem obstaja regularen izraz* E *nad* Σ *z lastnostjo* $L(E)=L(M)$*, kjer je* $L(M)$ *jezik, ki ga avtomat* M *sprejme.* ■

Zopet omenimo le idejo dokaza. Celoten dokaz je na voljo v [19, str.45]. Dokaz gradimo tako, da postopoma sestavljamo boljše približke jezika $L(M)$. Za vsak tak približek dokažemo, da ga lahko predstavimo z regularnim izrazom.

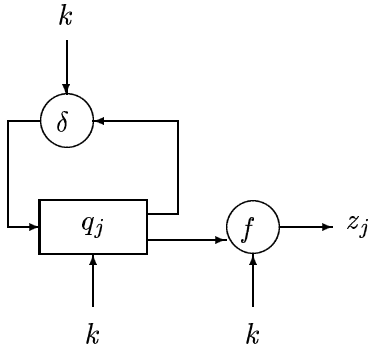
Zaradi izrekov 3.10 in 3.11 bomo v nadaljevanju jezikom, ki jih sprejemajo končni avtomati, rekli **regularni jeziki.** Več o lastnostih regularnih jezikov in izrazov je na voljo v [19].

3.2 Generatorji toka ključev kot končni avtomati

Pri sinhroni tokovni šifri lahko generator toka ključev gledamo kot poseben deterministično končen avtomat. Od klasičnega deterministčnega končnega avtomata, definiranega v definiciji 3.4 se razlikuje v dodatni funkciji. Tako je generator toka ključev kot končen deterministično avtomat šesterka, $(Q, \Sigma, \delta, q_0, F, f)$, kjer je:

1. Q neka končna množica stanj,
2. Σ končna abeceda,
3. funkcija prehodov δ neka totalna funkcija $\delta : Q \times \Sigma \to Q$,
4. $q_0 \in Q$ začetno stanje,
5. $F \subseteq Q$ neka podmnožica končnih stanj in
6. **izhodna funkcija** $f : Q \times \Sigma \to \Sigma$, ki iz trenutnega stanja in simbola abecede generira izhodni simbol.

Včasih tako konstruiranemu generatorju toka ključev dodamo še izhodno abecedo, ponavadi pa sta izhodna abeceda in Σ identični. Ključ je lahko določen simbol abecede, lahko pa je tudi beseda nad abecedo Σ. V tem primeru, moramo funkciji δ in f razširiti nad množico besed Σ^*, podobno kot smo to storili pri deterministčnih končnih avtomatih. Ključ lahko določa prehodno funkcijo, izhodno funkcijo in začetno stanje. Generator toka ključev kot deterministični končni avtomat je ilustriran na sliki 3.1.



Slika 3.1: Generator toka ključev kot končni avtomat.

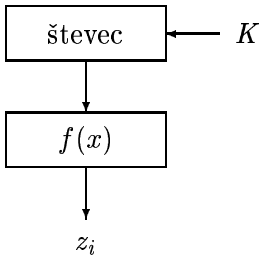
Osnovni problem konstrukcije generatorja toka ključev je najti funkcijo δ, ki jo imenujemo tudi *funkcija naslednjega stanja* in izhodno funkcijo f, ki generira tok ključev $z^\infty = z_0 z_1 \ldots$, in

zadošča določenim kriptografskim zahtevam. V ta namen za generatorje toka ključev uporabljajo posebne razrede deterministočnih končnih avtomatov, *števce.*

Generatorji toka ključev, ki temeljijo na števcih

Definicija 3.12. Števec *je deterministočen končen avtomat s periodo N, ki ciklično šteje števila od 0 do N-1. To je, DKA z izhodno funkcijo f, ki zaporedoma ciklično vrača števila* $0 \ldots N-1$.

Generator toka ključev, ki temelji na števcu, je sestavljen iz števca C, ključa K in funkcije $f : \mathbb{Z}_N \to G$, kjer je $(G, +)$ Abelova grupa. Pri tem je ključ $K \in \{0, 1, \ldots, N-1\}$ in števec začne ciklično šteti pri vrednosti K. Argumenti funkcije f so zaporedna cela števila, ki jih konstruira števec, izhodno zaporedje z^∞ generatorja pa je dano z $z_i = f(i + K) \mod M, \quad 0 \leq M \leq N$. Tak generator je razviden tudi s slike 3.2.



Slika 3.2: NSG.

Kot je razvidno s slike, je ključ K začetna vrednost v generatorju toka ključev, medtem ko je funkcija f fiksirana. Takim generatorjem toka ključev pravimo tudi *naravni generatorji zaporedij* (angl. Natural Stream Generator - NSG) in so pomembni zato, ker lahko z njimi generiramo vsako periodično zaporedje. Več o NSG je na voljo v [7].

3.3 Generatorji toka ključev s teorijo števil

Alternativa zgornjim generatorjem toka ključev so generatorji psevdo naključnih števil, ki temeljijo na različnih problemih teorije števil. V tem razdelku bodo predstavljeni osnovni primeri takšnih generatorjev in njihove lastnosti. Več o generatorjih psevdo naključnih števil je na voljo v [22].

1. **Multiplikativen generator**
   Naj bodo (a, b, M) celoštevilski parametri in $0 \leq x_0 \leq M-1$ seme. Generator je definiran s pravilom:
   $$x_{n+1} = (x_n a + b) \mod M,$$
   kjer je $x = x_0, x_1, x_2, \ldots$ zaporedje psevdo naključnih števil.

   Lastnosti:

   (a) Če poznamo M, lahko a in b izračunamo iz treh zaporednih členov zaporedja x, x_1, x_2, x_3.

   (b) Če a, b in M niso znani, potem obstaja polinomski algoritem, ki izračuna x'_{n+1} iz $x_1, x_2, \ldots, x_n$, glej [5].

2. **$1/p$ generator**
   Naj bo p praštevilo in d izbrana baza številskega sistema. Potem lahko ulomek $1/p$ zapišemo kot
   $$\frac{1}{p} = 0, d_1 d_2 \dots d_j d_{j+1} \dots$$
   v bazi d. Izbrana parametra sta (p, d), seme je pozitivno število j, generator pa je definiran s pravilom:
   $$x_n = d_{j+n}.$$
   Več o lastnostih tega generatorja je na voljo v [5, str.307-339].

3. **Potenčni generator**
   Naj bosta (d, N) izbrana parametra in x_0 seme. Generator je definiran s pravilom:
   $$x_{n+1} = x_n^d \mod N.$$
   Tak tip generatorja se v praksi uporablja v različnih sistemih, zato si bomo podrobneje ogledali dve različici tega generatorja, *RSA generator*, ki se uporablja v asimetričnem kriptosistemu RSA in *kvadratni generator*. Več o lastnostih potenčnega generatorja je na voljo v [22].

   (a) *RSA generator*
   Naj bo $N = p_1 \cdot p_2$, kjer sta p_1 in p_2 praštevili. Parameter d si izberemo tako, da je $\gcd(d, (p_1 - 1)(p_2 - 1)) = 1$. Naj bo x_0 izbrano k-bitno seme, kjer je velikost k odvisna od p_1 in p_2, ki naj bosta $k/2$ bitni števili. Potem je
   $$x_{n+1} = x_n^d \mod N.$$

   (b) *Kvadratni generator*
   Drugi primer potenčnega generatorja kvadratni generator, kjer je $N = p_1 p_2$ in $p_1 \equiv p_2 \equiv 3 \mod 4$. Za parameter d vzamemo $d = 2$. Generator je potem
   $$x_{n+1} = x_n^2 \mod N.$$

4. **Eksponentni generator**
   Naj bosta (q, N) izbrana parametra in x_0 izbrano seme. Generator je definiran s pravilom
   $$x_{n+1} = q^{x_n} \mod N.$$
   Več o eksponentnem generatorju je na voljo v [5, str.24].

Predstavljeni generatorji predstavljajo samo vzorec generatorjev psevdo-naključnih števil. Kljub temu, da imajo nekateri med njimi dobre statistične lastnosti, se v tokovnih šifrah ne uporabljajo, saj so njihove implementacije počasnejše (dražje) od generatorjev toka ključev, ki temeljijo na pomičnih registrih.

3.4 Kriptografske lastnosti generatorjev toka ključev

Dober generator toka ključev mora zadoščati določenim zahtevam, ki ga obvarujejo pred znanimi napadi. Katerim zahtevam mora zadoščati je odvisno od uporabe generatorja toka ključev in od možnih napadov nanj. Obstaja nekaj osnovnih zahtev, ki izhajajo iz kriptografskih lastnosti zaporedij, saj je navsezadnje tok ključev zaporedje. Te lastnosti so:

1. linearna zahtevnost,
2. sferna zahtevnost,
3. porazdelitev vzorcev,
4. lastnost avtokorelacije.

Najbolj pomembna od naštetih je prva, linearna zahtevnost, saj se neposredno nanaša na poseben primer generatorja toka ključev, LFSR, ki ga kot gradnik uporabljamo v mnogih generatorjih toka ključev in ga bomo v nadaljevanju tudi podrobneje analizirali. Ostale lastnosti bomo samo definirali in omenili razloge zanje.

Linearna zahtevnost

Definicija 3.13. *Naj bo s^n zaporedje $s_0 s_1 \dots s_{n-1}$ dolžine n, katerega elementi ležijo v obsegu GF(q). (V primeru $n = 0$, je s^0 prazno zaporedje, v primeru $n = \infty$, pa s^∞ označuje polneskončno zaporedje $s_0 s_1 \dots s_n \dots$.)* **Linearna zahtevnost** *zaporedja s^n glede na GF(q), je najmanjše nenegativno celo število $L = L(s^n)$, tako da obstajajo $c_1, c_2, \dots c_L \in GF(q)$ za katere velja*

$$s_j + c_1 s_{j-1} + \cdots + c_L s_{j-L} = 0, \quad \text{za } L \le j \le n. \tag{3.1}$$

Iz definicije takoj sledijo naslednje lastnosti:

1. $L(s^n) = 0$ natanko tedaj, ko je $s^n = 0^n$, kjer je 0^n zaporedje samih ničel dolžine n.
2. $L(s^n) = n$ natanko tedaj, ko je $s^{n-1} = 0^{n-1}$ in $s_{n-1} \neq 0$.

Ena izmed interpretacij linearne zahtevnosti $L(s^n)$ je dolžina L najkrajšega LFSR-ja, ki generira zaporedje s^n. Ekvivalenčnost obeh definicij bomo pokazali v poglavju o LFSR-generatorju. Za računanje linearne zahtevnosti danega zaporedja in konstrukcijo ustreznega LFSR-ja obstaja učinkovit algoritem, ki je predstavljen v poglavju o LFSR-generatorju.

S stališča kriptografije, je najbolj zanimiva linearna zahtevnost periodičnih zaporedij. Očitno je, da če je s^∞ periodično zaporedje s periodo N, potem je $L(s^\infty) \le N$, in obstajajo enolični koeficienti $c_1, c_2, \dots, c_L$, tako da je zadoščeno pogoju (3.1) za vse j, $j \ge 1$, kjer je $L = L(s^\infty)$. Še več, za periodična zaporedja je $c_L \neq 0$, tako da je linearna zahtevnost periodičnega zaporedja natanko red homogene linearne rekurzije (3.1) najmanjše stopnje, ki ustreza zaporedju.

Definicija 3.14. *Naj bo s^∞ zaporedje elementov iz GF(q). Polinom $f_s(x) = f(x)$ nad GF(q) je* **minimalen polinom zaporedja**, *če je*

1. $f(x) = c_0 + c_1 x + \cdots + c_n x^n$, in $c_0 = 1$, $c_n \neq 0$,
2. *če velja linearna rekurzija $s_v = c_1 s_{v-1} + c_2 s_{v-2} + \cdots + c_n s_{v-n}$, za vsak $v \ge n$ in*
3. *ne obstaja noben polinom manjše stopnje, ki ustreza pogojema 1 in 2.*

Trditev 3.15. *Linearna zahtevnost zaporedja je enaka stopnji minimalnega polinoma tega zaporedja.* ■

Oglejmo si zdaj rodovno funkcijo zaporedja s^∞ nad obsegom $GF(q)$, ki je definirana kot

$$s(x) = \sum_{i=0}^{\infty} s_i x^i. \tag{3.2}$$

Velja naslednji izrek.

Izrek 3.16. *Rodovna funkcija vsakega periodičnega zaporedja se da izraziti kot racionalna funkcija, tj. obstajata polinoma f in g, tako da velja*

$$s(x) = \frac{g(x)}{f(x)},$$

kjer $f(0) \neq 0$ in $\deg(g) < \deg(f)$.

Dokaz. Naj bo s^∞ periodično zaporedje s periodo N in $s^N(x) = s_0 + s_1x + \cdots + s_{N-1}x^{N-1}$. Potem velja

$$(1 - x^N)s(x) = s^N(x),$$

saj je $s_{j+N} = s_j$ za vsak j, $j \geq 0$. Sledi

$$s(x) = \frac{s^N(x)}{1 - x^N}.$$

■

Tako zapisano rodovno funkcijo zaporedja s^∞ imenujemo **racionalna oblika** rodovne funkcije $s(x)$ zaporedja s^∞.

Izrek 3.17. *Naj bo*

$$s(x) = \frac{r(x)}{f(x)}, \text{ kjer je } \deg(f) > \deg(r),\ f(0) = 1,$$

racionalna oblika rodovne funkcije periodičnega zaporedja s^∞. Potem je f(x) minimalen polinom zaporedja s^∞ natanko tedaj, ko je $\gcd(f(x), r(x)) = 1$, kjer je gcd(f,g) največji skupni delitelj polinomov f in g.

Dokaz. Naj bo $f_s(x) = c_0 + c_1x + \cdots c_nx^n$ minimalen polinom periodičnega zaporedja s^∞. Potem velja

$$f_s(x)s(x) = \sum_{k=0}^{\infty} \left(\sum_{0 \leq i \leq \min\{n,k\}} c_i s_{k-i} \right) x^k \quad \text{in} \quad \sum_{0 \leq i \leq \min\{n,k\}} c_i s_{k-i} = 0,$$

za vsak k, $k \geq n$. Sledi

$$f_s(x)s(x) = \sum_{k=0}^{n-1} \left(\sum_{0 \leq i \leq k} c_i s_{k-1} \right) x^k. \tag{3.3}$$

Označimo polinom na desni strani enačbe (3.3) z $r_s(x)$ in pokažimo, da je $\gcd(f_s, r_s) = 1$. Naj bo

$$g(x) = \frac{f_s(x)}{\gcd(f_s, r_s)} = g_0 + g_1x + \cdots + g_mx^m \quad \text{in} \quad h(x) = \frac{r_s(x)}{\gcd(f_s, r_s)}.$$

Sledi, $\deg(h) < \deg(g)$ in $g(x)s(x) = h(x)$. To pomeni, da je

$$\sum_{0 \leq i \leq m} g_i s_{k-1} = 0,$$

za vsak k, $k \geq m$. Ker je f_s minimalen polinom, je $k = m$ in zato tudi $\gcd(f_s, r_s) = 1$.

Dokažimo še obrat. Naj bo $s(x) = g(x)/f(x)$ racionalna oblika rodovne funkcije $s(x)$, kjer je $\gcd(f, g) = 1$ in $f(0) = 1$. Če je $\deg(f) = m$ in $f(x) = \sum_{i=0}^{m} b_i x^i$, $b_0 = 1$, potem lahko zaporedje s^∞ generiramo z naslednjo rekurzivno enačbo:

$$s_v = b_1 s_{v-1} + \ldots + b_m s_{v-m} \text{ za } v \geq m.$$

Minimalni polinom f_s deli f, glej izrek 6.42 v [21, str.210]. Naj bo $f = f_s \cdot h$. Potem je $g(x) = f(x)s(x) = hf_s s(x) = h(x)r_s(x)$. Ker je $\gcd(f, g) = 1$, mora biti $h(x)$ enak 1. Torej je $f = f_s$ minimalen polinom zaporedja s^∞. ■

Če je v racionalni obliki rodovne funkcije $s(x) = g(x)/f(x)$ zaporedja s^∞, polinom f v imenovalcu minimalen polinom zaporedja, potem tako obliko rodovne funkcije imenujemo **reducirana racionalna oblika.**

Izrek 3.18. *Naj bo s^∞ periodično zaporedje s periodo N. Potem je minimalen polinom zaporedja s^∞ enak*

$$f_s(x) = \frac{1 - x^N}{\gcd(s^N(x), 1 - x^N)}.$$

Dokaz. Očitno je $s^N(x)/(1 - x^N)$ racionalna oblika rodovne funkcije $s(x)$, prav tako tudi $g(x)/f(x)$, kjer je

$$g(x) = \frac{s^N(x)}{\gcd(s^N(x), 1 - x^N)}$$

in

$$f(x) = \frac{1 - x^N}{\gcd(s^N(x), 1 - x^N)}.$$

Ker je $\gcd(g, f) = 1$, lahko uporabimo izrek 3.17, po katerem je $f(x)$ minimalen polinom. ■

Izrek 3.19. *Naj bosta s(x) in t(x) zaporedoma racionalni obliki rodovnih funkcij zaporedij s^∞ in t^∞ ter*

$$s(x) = \frac{r_s(x)}{f_s(x)}, \quad t(x) = \frac{r_t(x)}{f_t(x)}.$$

Potem je minimalen polinom vsote zaporedij s in t enak

$$f_{s+t} = \frac{f_s f_t}{\gcd(f_s f_t, f_s r_t + r_s f_t)}.$$

Dokaz. Iz reduciranih racionalnih oblik rodovnih funkcij zaporedij s^∞ in t^∞ lahko izpeljemo naslednjo formulo:

$$s(x) + t(x) = \frac{f_s r_t + r_s f_t}{f_s f_t}.$$

Če v ulomku na desni imenovalec in števec krajšamo z $\gcd(f_s f_t, f_s r_t + r_s f_t)$, potem je v okrajšanem ulomku največji skupni delitelj imenovalca in števca enak 1 in za dokaz izreka lahko uporabimo izrek 3.17. ■

Iz izreka 3.19 sledi preprosta posledica, ki se lahko uporabi pri konstrukciji generatorja toka ključev, sestavljenega iz več vzporednih LFSR-generatorjev.

Posledica 3.20. *Naj bosta s^∞ in t^∞ zaporedji. Potem je $L(s + t) \leq L(s) + L(t)$.* ■

Utežena in sferna zahtevnost

Linearna zahtevnost je pomembna lastnost toka ključev, ni pa edina. Zaporedje ima lahko veliko linearno zahtevnost, vendar pa se ga kljub temu da aproksimirati z zaporedji, ki imajo majhno linearno zahtevnost, glej [5, str.30]. Odločilni lastnosti za dobro aproksimacijo sta prav utežena in sferna zahtevnost, ki si ju bomo v nadaljevanju podrobneje ogledali.

Definicija 3.21. *Naj bo $s = s^n$ zaporedje elementov iz $GF(q)$ in $L(s)$ njegova linearna zahtevnost. Naj $W_H(y)$ označuje Hammingovo utež, tj. število elementov zaporedja y različnih od nič.* **u-težena zahtevnost zaporedja** *s (angl. Weight Complexity) je število*

$$\mathrm{WC}_u(s) = \min\{L(s+y),\ W_H(y) = u\}.$$

Oglejmo si zdaj geometrijsko ozadje zgornje definicije. Naj bo $GF(2)^n$ prostor s Hammingovo razdaljo d_H ($d_H(x,y)$ = število mest, kjer se vektorja razlikujeta). Naj bo $S_u(s) = \{y\ ;\ d_H(s,y) = u\}$. Potem sledi $\mathrm{WC}_u(s) = \min\{L(y),\ y \in S_u(s)\}$.

Definicija 3.22. *Naj bo $B_u(s) = \{y\ ;\ d_H(s,y) \leq u\}$ krogla s središčem v točki s.* **u-sferno zahtevnost** *(angl. Sphere Complexity) definiramo kot*

$$\mathrm{SC}_u(s) = \min\{L(y),\ y \in B_u(s)\}.$$

Iz kriptografskega stališča nas zanima, koliko se linearna zahtevnost zaporedja zmanjša, če spremenimo določeno število členov. Pri tem bodo števila $\mathrm{WC}_u(s), \mathrm{SC}_u(s)$ in $|L(s) - SC_u(s)|$ mere stabilnosti linearne zahtevnosti zaporedja s.

Nekaj osnovnih lastnosti uteženе zahtevnosti za končna zaporedja $s = s^n$:

1. $\mathrm{WC}_0(s) = L(s)$,
2. $L(s) - 1 \leq \mathrm{WC}_n(s) \leq L(s) + 1$,
3. $\mathrm{WC}_u(s \oplus y) \leq \mathrm{WC}_u(s) + L(s)$.

Prva lastnost je posplošitev linearne zahtevnosti, druga lastnost nam daje spodnjo in zgornjo mejo u-uteženе zahtevnosti, tretja lastnost pa zagotavlja zgornjo mejo u-uteženе zahtevnosti vsote dveh zaporedij.

Podobno lahko sferno in uteženo zahtevnost definiramo za periodična zaporedja.

Definicija 3.23. *Naj bo s^∞ zaporedje elementov iz $GF(q)$ s periodo N. Sferna in u-uteženа zahtevnost sta definirani z*

$$\begin{aligned} \mathrm{WC}_u(s^\infty) &= \min\{L(s^\infty + t^\infty),\ W_H(t^N) = u,\ \mathrm{Per}(t^\infty) = N\}, \\ \mathrm{SC}_u(s^\infty) &= \min\{L(s^\infty + t^\infty),\ 0 < W_H(t^N) \leq u,\ \mathrm{Per}(t^\infty) = N\}. \end{aligned}$$

Ena izmed interpretacij $\mathrm{SC}_u(s^\infty)$ je dolžina najkrajšega LFSR-ja, ki generira zaporedje t^∞, ki se z zaporedjem s^∞ ujema z verjetnostjo večjo od $1-(u/N)$, kjer je N perioda zaporedja s^∞. Podobno, $\mathrm{WC}_u(s^\infty)$ lahko interpretiramo kot dolžino najkrajšega LFSR-ja, ki generira zaporedje t^∞, ki se z zaporedjem s^∞ ujema z verjetnostjo enako $1-(u/N)$.

Primer. Naj bo $s^\infty = \underbrace{0\ldots01}_{N}\underbrace{0\ldots01}_{N}\ldots$. Linearna zahtevnost tega zaporedja je N, sferna zahtevnost $SC_1(s^\infty) = 1$, tj. obstaja LFSR dolžine 1, ki generira zaporedje z verjetnostjo ujemanja $1-(1/N)$. •

Poglavje 4

LINEARNI POVRATNI POMIČNI REGISTER

Linearni povratni pomični register (angl. Linear Feedback Shift Register - LFSR) je linearni generator zaporedja. Uporablja se v mnogih generatorjih toka ključev. Razlogov za to je več. Naštejmo samo najpomembnejše.

1. LFSR-generator je zelo primeren za hardversko implementacijo.
2. Generira zaporedja z veliko periodo.
3. Generira zaporedja z dobrimi statističnimi lastnostmi.
4. Zaradi enostavne strukture je dobro analiziran z različnimi algebraičnimi tehnikami.

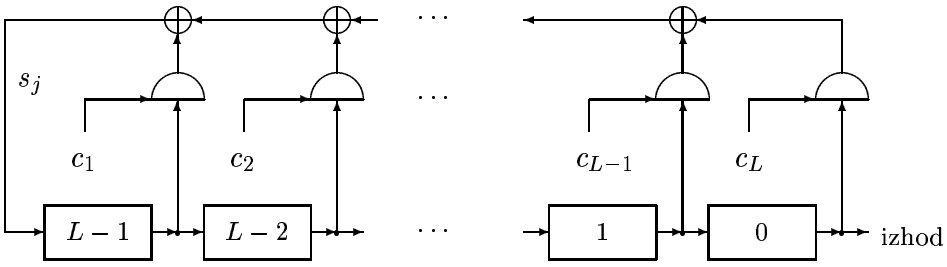
V nadaljevanju najprej definiramo LFSR-generator in naštejemo osnovne lastnosti. Tu uporabimo osnovne lastnosti končnih obsegov, in najpomembnejše tudi dokažemo. Ostale lahko najdemo v [21]. Nato sledi analiza linearne zahtevnosti LFSR-generatorja. Poglavje zaključimo z Berlekamp-Massey-jevim algoritmom za izračun linearne zahtevnosti danega zaporedja in konstrukcijo najkrajšega LFSR-generatorja, ki generira neko dano zaporedje.

4.1 LFSR-generator

V tem razdelku bomo najprej definirali linearni pomični register. Nato si bomo ogledali njegove osnovne lastnosti, kjer bomo največ pozornosti namenili periodi. Za analizo periode bomo potrebovali lastnosti končnih obsegov. Zato je del razdelka posvečen tudi končnim obsegom. Glaven namen razdelka bo povezati red polinoma s periodo LFSR-generatorja.

Definicija 4.1. Linearni povratni pomični register (LFSR) *dolžine L je linearni generator zaporedja $\{s^n\}$. Sestavljen je iz L enot, ki jih označimo z $0, 1, \ldots, L-1$, glej sliko 4.1. Vsaka enota je sposobna hranjenja enega bita. Ima en izhod in vhod ter uro, ki kontrolira premik podatkov. Vsako časovno enoto se izvedejo naslednje operacije.*

1. *Vsebina enote 0 je izhodna informacija in tvori del izhodnega zaporedja $\{s^n\}$.*
2. *Vsebina enote i se prepiše v enoto $i-1$ za $1 \leq i \leq L-1$.*
3. *Nova vsebina enote $L-1$ je povratni bit s_j, ki se izračuna tako, da seštejemo fiksirano število vsebin d prejšnjih enot iz množice $\{0, 1, \ldots, L-1\}$ po modulu 2.*



Slika 4.1: Delovanje LFSR-generatorja dolžine L.

Slika 4.1 prikazuje LFSR. Glede na definicijo 4.1 je na sliki vsak c_i bodisi 0 ali 1; polkrogi predstavljajo AND vrata; povratni bit s_j pa je vsota vsebin vseh tistih enot i, $0 \leq i \leq L-1$, za katere je $c_{L-i} = 1$.

Naj bo

$$C(x) = 1 + c_1 x + c_2 x^2 + \cdots + c_L x^L \ \in \mathbb{Z}_2[x].$$

Potem pravkar definirani LFSR-generator označimo z **LFSR(L,C(x))**.

Definicija 4.2. *LFSR-generator LFSR(L,C(x)) je* **nesingularen**, *če je stopnja polinoma $C(x)$ enaka L, torej $c_L = 1$. Če je začetna vsebina enote i, $s_i \in \{0,1\}$ za vsak i, $0 \leq i \leq L-1$, potem podzaporedju $[s_{L-1}, s_{L-2}, \ldots, s_1, s_0]$ pravimo* **začetno stanje** *za LFSR$(L, C(x))$.*

Ni se težko prepričati, da je naslednji člen linearna kombinacija L prejšnjih. Tako velja naslednja trditev.

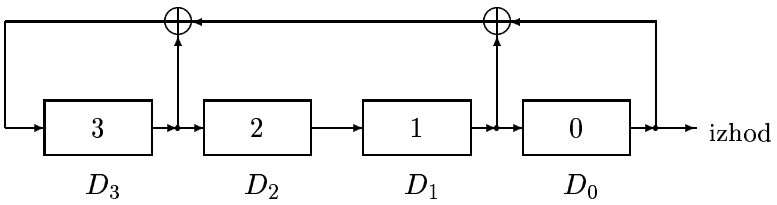
Trditev 4.3. *Če je podzaporedje $[s_{L-1}, \ldots, s_1, s_0]$ začetno stanje za LFSR$(L, C(x))$, potem je izhodno zaporedje $s = s_0, s_1, s_2, \ldots$ enolično določeno z naslednjo rekurzijo*

$$s_j = (c_1 s_{j-1} + c_2 s_{j-2} + \cdots + c_L s_{j-L}) \mod 2, \tag{4.1}$$

za vsak j, $j \geq L$. ■

Za ilustracijo delovanja LFSR-generatorja si oglejmo naslednji primer:

Primer. Naj bo dan generator LFSR$(4, 1 + x + x^4)$, prikazan na sliki 4.2.



Slika 4.2: Generator LFSR$(4, 1 + x + x^4)$.

Če je $[0,0,0,0]$ začetno stanje, potem je izhodno zaporedje zaporedje samih ničel. Vzemimo za začetno stanje $[0,1,1,0]$. V tabeli 4.1. so prikazane vsebine posameznih enot po t-tem koraku s tem začetnim stanjem.

t	D_3	D_2	D_1	D_0
0	0	1	1	0
1	0	0	1	1
2	1	0	0	1
3	0	1	0	0
4	0	0	1	0
5	0	0	0	1
6	1	0	0	0
7	1	1	0	0
8	1	1	1	0
9	1	1	1	1
10	0	1	1	1
11	1	0	1	1
12	0	1	0	1
13	1	0	1	0
14	1	1	0	1
15	0	1	1	0

Tabela 4.1: Stanja generatorja LFSR$(1 + x + x^4)$ z začetnim stanjem $[0, 1, 1, 0]$ po t-tem koraku.

Izhodno zaporedje je $s = 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, \ldots$ in je periodično s periodo 15. •

Za vsak generator LFSR(L,C(x)) z začetnim stanjem $s = [s_0, s_1, \ldots, s_{L-1}]$, lahko konstruiramo matriko A, tako da bo produkt $A \cdot s = [s_1, s_2, \ldots s_L]$. Torej je $A^i \cdot s = [s_i, s_{i+1}, \ldots, s_{i+L-1}]$. Matrika A ima obliko:

$$A = \begin{pmatrix} 0 & 1 & 0 & \ldots & 0 & 0 \\ 0 & 0 & 1 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 0 & 1 \\ c_L & c_{L-1} & c_{L-2} & \ldots & c_2 & c_1 \end{pmatrix}. \tag{4.2}$$

Primer. Naj bo $s = [1, 0, 1, 0]$ začetno stanje generatorja LFSR$(4, 1 + x^3 + x^4)$. Potem je ustrezna matrika A enaka

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}$$

in $s_4 = s_1 + s_0 = 1,\ s_5 = s_2 + s_1 = 1, \ldots$. •

Trditev 4.4. *Vsako izhodno zaporedje za vsa možna začetna stanja generatorja LFSR$(L, C(x))$ je periodično natanko tedaj, ko je povratni polinom $C(x)$ stopnje L, torej natanko tedaj, ko je generator LFSR nesingularen.*

Preden dokažemo zgornji izrek, si poglejmo, kako lahko definiramo periodo generatorja LFSR s pomočjo pripadajoče matrike A. Če ima izhodno zaporedje LFSR-generatorja periodo N, potem je $s_{N+i} = s_i$ za vsak $i \geq 0$. Če to zapišemo v matrično obliko, lahko hitro preverimo, da je to ekvivalentno enakosti $A^N \cdot \mathbf{s} = \mathbf{s}$, kjer je $\boldsymbol{s}$ začetno stanje.

Dokaz. Naj bo A matrika, ki pripada generatorju LFSR$(L, C(x))$. Recimo da je LFSR singularen, tj. $c_L = 0$. Potem matrika A ni obrnljiva, saj je prvi stolpec sestavljen iz samih ničel. Determinanta matrika A je torej 0 in zato ne obstaja tak N, da je $A^N \cdot \boldsymbol{s} = \boldsymbol{s}$, kjer je

$\boldsymbol{s} = [s_0, s_1, \ldots, s_{L-1}]$ začetno stanje danega LFSR-generatorja, oziroma $A^N = I$. Torej izhodno zaporedje LFSR-generatorja ni periodično. ■

V trditvi 4.4 smo za definicijo periode vzeli tak N, da je $s_{i+N} = s_i$ za vsak $i \geq 0$. Če je LFSR singularen, torej $c_L = 0$, potem se po določenih korakih ponovi stanje, ki pa ni začetno. V nadaljevanju bomo predpostavili, da so vsi LFSR-generatorji nesingularni. Preden formuliramo in dokažemo pomemben izrek o periodi zaporedij, generiranih z LFSR-generatorjem, moramo definirati nekaj algebraičnih pojmov in dokazati izreke, ki jih bomo potrebovali.

Definicija 4.5. *Nekonstanten polinom $f(x)$ nad obsegom $\mathbb{F}$, tj. $f(x) \in \mathbb{F}[x]$, je* **nerazcepen** *nad obsegom $\mathbb{F}$, če se ga ne da zapisati kot produkt dveh nekonstantnih polinomov iz $\mathbb{F}[x]$.*

Definirajmo sedaj naslednjo ekvivalenčno relacijo nad polinomi. Naj bodo $f(x), h(x)$, $g(x) \in \mathbb{F}[x]$ polinomi. Pravimo da sta polinoma $g(x)$ in $h(x)$ **kongruentna** po modulu $f(x)$, kadar $f(x)$ deli polinom $g(x) - h(x)$, tj.

$$g(x) \equiv h(x) \mod f(x) \Leftrightarrow f(x) \mid g(x) - h(x).$$

Z $\mathbb{F}[x]/(f(x))$ označimo množico ekvivalenčnih razredov polinomov iz $\mathbb{F}[x]$ stopnje manjše od $n = \deg(f(x))$.

Lema 4.6. *Naj bo $\mathbb{F}$ komutativen obseg (polje), $f(x) \in \mathbb{F}[x]$ polinom nad obsegom $\mathbb{F}$. Naj bo $\mathbb{F}[x]/(f(x))$ množica ekvivalenčnih razredov definirana zgoraj. Potem veljata naslednji trditvi.*

1. *$\mathbb{F}[x]/((f(x))$ je za seštevanje po komponentah in množenje po modulu $f(x)$ komutativen kolobar.*
2. *Če je polinom $f(x) \in \mathbb{F}[x]$ nerazcepen, je $\mathbb{F}[x]/(f(x))$ komutativen obseg.* ■

Leme 4.6 tu ne bomo dokazali. Dokaz uporablja kolobarje polinomov, ideale, praideale in je na voljo v [21].

Lema 4.7. *Naj bo $f \in \mathbb{F}_q[x]$ nerazcepen polinom nad končnim obsegom $\mathbb{F}_q$.*

1. *Naj bo α koren polinoma f v razširitvi obsega $\mathbb{F}_q$. Potem za polinom $h \in \mathbb{F}_q[x]$ velja, $h(\alpha) = 0$ natanko tedaj, ko f deli polinom h.*
2. *Naj bo polinom f stopnje m. Potem $f(x)$ deli $x^{q^n} - x$ natanko takrat, ko m deli n.*

Dokaz.

1. Naj bo a vodilni koeficient polinoma f in naj bo $g(x) = a^{-1}f(x)$. Potem je g moničen (tj. g ima vodilni koeficient 1) nerazcepen polinom v $\mathbb{F}_q[x]$. Ker je α koren polinoma $f(x)$, je $g(\alpha) = 0$. Ker je f nerazcepen nad $\mathbb{F}_q$, je g minimalni polinom za element α nad $\mathbb{F}_q$, tj. generator ideala $J = \{h \in \mathbb{F}_q[x],\ h(\alpha) = 0\}$. Po izreku 1.82 v [21, str.31] sledi, da za polinom $h(x) \in \mathbb{F}_q$ velja, $h(\alpha) = 0$ če in samo če polinom g deli h. Sledi, f deli g.

2. Naj $f(x)$ deli $x^{q^n} - x$. Naj bo α koren polinoma f v razpadnem obsegu polinoma f nad $\mathbb{F}_q$. Potem je $\alpha^{q^n} = \alpha$, torej $\alpha \in \mathbb{F}_{q^n}$. Sledi, $\mathbb{F}_q(\alpha)$ je podobseg obsega F_{q^n}. Ker je $[\mathbb{F}_q(\alpha) : \mathbb{F}_q] = m$ in $[\mathbb{F}_{q^n} : \mathbb{F}_q] = n$, sledi, da m deli n.
   Dokažimo še obrat. Naj m deli n. Potem $\mathbb{F}_{q^n}$ vsebuje $\mathbb{F}_{q^m}$ kot podobseg. Naj bo α koren polinoma f v razpadnem obsegu polinoma f nad $\mathbb{F}_q$. Potem je $[\mathbb{F}_q(\alpha) : \mathbb{F}_q] = m$ in tako $\mathbb{F}_q(\alpha) = \mathbb{F}_{q^m}$. Ker je $\alpha \in \mathbb{F}_{q^n}$, sledi $\alpha^{q^n} = \alpha$, in α je koren polinoma $x^{q^n} - x \in \mathbb{F}_q[x]$. Iz prvega dela sledi, da polinom $f(x)$ deli polinom $x^{q^n} - x$. ■

Izrek 4.8. *Naj bo f nerazcepen polinom iz kolobarja $\mathbb{F}_q[x]$ stopnje m. Potem je vsak koren α polinoma f element obsega $\mathbb{F}_{q^m}$. Še več, vsi koreni polinoma f so enostavni in so enaki m elementom $\alpha, \alpha^q, \alpha^{q^2}, \ldots, \alpha^{q^{m-1}}$ iz obsega $\mathbb{F}_{q^m}$, ki so paroma različni.*

Dokaz. Naj bo α koren polinoma f v razpadnem obsegu f nad obsegom $\mathbb{F}_q$. Potem je $[\mathbb{F}_q(\alpha) : \mathbb{F}_q] = m$, torej $\mathbb{F}_q(\alpha) = \mathbb{F}_{q^m}$ in $\alpha \in \mathbb{F}_{q^m}$. V nadaljevanju bomo pokazali, če je $\beta \in \mathbb{F}_{q^m}$ koren polinoma f, je β^q tudi koren polinoma f. Naj bo $f(x) = a_m x^m + \cdots + a_1 x + a_0$, kjer $a_i \in \mathbb{F}_q$ za vsak $0 \leq i \leq m$, in $f(\beta) = 0$. Potem velja

$$\begin{aligned} f(\beta^q) &= a_m \beta^{qm} + \cdots + a_1 \beta^q + a_0 \\ &= a_m^q \beta^{qm} + \cdots + a_1^q + a_0^q \\ &= (a_m \beta^m + \cdots + a_1 \beta + a_0)^q \\ &= f(\beta)^q \\ &= 0. \end{aligned}$$

Torej so elementi $\alpha, \alpha^q, \ldots, \alpha^{q^{m-1}}$ vsi koreni polinoma f. Ostane nam še dokaz, da so vsi paroma različni. Predpostavimo, da je $\alpha^{q^j} = \alpha^{q^k}$ za neka k in j, kjer je $0 \leq j, k \leq m-1$. Če obe strani pomnožimo z $\alpha^{q^{m-k}}$, dobimo

$$\alpha^{q^{m-k+j}} = \alpha^{q^m} = \alpha.$$

Po točki 1 leme 4.7 sledi, da $f(x)$ deli $x^{q^{m-k+j}} - x$. Po točki 2 leme 4.7 je to mogoče natanko tedaj, ko m deli $m - k + j$. Ker pa je $0 < m - k + j \leq m$, sledi $j = k$. ■

Lema 4.9. *Naj bo $f \in \mathbb{F}_q[x]$ polinom stopnje $m \geq 1$ in $f(0) \neq 0$. Potem obstaja tako pozitivno celo število $e \leq q^m - 1$, da polinom $f(x)$ deli $x^e - 1$.*

Dokaz. Kolobar $\mathbb{F}_q[x]/(f(x))$ vsebuje $q^m - 1$ neničelnih elementov. Elementi $x^j + (f(x))$ za $j = 0, 1, \ldots, q^m - 1$, so od nič različni, zato obstajata taki celi števili $0 \leq r < s \leq q^m - 1$, da je $x^s \equiv x^r \mod f(x)$. Ker sta si x in $f(x)$ tuja, sledi $x^{s-r} \equiv 1 \mod f(x)$, torej $f(x)$ deli polinom $x^{r-s} - 1$, kjer $0 < r - s \leq q^m - 1$. ■

Definicija 4.10. *Naj bo $f \in \mathbb{F}_q[x]$ neničelen polinom. Če je $f(0) \neq 0$, potem je najmanjše pozitivno celo število e, za katerega $f(x)$ deli polinom $x^e - 1$,* **red polinoma** *f in ga označimo z $\operatorname{ord}(f)$. Če je $f(0) = 0$, potem je $f(x) = x^h g(x)$, kjer je $h \in \mathbb{N}$, $g \in \mathbb{F}_q[x]$ in $g(0) \neq 0$. Red polinoma f je potem red polinoma g, tj. $\operatorname{ord}(f) = \operatorname{ord}(g)$.*

Izrek 4.11. *Naj bo $f \in \mathbb{F}_q[x]$ nerazcepen polinom nad poljem $\mathbb{F}_q$ stopnje m in $f(0) \neq 0$. Potem je $\operatorname{ord}(f)$ enak redu elementa x v multiplikativni grupi $\mathbb{F}_{q^m}^*$.*

Dokaz. Naj bo N red elementa x v multiplikativni grupi $\mathbb{F}_{q^m}^*$, oziroma polinom $f(x)$ deli $x^N - 1$. Preveriti moramo še, da je N res red polinoma $f(x)$. Če $f(x)$ deli polinom $x^k - 1$, kjer $k \leq N$, potem je $x^k \equiv 1 \mod f(x)$ oziroma je k red elementa x, multiplikativne grupe $\mathbb{F}_{q^m}^*$ in $k \geq N$. ■

Posledica 4.12. *Če je $f \in \mathbb{F}_q[x]$ nerazcepen polinom stopnje m nad $\mathbb{F}_q$, potem red polinoma f, tj. $\operatorname{ord}(f)$, deli $q^m - 1$.*

Dokaz. Če je $f(x) = cx$, kjer $c \in \mathbb{F}_q^*$, potem je $\operatorname{ord}(f) = 1$ in je trditev očitna. Sicer vemo, da je $\mathbb{F}_{q^m}^*$ grupa reda $q^m - 1$. Ker je $f(x)$ nerazcepen, je $f(0) \neq 0$. Po izreku 4.11 je $\operatorname{ord}(f)$ enak redu polinoma x v grupi $\mathbb{F}_{q^m}^*$. Red vsakega elementa pa deli red grupe in rezultat sledi. ■

Lema 4.13. *Naj bo c naravno število. Potem polinom $f(x) \in \mathbb{F}_q[x]$, $f(0) \neq 0$ deli polinom $x^c - 1$ natanko tedaj, ko red polinoma f, tj. $\operatorname{ord}(f)$, deli število c.*

Dokaz. Če $e = \mathrm{ord}(f)$ deli c, potem $f(x)$ deli polinom $x^e - 1$, ki deli polinom $x^c - 1$, torej $f(x)$ deli $x^c - 1$.

Dokažimo še obrat. Če $f(x)$ deli $x^c - 1$, velja $c \geq e$ in lahko zapišemo $c = me + r$, kjer je m pozitivno celo število in $0 \leq r < e$. Ker $x^c - 1 = (x^{me} - 1)x^r + (x^r - 1)$, sledi da $f(x)$ deli $x^r - 1$, kar pa je mogoče le, če $r = 0$. Torej e deli c. ■

Definicija 4.14. *Nerazcepen polinom $f(x)$ stopnje m nad obsegom $\mathbb{F}_q$ je* **primitiven**, *če je element x generator grupe $\mathbb{F}_{p^m}^*$, tj. multiplikativne grupe vseh neničelnih elementov $\mathbb{F}_{p^m} = \mathbb{F}_p[x]/(f(x))$.*

Naslednji izrek bo pomemben za LFSR-generator. Z njegovo pomočjo bomo karakterizirali periodo izhodnega zaporedja.

Izrek 4.15. *Nerazcepen polinom $f \in \mathbb{F}_q[x]$ stopnje m je primitiven natanko tedaj, ko je* $\mathrm{ord}(f) = q^m - 1$.

Dokaz. Ker je f primitiven, sledi da je element x generator grupe $\mathbb{F}_{q^m}^*$. Red elementa x je torej enak moči grupe, ki je $q^m - 1$. Iz izreka 4.11 sledi, da je red polinoma f enak $\mathrm{ord}(f) = q^m - 1$.

Dokažimo še obrat. Naj bo red polinoma f enak $\mathrm{ord}(f) = q^m - 1$. Po izreku 4.11 je red elementa x v multiplikativni grupi $\mathbb{F}_{q^m}^*$ enak $q^m - 1$. Moč grupe generirane z x je vsaj $q^m - 1$, saj red elementa deli moč grupe. Ker pa je moč multiplikativne grupe $\mathbb{F}_{q^m}^*$ enaka $q^m - 1$, je grupa generirana z x kar $\mathbb{F}_{q^m}^*$. ■

Definicija 4.16. *Naj bo $f(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0 \in \mathbb{F}_q[x]$ in $a_n \neq 0$.* **Recipročni polinom** *f^* polinoma f je definiran kot*

$$f^*(x) = x^n f\left(\frac{1}{x}\right) = a_0 x^n + a_1 x^{n-1} + \cdots + a_{n-1}x + a_n.$$

Izrek 4.17. *Naj bo f neničelen polinom v $\mathbb{F}_q[x]$ in f^* njegov recipročni polinom. Potem je* $\mathrm{ord}(f) = \mathrm{ord}(f^*)$.

Dokaz. Najprej si oglejmo primer, ko je $f(0) \neq 0$. V tem primeru izrek sledi iz dejstva, da $f(x)$ deli polinom $x^e - 1$ če in samo če $f^*(x)$ deli polinom $x^e - 1$. Če pa je $f(0) = 0$, potem zapišemo $f(x)$ kot $f(x) = x^h g(x)$, kjer $h \in \mathbb{N}$ in $g \in \mathbb{F}_q[x]$, ki zadošča $g(0) \neq 0$. Potem je $\mathrm{ord}(f) = \mathrm{ord}(g) = \mathrm{ord}(g^*) = \mathrm{ord}(f^*)$, kjer druga enakost velja po prvem delu, saj je $g^* = f^*$. ■

Zdaj se lahko vrnemo k LFSR-generatorju. Velja naslednji izrek, ki nam bo opisal obnašanje periode zaporedij, generiranih z LFSR-generatorji.

Izrek 4.18. *Naj bo $C(x) \in \mathbb{Z}_2[x]$ polinom stopnje L. Če je $C(x)$ nerazcepen nad $\mathbb{Z}_2$, potem vsak od $2^L - 1$ neničelnih začetnih stanj generatorja $LFSR(L, C(x))$ generira zaporedje s periodo, ki je enaka redu* $\mathrm{ord}(C(x))$ *povratnega polinoma $C(x)$.*

Dokaz. Karakteristični polinom matrike A je ravno recipročni polinom povratnega polinoma LFSR-generatorja. To lahko hitro preverimo z razvojem determinante $\det(A - xI)$ po zadnjem stolpcu. Povratni polinom LFSR-generatorja $C(x)$ je nerazcepen, zato je nerazcepen tudi njegov recipročni polinom $C(x)^*$. Torej je $C(x)^*$ karakterističen in ker je nerazcepen, tudi minimalen polinom matrike A. Red matrike A je tako najmanjše naravno število N, da je $A^N = I$. Iz linearne algebre vemo, da minimalen polinom matrike deli poljuben matrični polinom $f(X)$, za katerega velja, da je $f(A) = 0$. Oglejmo si zdaj polinom $x^N - 1$, kjer je N red matrike A. Ker je N red matrike A, je $A^N - 1 = 0$. Torej minimalni polinom matrike A, tj. $C(x)^*$ deli polinom

$x^N - 1$. Če bi $C(x)^*$ delil polinom $x^k - 1$ za $k < N$, potem bi bil red matrike A enak k in ne N. Torej recipročni polinom $C(x)^*$ ne deli nobenega polinoma oblike $x^k - 1$ za $k < N$. Red recipročnega polinoma $C(x)^*$ povratnega polinoma $C(x)$ je zato enak N. Po izreku 4.17 sledi, da je red polinoma $C(x)$ enak $\mathrm{ord}(C(x)) = \mathrm{ord}(C(x)^*) = N$. Perioda zaporedja, generiranega z generatorjem $\mathrm{LFSR}(L, C(x))$, je enaka redu matrike A, kar je razvidno iz konstrukcije matrike. ■

Izrek 4.18 motivira naslednjo definicijo.

Definicija 4.19. *Če je $C(x) \in \mathbb{Z}_2[x]$ primitiven polinom stopnje L, potem generatorju $LFSR(L, C(x))$ pravimo* **LFSR z maksimalno dolžino**, *izhodnim zaporedjem, ki jih le-ta generira, pa* **m-zaporedja**.

Naslednja trditev demonstrira statistične lastnosti LFSR-generatorjev z maksimalno dolžino, oziroma njihovih izhodnih m-zaporedij. Dokaza ne navajamo, ker statistične lastnosti niso glavni predmet naše analize. Bralec ga lahko najde v [27].

Trditev 4.20. *Naj bo $\boldsymbol{s}$ m-zaporedje, ki je generirano z generatorjem $LFSR(L, C(x))$ z maksimalno dolžino. Naj bo k celo število, $1 \leq k \leq L$, in naj bo $\bar{s}$ poljubno podzaporedje zaporedja $\boldsymbol{s}$ dolžine $2^L + k - 2$. Potem se vsako neničelno zaporedje dolžine k pojavi natanko 2^{L-K}-krat kot podzaporedje $\bar{s}$. Še več, ničelno podzaporedje dolžine k, se pojavi natanko $2^{L-k} - 1$-krat kot podzaporedje $\bar{s}$. Z drugimi besedami, porazdelitev vzorca s fiksno dolžino največ L je skoraj enakomerna.* ■

4.2 LFSR in linearna zahtevnost

V tem razdelku so podani osnovni rezultati glede linearne zahtevnosti zaporedja, kjer linearno zahtevnost interpretiramo kot dolžino najkrajšega LFSR-generatorja, ki generira zaporedje.

Definicija 4.21. *LFSR generira zaporedje s^∞, če obstajo začetno stanje, za katerega je izhodno zaporedje LFSR-generatorja enako zaporedju s. Podobno, LFSR generira zaporedje s^n, če obstaja začetno stanje, za katerega je prvih n členov izhodnega zaporedja LFSR-generatorja enakih s^n.*

Definicija 4.22. Linearna zahtevnost *zaporedja s^∞, ki jo označimo z $L(s)$, je definirana z naslednjim pravilom:*

1. *če je s ničelno zaporedje, tj. $s = 0, 0, \ldots$, potem je $L(s) = 0$;*
2. *če ne obstaja LFSR, ki generira s, potem je $L(s) = \infty$;*
3. *sicer je $L(s)$ dolžina najkrajšega LFSR-generatorja zaporedja s.*

Če je zaporedje s končno, tj. $s = s^n$, potem je $L(s^n)$ dolžina najkrajšega LFSR-generatorja, ki generira zaporedje s temi prvimi n členi.

Trditev 4.23. *Definiciji 3.13 za $L(s^n)$ in 4.22 za $L(s)$ sta v obsegu $GF(2)$ ekvivalentni.*

Dokaz. Naj bo s^n končno zaporedje dolžine n. V razdelku o linearni zahtevnosti smo $L(s^n)$ definirali kot najmanjše nenegativno število L', tako da obstajajo $c_1, c_2, \ldots, c_L \in GF(q)$, za katere je $s_j = c_1 s_{j-1} + \cdots + c_{L'} s_{j-L'}$ za vse $L' \leq j \leq n$. Naj bo $\mathrm{LFSR}(L, C(x))$ najkrajši LFSR-generator, ki generira zaporedje s^n. Potem je $s_j = c_1 s_{j-1} + \cdots + c_L s_{j-L}$ za vsak j, $j \geq L$. Po predpostavki so koeficienti c_i v obsegu $GF(2)$. Sledi $s_j + c_1 s_{j-1} + \cdots c_L s_{j-L} = 0$. Očitno velja $L' = L$ in trditev je dokazana. ■

Trditev 4.24. *Naj bo LFSR$(L, C(x))$ nesingularen in naj bo polinom $C(x)$ nerazcepen nad $\mathbb{Z}_2$. Potem vsako od $2^L - 1$ neničelnih začetnih stanj LFSR-generatorja generira zaporedje z linearno zahtevnostjo L.*

Dokaz. Naj bo $s_0, s_1, \ldots s_{L-1}$ poljubno neničelno začetno stanje LFSR$(L, C(x))$. Naj bo $s = s_0, s_1, \ldots s_L, s_{L+1}, \ldots$ izhodno zaporedje in $f_s(D)$ minimalen polinom zaporedja s. Očitno $f_s(D)$ deli polinom $C(x)$. Ker pa je $C(x)$ nerazcepen nad $\mathbb{Z}_2$, velja $f_s(D) = C(x)$. Torej je linearna zahtevnost vsakega zaporedja generiranega z LFSR$(L, C(x))$, kjer je $C(x)$ nerazcepen nad $\mathbb{Z}_2$, z neničelnim začetnim stanjem, enaka stopnji minimalnega polinoma, ki je L. ■

Posledica zgornjega dokaza je dejstvo, da je nerazcepen povratni polinom LFSR-generatorja tudi minimalen polinom za neničelna izhodna zaporedja generirana s tem LFSR-generatorjem. Naslednja trditev nam poda pričakovano linearno zahtevnost naključnega zaporedja. Najprej definirajmo naslednjo funkcijo.

Definicija 4.25. *Naj bo n naravno število. Potem $\boldsymbol{B(n)}$ označuje* **funkcijo parnosti**, *ki je definirana z naslednjim pravilom. $B(n) = 0$, če je n sodo število, $B(n) = 1$, če je n liho število.*

Trditev 4.26. *Naj bo s^n izbrano naključno izmed vseh binarnih zaporedij dolžine n. Naj bo $L(s^n)$ linearna zahtevnost zaporedja s^n. Potem velja.*

1. *Pričakovana linearna zahtevnost zaporedja s^n je*

$$E(L(s^n)) = \frac{n}{2} + \frac{4 + B(n)}{18} - \frac{1}{2^n}\left(\frac{n}{3} + \frac{2}{9}\right).$$

   Torej je za dovolj velike n, $E(L(s^n)) \approx n/2 + 2/9$ če je n sod, in $E(L(s^n)) \approx n/2 + 5/18$ če n lih.

2. *Varianca linearne zahtevnosti zaporedja s^n je*

$$Var(L(s^n)) = \frac{86}{81} - \frac{1}{2^n}\left(\frac{14 - B(n)}{27}n + \frac{82 - 2B(n)}{81}\right) - \frac{1}{2^{2n}}\left(\frac{1}{9}n^2 + \frac{4}{27}n + \frac{4}{81}\right).$$

   Torej je $Var(L(s^n)) \approx 86/81$ za dovolj velik n.

■

Trditve ne bomo dokazali. Dokaz je tehnične narave in je na voljo v [27].

4.3 Berlekamp-Masseyjev algoritem

V tem razdelku je prikazan algoritem, ki za dano končno zaporedje konstruira najkrajši LFSR-generator, ki to zaporedje generira. S tem algoritmom lahko računamo tudi linearno zahtevnost zaporedja, uporaben pa je tudi pri dekodiranju BCH-kod za odpravljanje napak, glej [23]. Ključno vlogo v algoritmu bo imel naslednji izrek.

Izrek 4.27. *Če nek LFSR-generator dolžine L, generira zaporedje $s_0, s_1, \ldots, s_{N-1}$, ne generira pa zaporedja $s_0, s_1, \ldots s_{N-1}, s_N$, potem ima vsak LFSR-generator, zaporedja $s_0, \ldots, s_N$, dolžino L', ki zadošča pogoju*

$$L' \geq N + 1 - L. \tag{4.3}$$

Dokaz. Za $L \geq N$ je izrek trivialen, zato privzemimo, da je $L < N$. Naj bodo $c_1, c_2, \ldots, c_L$ koeficienti povratnega polinoma LFSR-generatorja, ki generira zaporedje $s_0, s_1, \ldots, s_{N-1}$, ne generira pa zaporedja $s_0, s_1, \ldots s_{N-1}, s_N$. Naj bodo $c_1^{'}, c_2^{'}, \ldots, c_{L'}^{'}$ koeficienti povratnega polinoma LFSR-generatorja zaporedja $s_0, \ldots, s_N$. Recimo, da je $L^{'} \leq N - L$. Po predpostavki velja

$$\begin{aligned} \sum_{i=1}^{L} c_i s_{j-i} &= s_j, \ j = L, L+1, \ldots, N-1, \\ \sum_{i=1}^{L} c_i s_{j-1} &\neq s_N, \ j = N, \end{aligned} \tag{4.4}$$

in

$$\sum_{k=1}^{L^{'}} c_k^{'} s_{j-k} = s_j, \ j = L^{'}, L^{'}+1, \ldots, N. \tag{4.5}$$

Če sedaj združimo (4.4) in (4.5), dobimo

$$\sum_{i=1}^{L} c_i s_{N-i} = \sum_{i=1}^{L} c_i \sum_{k=1}^{L^{'}} c_k^{'} s_{N-i-k}, \tag{4.6}$$

kjer je pisava na levi strani enakosti upravičena, saj je množica $\{s_{N-L}, s_{N-L+1}, \ldots, s_{N-1}\}$ podmnožica množice $\{s_{L'}, s_{L'+1}, \ldots, s_{N-1}\}$. Če v vsoti (4.6) zamenjamo vrstni red seštevanja, dobimo

$$\begin{aligned} \sum_{i=1}^{L} c_i s_{N-i} &= \sum_{k=1}^{L^{'}} c_k^{'} \sum_{i=1}^{L} c_i s_{N-k-i} \\ &= \sum_{k=1}^{L^{'}} c_k^{'} s_{N-k} \\ &= s_N \end{aligned} \tag{4.7}$$

kjer smo zaporedoma uporabili (4.4) in (4.5). Uporaba (4.4) je upravičena z dejstvom, da je $\{s_{N-L'}, s_{N-L'+1}, \ldots, s_{N-1}\}$ podmnožica $\{s_L, s_{L+1}, \ldots, s_{N-1}\}$. Če si zdaj ogledamo rezultat (4.7), vidimo, da je v protislovju s predpostavko (4.4), iz česar sledi, da je začetna predpostavka $L^{'} \leq N - L$ napačna. Sledi torej $L^{'} \geq N + 1 - L$, s čimer je izrek dokazan. ■

Naj bo s^{∞} zaporedje in naj bo $L_N(s)$ linearna zahtevnost podzaporedja $s_0, s_1, \ldots, s_{N-1}$. Po prejšnjem izreku je $L_N(s) \leq N$. Še več, linearna zahtevnost $L_N(s)$ kot funkcija N mora biti monotono nepadajoča.

Lema 4.28. *Če nek LFSR-generator dolžine $L_N(s)$ generira zaporedje $s_0, s_1, \ldots, s_{N-1}$ in ne $s_0, s_1, \ldots, s_{N-1}$, s_N, potem velja*

$$L_{N+1}(s) \geq \max\{L_N(s), N + 1 - L_N(s)\}.$$

Dokaz. Iz monotonosti $L_N(s)$ sledi $L_{N+1}(s) \geq L_N(s)$. Iz predpostavk in izreka 4.27 sledi $L_{N+1}(s) \geq N + 1 - L_N(s)$. ■

Lemo 4.28 bomo uporabili v nadaljevanju, ko bomo dokazovali minimalnost dolžine pomičnega registra dobljenega z algoritmom za sintezo LFSR-generatorja. Posledica bo dokaz, da lahko neenakost v lemi 4.28 nadomestimo z enakostjo.

Algoritem za sintezo LFSR-generatorja

V tem razdelku podamo konstrukcijo algoritma za generiranje LFSR-generatorja dolžine $L_N(s)$, ki generira zaporedje $s_0, s_1, s_2, \ldots, s_{N-1}$ za $N = 1, 2, 3, \ldots$. Znana posledica definicije linearne zahtevnosti je dejstvo, da je $L_{N+1}(s) = N + 1$ natanko tedaj, ko so $s_0, s_1, \ldots, s_{N-1}$ vsi enaki nič in $s_N \neq 0$. V tem primeru v lemi 4.28 velja enakost. Naslednji izrek je osnova algoritma. Njegov dokaz bo konstruktiven in bo hkrati nakazal potek algoritma za sintezo najkrajšega LFSR-generatorja, ki generira zaporedje $s_0, s_1, \ldots, s_{n-1}$.

Izrek 4.29. *Naj bo $L_N(s)$ linearna zahtevnost zaporedja $s_0, s_1, \ldots, s_{N-1}$. Če nek LFSR-generator dolžine $L_N(s)$ zaporedja $s_0, s_1, \ldots, s_{N-1}$ generira tudi $s_0, s_1, \ldots, s_{N-1}, s_N$, potem je $L_{N+1}(s) = L_N(s)$. Če pa je $L_N(s)$ dolžina LFSR-generatorja zaporedja $s_0, s_1, \ldots, s_{N-1}$, ki ne generira zaporedja $s_0, s_1, \ldots, s_{N-1}, s_N$, potem*

$$L_{N+1}(s) = \max\{L_N(s), N + 1 - L_N(s)\}. \tag{4.8}$$

Dokaz. Prvi del trditve sledi iz definicije 4.22 linearne zahtevnosti, zato dokažimo enakost (4.8). Naj bo s dano zaporedje in naj

$$C^{(N)}(x) = 1 + c_1^{(N)}x + \cdots + c_{L_N(s)}^{(N)}x^{L_N(s)} \tag{4.9}$$

označuje povratni polinom najkrajšega LFSR-generatorja zaporedja $s_0, s_1, \ldots, s_{N-1}$. Dolžina takega LFSR-generatorja je torej $L_N(s)$. V primeru $N = 1$ je postopek preprost, enakost (4.8) pa sledi iz definicije 4.22 linearne zahtevnosti. Predpostavimo da smo našli $L_N(s)$ in nek $C^{(N)}(x)$ za $N = 1, 2, \ldots, n$, da velja enakost v lemi 4.28 za $N = 1, 2, \ldots, n-1$. Potem želimo najti $L_{n+1}(s)$ in nek polinom $C^{(n+1)}(x)$ in pokazati, da v lemi 4.28 velja enakost za $N = n$. Po predpostavki sledi

$$s_j + \sum_{i=1}^{L_n(s)} c_i^{(n)} s_{j-i} = \begin{cases} 0, & j = L_n(s), \ldots, n-1 \\ d_n, & j = n \end{cases} \tag{4.10}$$

kjer je d_n razlika med s_n in $(n+1)$ bitom generiranim z najkrajšim LFSR-generatorjem, ki generira prvih n bitov zaporedja s. Imenujemo jo *naslednja razlika* (angl. next disrepancy). Če je $d_n = 0$, potem ta isti LFSR-generator generira tudi prvih $n+1$ bitov zaporedja s in tako je $L_{n+1}(s) = L_n(s)$, in $C^{(n+1)}(x) = C^{(n)}(x)$.

Če pa je $d_n \neq 0$, potem moramo najti nov LFSR-generator, ki bo generiral prvih $n+1$ bitov zaporedja s. Naj bo m dolžina zaporedja, pred zadnjo spremembo v minimalni dolžini registra, tj.

$$L_m(s) < L_n(s), \quad L_{m+1} = L_n(s). \tag{4.11}$$

Zaradi zahteve po spremembi dolžine, LFSR-generator s povratnim polinomom $C^{(m)}(x)$ in dolžino $L_m(s)$ ne more generirati $s_0, s_1, \ldots, s_{m-1}, s_m$. Torej velja

$$s_j + \sum_{i=1}^{L_m(s)} c_i^{(m)} s_{j-i} = \begin{cases} 0, & j = L_m(s), \ldots, m-1 \\ d_m \neq 0, & j = m. \end{cases} \tag{4.12}$$

Po predpostavki v lemi 4.28 velja enakost za $N = m$, tako da je

$$L_{m+1} = L_n(s) = \max\{L_m(s), m + 1 - L_m(s)\}$$

in če pri tem upoštevamo še (4.11), dobimo

$$L_n(s) = m + 1 - L_m(s). \tag{4.13}$$

Zdaj trdimo, da je povratni polinom

$$C(x) = C^{(n)}(x) - d_n d_m^{-1} x^{n-m} C^{(m)}(x) \tag{4.14}$$

pravilna izbira za $C^{(n+1)}(x)$. Hitro lahko opazimo, da je stopnja polinoma $C(x)$ največ

$$\max\{L_n(s), n-m+L_m(s)\} = \max\{L_n(s), n+1-L_n(s)\},$$

kjer enakost sledi iz (4.13). Torej je $C(x)$ povratni polinom za LFSR-generator dolžine L, kjer je

$$L = \max\{L_n(s), n+1-L_n(s)\}. \tag{4.15}$$

Iz definicije polinoma (4.14) sledi

$$\begin{aligned} s_j + \sum_{i=1}^{L} c_i s_{j-i} &= s_j + \sum_{i=1}^{L_n(s)} c_i^{(n)} s_{j-i} - d_n d_m^{-1} \cdot \left[s_{j-n+m} + \sum_{i=1}^{L_m(s)} c_i^{(m)} s_{j-n+m-i} \right] \\ &= \begin{cases} 0, & j = L, L+1, \ldots, n-1 \\ d_n - d_n d_m^{-1} d_m = 0, & j = n \end{cases} \end{aligned}$$

kjer je zadnja enakost posledica uporabe (4.10) in (4.12). Torej sledi, da LFSR-generator dolžine L s povratnim polinomom $C(x)$ generira $n+1$ bitov $s_0, s_1, \ldots, s_n$. Ker L v (4.15) zadostuje lemi 4.28 z enakostjo, velja $L = L_n(s)$, zato je enakost v lemi 4.28 vedno dosežena. Izrek je s tem dokazan. ■

Algoritem 4.1. *Algoritem za sintezo najkrajšega LFSR-generatorja.*

1. $1 \to C(x)$; $1 \to B(x)$; $1 \to D$;
   $0 \to L$; $1 \to b$; $0 \to N$;
2. **if** $N = n$ **then stop else**
   $$d = s_N + \sum_{i=1}^{L} c_i s_{N-i}.$$
3. **if** $d = 0$ **then** $D+1 \to D$ **and goto** *6.*
4. **if** $d \neq 0$ **and** $2L > N$ **then**
   $C(x) - db^{-1}x^D B(x) \to C(x)$;
   $D+1 \to D$;
   goto *6.*
5. **if** $d \neq 0$ **and** $2L \leq N$ **then**
   $C(x) \to T(x)$;
   $C(x) - db^{-1}x^D B(x) \to C(x)$;
   $N+1-L \to L$;
   $T(x) \to B(x)$;
   $d \to b$;
   $1 \to D$;
6. $N+1 \to N$ **and goto** *2.*

Za vsak n, ko je $N = n$ in je bil dosežen korak 2, so elementi v algoritmu v naslednji relaciji z elementi, ki se pojavijo v dokazu izreka 4.29:

$$\begin{aligned} C(x) &= C^{(n)}(x) \\ L &= L_n(s) \\ D &= n - m \\ d &= d_n, \text{ pod predpostavko, da je bil narejen izračun v koraku 2,} \\ B(x) &= C^{(m)}(x) \\ b &= d_m. \end{aligned}$$

Da algoritem res sledi konstrukciji dokaza izreka 4.29 bi moralo biti očitno, z dvema izjemama. Korak 5 je izveden natanko tedaj, ko je po izreku 4.29 potrebna sprememba dolžine LFSR-generatorja. V tem primeru bo trenutni polinom $C(x)$ za zaporedne iteracije zadnji povratni polinom pred zadnjo spremembo dolžine in zato postane novi $B(x) = C^{(m)}(x)$. Druga izjema je v koraku 2. Recimo da dobimo prvi neničelni d v koraku 2 z $N = k$. Sledi $s_0 = s_1 = \cdots = s_{k-1} = 0$ in $s_k \neq 0$. V tem trenutku je $L = L_k(s) = 0$ in zato je dolžina zaporedja pred zadnjo spremembo dolžine LFSR-generatorja, nedefinirana, saj noben LFSR ne more imeti dolžine manjše od nič. Torej pravila (4.14) ne moremo uporabiti. V tem primeru inicializacija v koraku 1 povzroči izvedbo koraka 5, katerega rezultat je $C(x) = C^{(k+1)}(x) = 1 - dx^{k+1}$ in $L = L_{k+1}(s) = k + 1$. Že prej smo pokazali, da je vsak LFSR dolžine $k + 1$, veljavna rešitev za tak primer.

Do sedaj smo se ukvarjali z iskanjem samo enega LFSR-generatorja z minimalno dolžino za dano zaporedje. S pomočjo algoritma za sintezo LFSR-generatorja pa lahko najdemo množico vseh LFSR-generatorjev z minimalno dolžino $L_n(s)$ za zaporedje $s_0, s_1, \ldots, s_{n-1}$. Velja naslednji izrek.

Izrek 4.30. *Naj bo $s_0, s_1, \ldots, s_{n-1}$ zaporedje na katerem uporabimo algoritem za sintezo LFSR-generatorja. Naj L, C(x), D, in B(x) označujejo vrednosti, ko se algoritem ustavi. Če je $2L \leq n$, potem je $C(x)$ povratni polinom enolično določenega LFSR-generatorja z minimalno dolžino L, ki generira dano zaporedje. Če pa je $2L > n$, potem je množica polinomov*

$$\{C(x) + Q(x)x^D B(x) \ ; \ \deg(Q(x)) < 2L - n\}$$

množica vseh povratnih polinomov za vse LFSR-generatorje danega zaporedja, z minimalno dolžino L.

Skica dokaza. Iz izreka 4.29 sledi, da ko je nek LFSR-generator dolžine $L_N(s)$ za zaporedje $s_0, s_1, \ldots, s_{N-1}$ in ne za zaporedje $s_0, s_1, \ldots, s_{N-1}, s_N$, potem pride do spremembe dolžine ($L_{N+1}(s) > L_N(s)$) natanko tedaj, ko je $2L_N(s) \leq N$. Sledi torej, da je minimalna dolžina LFSR-generatorja enolično določena natanko tedaj, ko je $2L_N(s) \leq N$. Torej, ko se algoritem ustavi pri $2L > n$, LFSR-generator, ki je rezultat algoritma, ni enoličen. V tem primeru bi bil LFSR-generator enolična rešitev le, če bi dodatno določili bite $s_n, s_{n+1}, \ldots, s_{2L-1}$ v dogovoru z izhodnim zaporedjem tega LFSR-generatorja. Z vsako tako določitvijo $2L - n$ dodatnih bitov, bi bila uporabljena samo koraka 3 in 4 algoritma za konstrukcijo novega povratnega polinoma. Torej vzorec $2L - n$ naslednjih razlik služi samo za določitev večkratnika nespreminjajočega se polinoma *B(x)*, ki je na koncu dodan, tako da dobimo končni rezultat. ■

Posledica 4.31. *Če je $2L_n(s) < n$, potem algoritem za sintezo LFSR-generatorja konstruira enoličen LFSR z minimalno dolžino L, tj. $L = L_n(s)$ in $C(x) = C^{(n)}(x)$, ko je $N = 2L_n(s)$ v koraku 2, tj. ko uporabi prvih $2L_n(s)$ bitov zaporedja.* ■

Poglavje 5

URNO-KONTROLIRANI POMIČNI REGISTRI

LFSR-generatorji generirajo zaporedja z dobrimi statističnimi lastnostmi in veliko periodo, kljub temu pa niso uporabni kot samostojni generatorji toka ključev v tokovnih šifrah, saj so zaradi linearnosti dovzetni za napade. Za povečanje linearne zahtevnosti, lahko uporabimo nelinearno funkcijo na izhodnem zaporedju ali vzporedne kombinacije različnih oziroma enakih LFSR-generatorjev.

Alternativna tehnika za doseganje večje varnosti pomičnih registrov je, da jih kontroliramo z uro. LFSR-generatorji so kontrolirani regularno, kar pomeni, da je izhodno zaporedje pomičnega registra sestavljeno iz vseh vrednosti tega registra. Nekateri registri pa so lahko kontrolirani neregularno s pomočjo drugih registrov, tako da napadi, ki temeljijo na regularnosti registrov, niso uspešni. Predhodniki takšnih sistemov so bili razni šifrirni stroji, ki so temeljili na rotorjih, kot na primer *Siemens T52* ali *Lorenz SZ40*, glej [6, 15]. Drug način kontroliranja registra so *kaskade*, kjer so vsi registri, razen prvega, kontrolirani s svojimi predhodniki. Take sisteme lahko gledamo tudi kot naravne naslednike zgodnjih šifrirnih strojev.

V nadaljevanju bomo v prvem razdelku definirali CCSR in razvili algebraične metode, ki jih potrebujemo pri analizi CCSR-generatorja. Nato si bomo ogledali linearno zahtevnost in periodo zaporedij generiranih s CCSR-generatorjem. Rezultate bomo primerjali z rezultati dobljenimi pri LFSR-generatorjih. V razdelku 2 si bomo ogledali različne sisteme, ki temeljijo na kaskadah (sestavljankah), nato pa bomo v razdelku 3 predstavili še nekatere druge CCSR-generatorje, kot na primer *stop-and-go* generator, samo-kontroliran pomični register in alternirajoči koračni generator, kjer en register določa, kateri od ostalih dveh bo reguliran. Na koncu bomo vpeljali še oznake, ki jih bomo potrebovali pri analizi varnosti v šestem in sedmem poglavju.

5.1 Osnovni sistemi in njihove lastnosti

Sistemi, ki jih bomo najprej preučevali bodo urno kontrolirani pomični registri (angl. Clock-Controlled Shift Register - CCSR)), kjer bodo registri kontrolirani z drugimi registri. Za začetek si oglejmo preprost primer.

Primer. Naj bosta dana dva regularno kontrolirana pomična registra A in B, ki zaporedoma generirata zaporedji celih števil $\{a_t\}$ in $\{b(t)\}$. Generator toka ključev bo sestavljen iz teh registrov, tok ključev pa označimo z $\{u(t)\}$. Naj bosta zaporedji $\{a_t\}$ in $\{b(t)\}$ naslednji:

$$\begin{aligned} \{a_t\}_{t=0}^{\infty} &= 0, 3, 1, 4, 5, 1, 1, \ldots \\ \{b(t)\}_{t=0}^{\infty} &= 1, 3, 5, 9, 6, 7, 2, 1, 8, 9, 4, 3, 2, 4, 5, 8, 7, 3, 4, \ldots . \end{aligned}$$

Če register A kontrolira register B, potem je izhodno zaporedje $\{u(t)\}$ tako sestavljenega generatorja toka ključev naslednje:

$$\{u(t)\}_{t=0}^{\infty} = 1, 9, 6, 8, 4, 7, 3 \ldots .$$

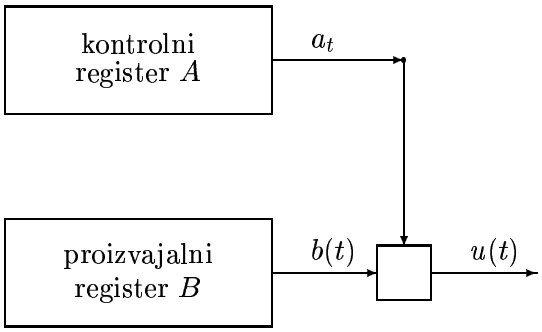
•

Definicija 5.1. Osnovni sistem, *ki generira zaporedje* $\{u(t)\}$, *je sestavljen iz dveh pomičnih registrov,* **kontrolnega registra** *(angl. Control Register - C-register) in* **proizvajalnega registra** *(angl. Generating Register - G-register). C-register in G-register generirata zaporedoma zaporedji celih števil* $\{a_t\}$ *in* $\{b(t)\}$. *Členi zaporedja* $\{u(t)\}$ *so definirani z enačbo*

$$u(t) = b\left(\sum_{k=0}^{t} a_k\right) \quad za\ t > 0, \tag{5.1}$$

kjer je začetno stanje $u(0) = b(a_0)$. *Pri tem moramo na vsakem koraku* k *poznati vsoto prvih* k *členov zaporedja* a_t, *ki jo označimo s* $\sigma(k) = \sum_{i=0}^{k} a_i$. *Tako lahko zgornjo enačbo zapišemo v obliki* $u(t) = b(\sigma(t))$.

Delovanje osnovnega sistema pri koraku t lahko opišemo tudi takole. Ko je osnovni sistem generiral vrednost $u(t-1)$, C-register generira nenegativno celo število a_t. Nato osnovni sistem naredi a_t korakov, preden generira naslednjo vrednost $u(t)$. Nazadnje C-register naredi še en korak, s katerim generira naslednjo vrednost a_{t+1}. Izhodno zaporedje $\{u(t)\}$ osnovnega sistema je podzaporedje zaporedja $\{b(t)\}$, ki ga generira G-register. Takim sistemom pravimo tudi **urno-kontrolirani pomični registri**, saj kontrolni register lahko nadomestimo z uro. Vsoto $\sigma(t)$ tako lahko interpretiramo tudi kot čas delovanja urno-kontroliranega pomičnega registra od inicializacije do trenutnega koraka. Delovanje osnovnega sistema je prikazano na sliki 5.1.



Slika 5.1: Shema osnovnega sistema.

Primer. Predpostavimo da je C-register periodičen s periodo M, tako da je $a_{t+M} = a_t$ za vsak $t \geq 0$. Naj bosta $\sigma(t) = \sum_{k=0}^{t} a_k$ in $S = \sum_{k=0}^{M-1} a_k$. Ker je $\sigma(t+M) = \sigma(t) + S$ za vsak $t \geq 0$, velja naslednja enakost:

$$u(i + jM) = b(jS + \sigma(i)),\ 0 \leq i < M,\ j \geq 0. \tag{5.2}$$

•

Oglejmo si nekaj posebnih primerov osnovnih sistemov.

1. Če a_t zavzame samo vrednosti 0 in 1, potem imamo tako imenovani *stop-and-go* generator.
2. Kontrolni register ni nujno sestavljen iz samo enega registra, ampak je lahko kaskada (sestavljanka) časovno-reguliranih registrov, kot bomo to videli kasneje.

Preden nadaljujemo z analizo osnovnih sistemov, ponovimo nekaj znanih pojmov, in lastnosti zaporedij. Z njimi bomo dokazali izrek, ki ga bomo potrebovali pri kasnejši analizi. Spomnimo se rodovne funkcije zaporedja $\{s(t)\}$, ki je potenčna vrsta $G(x) = \sum_{t=0}^{\infty} s(t)x^t$. V 3. poglavju smo pokazali, da lahko rodovno funkcijo periodičnega zaporedja s periodo N zapišemo v obliki

$$G(x) = \frac{\Phi(x)}{(1-x^N)},$$

kjer je $\Phi(x) = \sum_{t=0}^{N-1} s(t)x^t$. V posebnih primerih lahko rodovno funkcijo zapišemo tudi v racionalni obliki

$$G(x) = \frac{\phi(x)}{g(x)}, \tag{5.3}$$

kjer sta ϕ in g polinoma, $g(0) = 1$ in stopnja polinoma ϕ je manjša od stopnje polinoma g. V primeru če je zaporedje $\{s(t)\}$ generirano z LFSR-generatorjem dolžine n, potem lahko rodovno funkcijo zapišemo v obliki (5.3), kjer so koeficienti polinoma $g(x)$ določeni s povratnim polinomom, polinom $g(x)$ pa je stopnje n. Velja pa tudi obratno, vsako zaporedje, ki ima rodovno funkcijo oblike (5.3), lahko generiramo z LFSR-generatorjem, določenim s povratnim polinomom, ki je enak imenovalcu v racionalni obliki rodovne funkcije zaporedja.

Izrek 5.2. *Naj bo* $\boldsymbol{s}$ *neničelno zaporedje generirano s polinomom* g. *Minimum stopenj nerazcepnih faktorjev polinoma* g *je spodnja meja za linearno zahtevnost zaporedja* $\boldsymbol{s}$. *V posebnem primeru, če je* g *nerazcepen, je linearna zahtevnost neničelnega zaporedja* $\boldsymbol{s}$ *generiranega z* g *enaka stopnji polinoma* g.

Dokaz. Če je polinom g nerazcepen, potem iz izreka 3.17 sledi, da je g v rodovni funkciji (5.3) minimalni polinom. Upoštevajoč definicijo linearne zahtevnosti, sledi da je linearna zahtevnost enaka stopnji polinoma g. Če pa g ni nerazcepen, potem je $g(x) = p_1(x)p_2(x)\cdots p_k(x)$, kjer so $p_i(x)$ nerazcepni faktorji za vsak $i \leq i \leq k$. Racionalno obliko (5.3) krajšamo, dokler ne velja $\gcd(g(x), \phi(x)) = 1$. Potem je linearna zahtevnost zaporedja enaka stopnji imenovalca in je večja ali enaka minimalni stopnji nerazcepnih faktorjev polinoma g. ■

Vrnimo se zdaj k osnovnim sistemom. V nadaljevanju bomo predpostavili, da je zaporedje $\{a_t\}$ periodični s periodo M. Poiščimo polinome, ki generirajo zaporedje $\{u(t)\}$. Če začnemo pri i-tem členu tega zaporedja za nek i, $0 \leq i \leq M$, in izberemo vsak M-ti element zaporedja $u(t)$, potem iz (5.2) sledi, da je ta konstrukcija ekvivalentna naslednji. Če začnemo pri $t = \sigma(i)$-tem členu zaporedja $\{b(j)\}$ in nato izbiramo vsak S-ti element zaporedja $\{b(j)\}$, kjer je $S = \sigma(M-1)$. Takim zaporedjem bomo rekli **S-koračna podzaporedja** (angl. S-decimation sequence) zaporedja $\{b(t)\}$. Velja naslednji izrek.

Izrek 5.3. *Vsa* S*-koračna podzaporedja so lahko generirana z istim polinomom* $f_S(x)$, *katerega koreni so* S*-te potence korenov polinoma* $f(x)$, *povratnega polinoma za zaporedje* $\{b(j)\}$. ■

Dokaz zgornje trditve uporablja teorijo končnih obsegov in ga lahko najdemo npr. v [21, str.285-287].

Izhodno zaporedje $\{u(t)\}$ je sestavljeno iz M takih prepletenih zaporedij. Z drugimi besedami, če $\{u(t)\}$ zapišemo po vrsticah v tabelo sestavljeno iz M-tih stolpcev, potem je vsak

stolpec zaporedje, generirano s polinomom $f_S(x)$. Torej je zaporedje $\{u(t)\}$ lahko generirano z LFSR-generatorjem, sestavljenim iz povratnega polinoma $f_S(x)$, pri katerem vsako zakasnitev zamenjamo z verigo M zakasnitev. Tako je $\{u(t)\}$ generiran s povratnim polinomom $f_S(x^M)$. Vpeljimo zdaj naslednja pogoja:
P1: $f(x)$ je nerazcepen, stopnje n in reda N,
P2: S in N sta tuji si števili, tj. $\gcd(S, N) = 1$.
Za polinom $f_S(x)$ velja, da je tako kot $f(x)$ nerazcepen stopnje n in reda N, glej [3].

Primer. V nekaterih primerih je pogoj **P2** očitno izpolnjen. Vzemimo na primer stop-and-go sistem, kjer je C-register LFSR dolžine m s primitivnim povratnim polinomom stopnje m in periodo $2^m - 1$. Potem vsaka enota C-registra dobi $2^m/2$-krat v periodi vrednost 1. Če je a_t vsebina enote, potem iz $S = \sum_{k=0}^{2^m-2} a_k$ sledi, da je $S = 2^m/2$, torej je pogoju **P2** zadoščeno, če je N liho število. Alternativno, če je N praštevilo, potem S ne sme biti večkratnik števila N, da bi bilo pogoju **P2** zadoščeno. •

Naslednji izrek bo podal vrednost najmanjše stopnje nerazcepnega faktorja polinoma $g(x^M)$. Skupaj z izrekom 5.2 lahko tako določimo linearno zahtevnost poljubnega neničelnega zaporedja generiranega s polinomom $g(x^M)$. Ker je bil namen urno-kontroliranega pomičnega registra večja linearna zahtevnost izhodnega zaporedja, bomo izrek v celoti dokazali.

Izrek 5.4. *Naj bo $g(x)$ binaren nerazcepen polinom stopnje n in reda N. Potem je $\delta = (2^n - 1)/N$ celo število. Naj bo M liho pozitivno število in $M_\delta = M/\gcd(M, \delta)$. Naj bo $M_\delta = AB$ faktorizacija, kjer je A produkt praštevilskih faktorjev M_δ (ne nujno različnih), ki so tudi faktorji števila N. Potem je najnižja možna stopnja poljubnega nerazcepnega faktorja polinoma $g(x^M)$ enaka nA. Velja še več, obstaja vsaj en nerazcepen faktor te stopnje.*

Dokaz. Začnimo s preprostim rezultatom. Če je $g(x)$ binaren polinom, velja

$$g(\beta) = 0 \Longleftrightarrow g(\beta^2) = 0, \tag{5.4}$$

kjer je β poljuben element v poljubni razširitvi binarnega obsega. To je res, ker je

$$(g(x))^2 = \left(\sum g_i x^i\right)^2 = \sum g_i^2 x^{2i} = \sum g_i x^{2i} = g(x^2).$$

Druga enakost velja, ker so vsi faktorji števila 2 v križnem produkt ekvivalentni 0 v razširitvi binarnega obsega. Tretja enakost velja, ker je $g_i^2 = g_i$ za $g_i = 1$ ali $g_i = 0$.

Oglejmo si zdaj stopnje nerazcepnih faktorjev polinoma $\bar{g}(x) = g(x^M)$, kjer je $g(x)$ nerazcepen binaren polinom stopnje n in reda N. Naj bo δ celo število, tako da je $N\delta = 2^n - 1$ (posledica 4.12). Če je g primitiven, je $\delta = 1$. Predpostavili smo, da je M liho pozitivno celo število.

Tehnika, ki jo bomo uporabili je pregled velikosti *orbit* korenov polinoma $\bar{g}$, kjer orbite definiramo takole. Če je γ koren $\bar{g}$, potem je tudi koren nekega nerazcepnega faktorja $h(x)$ polinoma $\bar{g}$, stopnje recimo p. S ponavljanjem kvadriranja γ dobimo γ^τ, kjer je $\tau = 1, 2, 4, 8, \ldots, 2^{p-1}$, ki pa so izreku 4.8 ravno vsi koreni polinoma $h(x)$. Naslednje kvadriranje zopet da γ. Tako zaporedje rezultatov kvadriranj bomo imenovali **orbite**. Stopnjo najmanjšega nerazcepnega faktorja polinoma $\bar{g}$ lahko dobimo tako, da poiščemo velikost najmanjše orbite generirane s podmnožico svojih korenov. Koreni $g(x)$ so dani z različnimi vrednostmi

$$\beta^\tau, \ \tau = 1, 2, 4, 8, \ldots, 2^{n-1}, \tag{5.5}$$

kjer je β poljuben izbran koren.

Ker je po predpostavki NM liho število lahko pokažemo, da v ustrezni razširitvi binarnega obsega obstaja primitivni MN-ti koren enote, ω, tj. generator ciklične grupe $E^{(n)}$ korenov

polinoma $x^n - 1$ v razpadnem obsegu polinoma $x^n - 1$, glej [21, str.60]. Elementi ω^{kM}, $k = 0, 1, 2, \ldots, N-1$, so koreni polinoma $(x^N - 1)$, in so vsi različni, saj bi drugače ω ne bil primitiven MN-ti koren enote. Ker ima ta polinom kvečjemu N korenov, so ti dobljeni s temi elementi. Dalje opazimo, da je $g(x)$ faktor polinoma $(x^N - 1)$. Torej, če je β izbran koren polinoma $g(x)$, potem za nek K velja

$$\beta = \omega^{KM}. \tag{5.6}$$

Vrednost K je fiksirana, vendar o njej ni treba vedeti ničesar drugega, razen da sta si K in N tuja, saj bi bil drugače $\beta^{N/\gcd(K,N)}$ enota, ker pa je potenca manjša od N, β potem ni primitiven N-ti koren enote. Dalje, opazimo da je element

$$\gamma = \omega^K \tag{5.7}$$

koren polinoma $\bar{g}(x)$, saj je $\bar{g}(\gamma) = g(\gamma^M) = g(\omega^{KM}) = g(\beta) = 0$. Torej po (5.4) sledi, da so $\gamma, \gamma^2, \gamma^4, \gamma^8, \ldots$ koreni polinoma $\bar{g}(x)$, prav tako tudi produkti teh elementov s poljubnimi potencami α, kjer je

$$\alpha = \omega^N, \tag{5.8}$$

za katerega ni težko pokazati, da je M-ti koren enote. Oglejmo si zdaj naslednjo množico

$$\{\gamma^\tau \alpha^i;\ \tau = 1, 2, 4, 8, \ldots, 2^{n-1}, i = 0, 1, 2, 3, \ldots, M-1\}, \tag{5.9}$$

katere vsi elementi so koreni $\bar{g}(x)$. Ti elementi so različni, kajti če bi bila dva izmed njih enaka, lahko najdemo R in ℓ, tako da $\gamma^R \alpha^\ell = 1$, kjer $0 \le \ell < M$ in $R = \tau' - \tau''$, kjer sta τ' in τ'' potenci števila 2 kot v (5.5). Če zamenjamo zaporedoma γ in α z izrazi (5.7) in (5.8), mora eksponent od ω zadostovati pogoju $RK + \ell N$ je večkratnik MN, ker pa je K reltively prime to N, mora biti R večkratnik N, tako da je $\beta^{\tau'} = \beta^{\tau''}$, saj je $\beta^N = 1$. Torej elementi v (5.5) niso vsi različni, razen če $\tau' = \tau''$ ali $R = 0$. Če velja to, potem mora biti tudi $\ell = 0$. Torej so elementi v (5.9) različni in množica (5.9) je množica vseh korenov polinoma $\bar{g}(x)$, ki ima stopnjo Mn.

Oglejmo si zdaj velikosti orbit, generiranih s temi kvadrati teh korenov. Najprej opazimo, da zaradi (5.7) in (5.8) velja

$$\gamma^N = \alpha^K. \tag{5.10}$$

Prvih $n+1$ členov zaporedja, ki se začne z $\gamma\alpha^i$ za nek i, je enakih

$$\gamma\alpha^i, \gamma^2\alpha^{2i}, \gamma^4\alpha^{4i}, \ldots, \gamma^{Nd+1}\alpha^{(Nd+1)i},$$

od katerih je zadnji člen enak $\gamma\alpha^{Kd+(Nd+1)i}$. Če ta argument razširimo, je po nr kvadriranjih eksponent pri γ enak 1 in eksponent pri α enak

$$i2^{nr} + Kd(1 + 2^n + 2^{2n} + \cdots + 2^{n(r-1)}) = i + (Kd + Ndi) \cdot \frac{2^{nr} - 1}{Nd}.$$

Da bi bila orbita polna po nr kvadriranjih, mora biti

$$d(K + Ni)X_n(r) = \text{večkratnik } M, \tag{5.11}$$

kjer je

$$X_n(r) = \frac{2^{nr} - 1}{2^n - 1}. \tag{5.12}$$

Število elementov v orbiti, ki vsebuje $\gamma\alpha^i$, je nR, kjer je R najmanjša pozitivna vrednost r-ja, ki zadošča (5.11). Ker sta si $(K + Ni)$ in N tuja, mora $X_n(r)$ vsebovati vse praštevilske faktorje števila

$$M_\delta = \frac{M}{\gcd(M, d)}, \tag{5.13}$$

ki so tudi faktorji števila N. To je očitno, saj so faktorji skupni obema, d in M, izločeni iz (5.11).

Zapišimo zdaj $M_\delta = AB$, kjer je A produkt vseh praštevilskih faktorjev M_δ (ne nujno različnih), ki so hkrati tudi faktorji N, in naj bo B največji faktor števila M_δ, ki je tuj glede na N. Potem mora biti $X_n(r)$ večkratnik števila A. To se zgodi natanko tedaj, ko je r večkratnik A, glej [3, str.23]. Tako je najmanjša pozitivna vrednost števila r enaka A. Torej je najnižja možna stopnja poljubnega nerazcepnega faktorja polinoma $\bar{g}$ enaka nA, kar je tudi spodnja meja za linearno zahtevnost.

Za dokaz obstoja takega faktorja, določimo $r = A$ v (5.11) in pogledamo, če lahko najdemo vrednost za i, ki zadošča (5.11). Zadosten pogoj je, da je $K + Ni$ večkratnik števila B, kar pa vedno lahko dosežemo, saj sta si B in N tuja. ■

Če združimo izrek 5.2 in izrek 5.4, dobimo spodnjo mejo za linearno zahtevnost neničelnega zaporedja generiranega z $g(x^M)$. Tudi če je ta polinom razcepen, velja, da ima izmed vseh možnih zaporedij, generiranih s tem polinomom, manj kot $(M/A)2^{-nA}$ linearno zahtevnost, manjšo od polne nM, kar je stopnja polinoma $g(x^M)$. Ta delež je zelo majhen, če je le nA dovolj velik. Opisane rezultate lahko združimo v naslednji izrek.

Izrek 5.5. *Naj bo dan binaren register s povratnim polinomom $f(x)$ stopnje n in reda N, ki je kontroliran s kontrolnim registrom z liho periodo M, tako da je izhodno zaporedje dano z enačbami (5.1) in (5.2). Če je zadoščeno pogojema* **P1** *in* **P2**, *je najmanjša možna linearna zahtevnost neničelnega izhodnega zaporedja enaka nA, kjer je A definiran v izreku 5.4.* ■

Podoben izrek velja tudi za periodo zaporedja, generiranega s CCSR-generatorjem.

Izrek 5.6. *Naj bo M liho pozitivno število. Naj bo $M = A'B'$ faktorizacija, kjer je A' produkt praštevilskih faktorjev števila M (ne nujno različnih), ki so tudi faktorji števila N. Potem je najnižji možni red poljubnega nerazcepnega faktorja polinoma $g(x^M)$ enak NA'.* ■

Dokaz zgornje trditve je podoben dokazu izreka 5.4 in ga mi ne navajamo, saj v mnogih primerih periodo zaporedja lahko določimo na drugačen način, glej [3, str.19]. Če izreku 5.6 dodamo pogoj **P2** lahko dokažemo naslednji izrek.

Izrek 5.7. *Če je zadoščeno pogoju* **P2**, *potem je najmanjša perioda zaporedja $\{u(t)\}$ iz (5.1) enaka MN, kjer je N najmanša perioda zaporedja $\{b(t)\}$.* ■

Za dokaz zgornje trditve glej [16]. S pomočjo izreka 5.7 lahko dokažemo tudi naslednji izrek o linearni zahtevnosti.

Izrek 5.8. *Naj bo binaren register s povratnim polinomom $f(x)$ stopnje n in reda N kontroliran s kontrolnim registrom s periodo $M \geq 2$, ki je potenca števila 2 in naj bo izhodno zaporedje dano s (5.1) in (5.2). Če sta pogoja* **P1** *in* **P2** *izpolnjena, je najmanjša možna linearna zahtevnost poljubnega neničelnega izhodnega zaporedja, generiranega s tem CCSR-generatorjem, vsaj $nM/2$.*

Skica dokaza. Ker je M potenca števila 2, ki je tudi karakteristika obsega, je povratni polinom $f_S(x^M)$ izhodnega zaporedja enak $(f_S(x))^M$. Tu je $f_S(x)$ zopet nerazcepen, vendar pa je v racionalni obliki rodovne funkcije možno krajšanje, tako da nam ostane imenovalec $(f_S(x))^L$, kjer je $0 \leq L \leq M$. Če pa bi veljalo $L \leq M/2$, bi bil imenovalec delitelj polinoma $(1 - x^N)^{m/2}$. Perioda izhodnega zaporedja je potem enaka $MN/2$, kar pa je v protislovju z izrekom 5.7. Torej je najmanjša možna stopnja imenovalca v racionalni obliki rodovne funkcije vsaj $nM/2$. ■

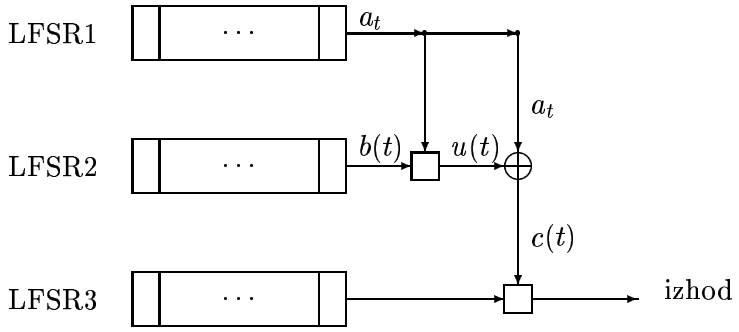
5.2 Sistemi kaskad

Za motivacijo si najprej oglejmo poseben primer izreka 5.4.

Posledica 5.9. *Če je g primitiven ($\delta = 1$) in če so vsi praštevilski faktorji števila M tudi faktorji števila $N = 2^n - 1$, potem je $A = M$ in $g(x^M)$ je nerazcepen.* ■

Upoštevajoč ta rezultat je v tem primeru konstrukcija CCSR-generatorja naslednja. Naj bosta dana dva LFSR-generatorja s primitivnim povratnim polinomom stopnje n in reda $2^n - 1$. Prvi LFSR služi kot kontrolni register, katerega binarno izhodno zaporedje $\{w(t)\}$ kontrolira drugi LFSR. Dva najbolj preprosta primera take konstrukcija sta *stop-and-go generator*, kjer je število korakov a_t enako $w(t)$, in *korak1-korak2* generator, kjer je $a_t = 1 + w(t)$. Ker je število enic v periodi m-zaporedja enako $2^n/2$, je lahko preveriti, da je pogoj **P2** izpolnjen, tako da ima izhodno zaporedje CCSR-generatorja linearno zahtevnost $n(2^n - 1)$. To pokažemo tako, da določimo $g = f_S$ v posledici 5.9. Perioda takega zaporedja pa je potem $(2^n - 1)^2$.

Vzemimo za kontrolni register zgornji CCSR s periodo $M = (2^n - 1)^2$, za G-register pa tretji LFSR dolžine n, s primitivnim povratnim polinomom. Glede na zgornje rezultate bi pričakovali, da so izpolnjeni vsi pogoji posledice 5.9, da dobimo izhodno zaporedje z linearno zahtevnostjo $n(2^n-1)^2$. Vendar pa v tem primeru pogoj **P2** ni izpolnjen, tako da povratni polinom koračnega zaporedja ni nujno primitiven in izreka 5.4 ne moremo uporabiti. Ta problem rešimo tako, da neposrednemu izhodnemu bitu $u(t)$ drugega LFSR-generatorja prištejemo binaren kontrolni bit $a_t \bmod 2$, da dobimo nov izhodni bit $c(t)$ C-registra, kjer je *neposredni izhodni bit* izhodni bit $u(t)$ drugega LFSR-generatorja, pred prištevanjem bita $a_t \bmod 2$. Tak sistem je prikazan na sliki 5.2.



Slika 5.2: Kaskada treh LFSR-generatorjev.

To idejo lahko posplošimo tudi na kaskado K-tih LFSR-generatorjev s primitivnimi povratnimi polinomi. Pri tem je binarno vhodno zaporedje na r-ti stopnji uporabljeno za kontroliranje te stopnje, hkrati pa ga prištejemo neposrednemu izhodnemu zaporedju r-te stopnje $\bmod 2$. Tako dobimo izhodno zaporedje te stopnje, ki ga nato vzamemo kot vhodno zaporedje za $(r+1)$ stopnjo. Vhodno zaporedje za prvo stopnjo (vrh kaskade) je zaporedje samih enic. Vhodno zaporedje pri r-ti stopnji ima periodo $M = N^{r-1}$ in izpolnjuje pogoj **P2**, s tem tudi posledico 5.9, zato ima neposredno izhodno zaporedje linearno zahtevnost nN^{r-1}. Velja še več, izhodno zaporedje r-te stopnje ima periodo N^r in izpolnjuje pogoj **P2**. Takemu sistemu pravimo **kaskade m-zaporedij.**

Da bi pokazali zakaj to res deluje, predpostavimo, da ima vhodno zaporedje r-te stopnje $(r > 1)$ periodo $M = N^{r-1}$ in naj bo število enic v tej periodi enako vrednosti S, ki je tuja glede na $N = 2^n - 1$. Zdaj uredimo MN zaporednih členov vhodnega zaporedja po vrsticah v tabelo

sestavljeno iz M-tih stolpcev in N-tih vrstic. Zaradi periodičnosti je vsak stolpec sestavljen ali iz samih ničel, ali iz samih enic. S stolpcev je zaradi števila enic v vhodnem zaporedju sestavljenih iz samih enic. Če enako uredimo neposredno izhodno zaporedje r-te stopnje, potem je vsak stolpec m-zaporedje, saj je S-koračno podzaporedje m-zaporedja. Torej je v vsakem stolpcu $v = (1/2)2^n$ enic. Izhodno zaporedje je XOR vhodnega zaporedja in neposrednega izhodnega zaporedja, torej je v MN členih tega zaporedja

$$S^{'} = S(N - v) + (M - S)v = SN + N^{r-1}v - 2Sv$$

enic. Število enic v izhodnem zaporedju in N sta si tuja, saj je N liho število, v je potenca števila 2 in S in N sta si tuja. Tako se izpolnjivost pogoja **P2** prenaša iz ene stopnje v drugo. Še več, ker sta si $S^{'}$ in $MN = N^r$ tuja, je to najmanjša perioda izhodnega zaporedja, saj ni mogoče porazdeliti $S^{'}$ enic enakomerno preko ustreznih sudivizij, če je MN večkratnik najmanjše periode.

Izhodno zaporedje K-te stopnje je vsota vseh neposrednih izhodnih zaporedij in zaporedja samih enic z vrha kaskade. Po izreku 5.4 je rodovna funkcija končnega zaporedja vsota parcialnih ulomkov z nerazcepnimi imenovalci stopnje

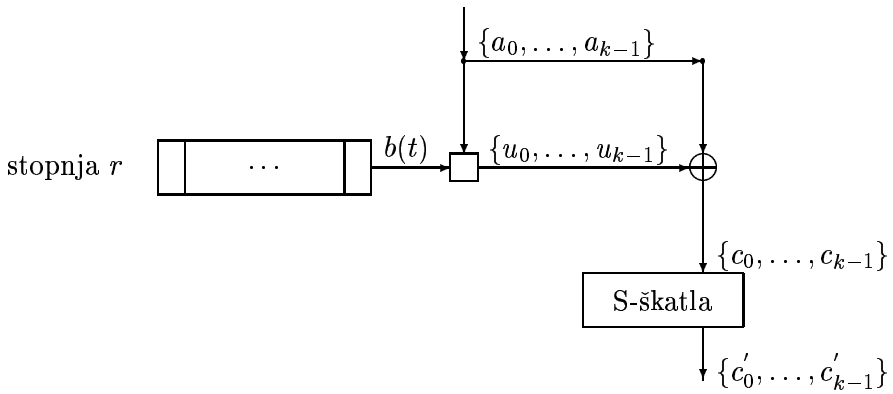
$$1, n, nN, nN^2, \ldots, nN^{K-1},$$

tako da je stopnja skupnega imenovalca, torej tudi linearna zahtevnost, enaka

$$1 + n(1 + N + N^2 + \cdots + N^{K-1}),$$

kar je malo večje od napovedane linearne zahtevnosti nN^{K-1}.

Pri kaskadah m-zaporedij se na vsakem koraku posamezen bit prenaša s stopnje na stopnjo. Podobno lahko naredimo tudi za k bitov, $k \leq n$, ki se prenašajo s stopnje na stopnjo vzporedno skozi obrnljivo S-škatlo. Če k predstavlja neko celo število med 0 in $2^k - 1$, lahko obrnljivo S-škatlo smatramo kot generator permutacij celih števil med 0 in $2^k - 1$. Kontroliranje naslednjega registra je narejeno s pomočjo enega ali več teh bitov. Če sta k in s tem tudi n dovolj velika potem so S-škatle lahko tudi bločne šifre. Delovanje takega sistema je razvidno na sliki 5.3.



Slika 5.3: Večbitne kaskade.

Prednost takega algoritma je v tem, da vzporedno generiranje bitov omogoča učinkovito implementacijo na mikroprocesorjih. Čeprav sta n in k enaka na vseh stopnjah, se lahko obrnljive S-škatle in povratne povezave spreminjajo. Linearna zahtevnost vsakega od k binarnih zaporedij,

ki sestavljajo izhodno zaporedje kaskade sestavljene iz K stopenj, je večja od nN^{K-1}, lahko pa jo z nelinearnostjo v zadnji S-škatli še povečamo. Take sistem imenujemo **večbitne kaskade**.

Slabost opisanih sistemov je v tem, da je linearna zahtevnost manjša od periode za faktor približno n/N, kar je mnogo manjše od 1. Pri pričakovanih vrednostih linearnih zahtevnosti naključnih zaporedij s periodo P pa je ta faktor približno 1, glej [27]. Alternativna konstrukcija sistema kaskad da linearno zahtevnost primerljivo s periodo. Pri tej konstrukciji je LFSR dolžine n s primitivnim povratnim polinomom zamenjan s cikličnim registrom dolžine p, kjer je p praštevilo. Podobno kot v prvem primeru zgoraj, tudi tu lahko pokažemo, da ima vhodno zaporedje na r-ti stopnji periodo p^{r-1} in izpolnjuje pogoj **P2**.

Za izračun linearne zahtevnosti pa najprej predpostavimo, da je število enic v periodi sodo. V tem primeru števec v racionalni obliki rodovne funkcije zadošča pogoju $\Phi(1) = \sum_{i=0}^{p-1} s_i$, kjer s_i za $i = 0, 1, \ldots p-1$ tvori eno periodo. Torej je $\Phi(0) = 0$ in je $(1-x)$ delitelj polinoma $\Phi(x)$. Po krajšanju ima povratni polinom obliko

$$C_p(x) = \frac{1-x^p}{1-x} = 1 + x + x^2 + \cdots + x^{p-1}.$$

Za uporabo izreka 5.5 mora biti polinom $C_p(x)$ nerazcepen. To pa je natanko tedaj, ko je $p-1$ najmanjše pozitivno število j, za katerega je $2^j - 1$ večkratnik števila p, oziroma natanko tedaj, ko je 2 primitiven element v GF(p), glej [16]. Če velja to, je C_p nerazcepen polinom stopnje $n = p-1$ in reda $N = p$. V notaciji izreka 5.4 je potem $\delta = 2^n - 1/N = (2^{p-1} - 1)/p$. V večini primerov sta si δ in p tuja, torej sta si tudi δ in $M = N^{r-1} = p^{r-1}$ tuja, tako da je $M_\delta = M = p^{r-1}$ in $A = p^{r-1}$. Torej ima neposredno izhodno zaporedje na r-ti stopnji linearno zahtevnost $(p-1)p^{r-1}$. Kaskade takih registrov imajo potem periodo r^K in linearno zahtevnost

$$1 + n(1 + N + N^2 + \cdots + N^{K-1}) = p^K.$$

Če pa ima vsebina poljubnega registra liho parnost, potem moramo vsebino komplementirati po bitih (da dobimo sodo parnost) in vsakemu izhodnemu bitu prišteti 1. Torej, če je liho število registrov z liho parnostjo, potem izhod prišteje vsem zaporedje samih enic z rodovno funkcijo $1/(1-x)$. To izniči faktor $1/(1-x)$ v razvoju parcialnih ulomkov rodovne funkcije, tako da je linearna zahtevnost zmanjšana na $p^K - 1$. Kaskade takega tipa imenujemo **kaskade p-ciklov**.

Kaskadni generatorji imajo tudi določene šibkosti, ki jih lahko napadalci izkoristijo, npr. lock-in efekt, opisan v [16], če se jih ne izognemo (na račun hitrosti).

5.3 Drugi sistemi

Vsi urno-kontrolirani pomični registri prikazani do sedaj so bili "brez zank", tj. če register A kontrolira register B neposredno ali posredno preko verige registrov, potem register B zagotovo ne kontrolira registra A direktno ali indirektno.

Oglejmo si zdaj na kratko sisteme, ki imajo zanke. Zgodnji primer takih sistemov medsebojno kontroliranih registrov predstavlja šifrirni stroj T52e, tako da so taki sistemi zanimivi tudi z zgodovinskega stališča. Najpreprostejši sistem z zankami je sestavljen iz LFSR-generatorja dolžine n s primitivnim povratnim polinomom, ki je kontroliran enkrat, če je izhodni bit 0 in dvakrat, če je izhodni bit 1. Slabost takih sistemov je močna korelacija med zaporednimi izhodnimi biti, kar je prav gotovo kriptografska slabost. Predlagan sistem, ki odpravi to slabost je *alternirajoči* generator, ki je sestavljen iz enega kontrolnega registra in dveh CCSR-generatorjev, ki ju označimo z SR in SR$'$. C-register generira binarno zaporedje, ki določi, kateri od pomičnih registrov SR in SR$'$ bo kontroliran. Izhodno zaporedje je binarna vsota izhodnih zaporedij SR in SR$'$. Originalen predlog, glej [18], je bil sestavljen iz C-registra, ki je generiral de Bruijn-ovo

zaporedje s periodo $M = 2^m$, ki ima enako število ničel in enic, tako da je S, definiran pri osnovnemu sistemu, enak $M/2$. V tem primeru sta registra SR in SR$'$ LFSR-generatorja z nerazcepnima povratnima polinomoma $f(x)$ in $f'(x)$, zaporedoma stopnje n in n', ter redov N in N', ki sta tuja glede na n in n'. Če za tak sistem uporabimo izreka 5.7 in 5.8, je dokaj enostavno pokazati, da je perioda izhodnega zaporedja MNN', linearna zahtevnost pa vsaj $((n+n')M)/2$.

5.4 $\mathcal{D}$-kontroliran CCSR

V tem razdelku je podana alternativna definicija in oznake, ki jih bomo uporabljali v nadaljevanju.

Definicija 5.10. *Naj bo $D = \{a_t\}_{t=1}^{\infty}$ izhodno zaporedje kontrolnega registra. Z $\mathcal{D}$ označimo zalogo vrednosti zaporedja D. Če je $\mathcal{D} = \{k, m\}$, potem je tak CCSR* **$\{k, m\}$-kontroliran**. *Če je $\mathcal{D} = [1, k] = \{1, 2, \ldots, k\}$, potem je CCSR* **$[1, k]$-kontroliran**.

Primer. Če je $\mathcal{D} = \{0, 1\}$ potem je $\{0, 1\}$-kontroliran CCSR kar stop-and-go generator. •

Poglavje 6

NAPADI NA CCSR

CCSR je primeren za uporabo v generatorjih toka ključev za tokovne šifre, saj imajo taki generatorji dolgo periodo in visoko linearno zahtevnost, kar smo pokazali v prejšnjem razdelku. Napadov na CCSR je več vrst, napadi na stop-and-go pomične registre so opisani v [2], napadi na kaskade CCSR so opisani v [4]. Mi si bomo v nadaljevanju ogledali napade na CCSR, ki temeljijo na problemu vložitve niza X v niz Y. Najprej si bomo ogledali Živkovićev algoritem, to je napad na $\{1,2\}$-kontroliran CCSR. Sledil bo bolj splošen napad na neregularno kontrolirane pomične registre, kjer je število preskokov na korak navzgor omejeno z d. Poglavje bomo zaključili z napadom na CCSR, kjer število preskokov na korak ni navzgor omejeno. Skupno vsem napadom je, da ima napadalec dan del toka ključev, iz katerega želi konstruirati začetno stanje CCSR-generatorja, ne da bi poznal koračno zaporedje. V tem poglavju bomo prikazali napade same. Varnost CCSR-generatorja, ter s tem tudi uspešnost teh napadov bomo analizirali v naslednjem poglavju. Še preden pa se lotimo napadov, bomo navedli nekaj definicij in prikazali problem vložitve.

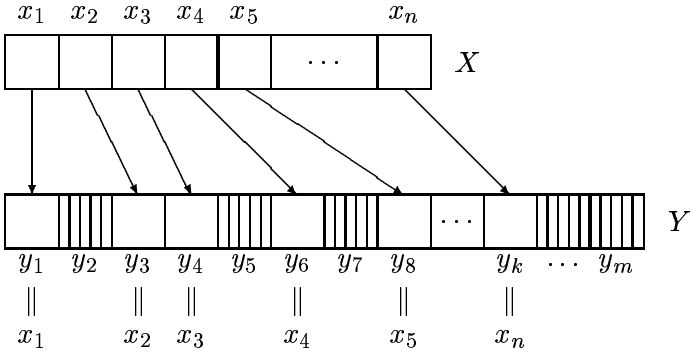
6.1 Verjetnostni model

Ogeljmo si najprej problem vložitve. Niz X lahko 1-vložimo v niz Y, če Y lahko generiramo iz X tako, da med dva zaporedna bita niza X vstavimo največ en bit in ustrezno število bitov na konec. Formalna definicija 1-vložitve je naslednja.

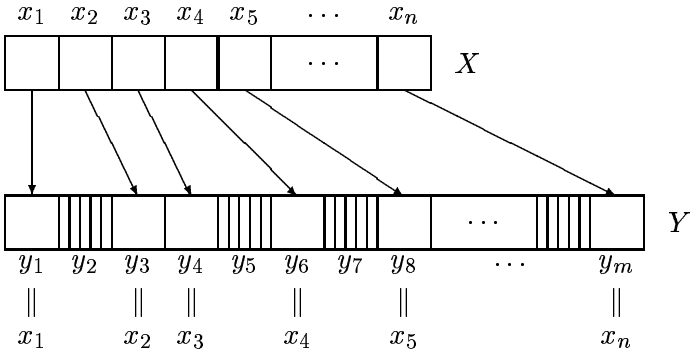
Definicija 6.1. *Naj bosta $X = \{x_i\}_{i=1}^{n}$ in $Y = \{y_i\}_{i=1}^{m}$ binarna niza zaporedoma dolžine n in m, $m \geq n$. Naj $X_i = \{x_j\}_{j=1}^{i}$ označuje prvih i členov niza X, kjer $1 \leq i \leq n$. Niz X lahko* **1-vložimo v** *niz Y, če obstaja zaporedje celih števil $S = \{s_i\}_{i=1}^{n}$, tako da je $x_i = y_{s_i}$ za $1 \leq i \leq n$ kjer je $s_1 = 1$ in $s_i - s_{i-1} \in \{1,2\}$ za $2 \leq i \leq n$.*
Če dodatno velja še $m - s_n \in \{0,1\}$, potem niz X lahko **strogo 1-vložimo v** *niz Y. Če $s_n = m$, potem pravimo, da X lahko* **1-vložimo na** *Y.*

Zgornja definicija nam prikaže problem vložitve v primeru, ko je CCSR $\{1,2\}$-kontroliran. 1-vložitev niza X v niz Y je prikazana tudi na sliki 6.1. Na sliki 6.2 je prikazana 1-vložitev niza X na niz Y. Definicijo lahko posplošimo tudi za primer, ko je $1 \leq a_i \leq d+1$, kjer je zaporedje a_i zaporedje, ki ga generira kontrolni register in je opisano v prejšnjem razdelku. Niz X lahko d-vložimo v niz Y, če X lahko konstruiramo iz Y tako, da brišemo največ d zaporednih bitov, pri čemer mora prvi bit niza Y ustrezati prvemu bitu niza X. Formalna definicija d-vložitve je naslednja.

Definicija 6.2. *Niz X je* **d-vložen v** *niz Y, če obstaja zaporedje celih števil $S = \{s_i\}_{i=1}^{n}$, tako da je $x_i = y_{s_i}$ za $1 \leq i \leq n$, kjer je $s_1 = 1$ in $1 \leq s_i - s_{i-1} \leq d$ za $2 \leq i \leq n$. $m - s_n \leq d$ potem X lahko* **strogo d-vložimo v** *Y.*



Slika 6.1: 1-vložitev niza X **v** niz Y. Med zaporedna bita niza X lahko dodamo največ en bit.



Slika 6.2: 1-vložitev niza X **na** niz Y, torej mora biti zadnji bit niza X enak zadnjemu bitu niza Y.

Če primerjamo obe definiciji vidimo, da je prva samo poseben primer druge, ko je $d = 1$. Obe definiciji smo navedli zaradi lažje predstave, poleg tega pa si bomo v nadaljevanju najprej ogledali napad v primeru $d = 1$.

Problem vložitve lahko definiramo tudi v primeru, ko d ni navzgor omejen, tj., ko je $\mathcal{D} = \mathbb{Z}^+$.

Definicija 6.3. *Niz X dolžine n lahko* **$\mathcal{D}$-vložimo v** *dan binaren niz Y dolžine m, $m \geq n$, če obstaja niz $D = \{d_i\}_{i=1}^{n}$, kjer $d_i \in \mathcal{D}$ in $x_i = y_s$ za $s = \left(\sum_{j=1}^{i} d_j\right)$ in $1 \leq i \leq n$.*

V primeru, ko d ni navzgor omejen, dovoljujemo tudi možnost $y_1 \neq x_1$. Podobno bi lahko dovolili tudi v definiciji 6.2, vendar bomo kasneje pri računanju verjetnosti to predpostavko potrebovali, medtem ko v primeru, ko d ni omejen navzgor, verjetnost lahko izračunamo tudi brez te predpostavke.

V primeru, ko d ni navzgor omejen, bomo potrebovali tudi *stopnjo brisanj*.

Definicija 6.4. *Naj bo $\{a_t\}$ koračno zaporedje, X slučajna spremenljivka z zalogo vrednosti $\mathcal{D}$ in $\mathcal{P} = \{P(X = d)\}_{d \in \mathcal{D}}$ njena porazdelitev. Naj bo $\bar{d} = E(X) = \sum_{d \in \mathcal{D}} d \cdot P(X = d)$ matematično upanje slučajne spremenljivke X.* **Stopnja brisanja** *p_d je definirana kot*

$$p_d = 1 - \frac{1}{\bar{d}}. \tag{6.1}$$

Stopnja brisanja je v bistvu relativno pričakovano število brisanj členov zaporedja Y, potrebnih za generiranje izhodnega zaporedja X. Stopnja brisanja je potrebna za kontroliranje verjetnosti zgrešenega dogodka v $\mathcal{D}$-vložitvi, poleg tega pa je uporabna za karakterizacijo učinkovitosti napada z neomejeno vložitvijo, kot bomo videli kasneje.

Bistvo analize napadov z vložitvami je računanje *vložitvene verjetnosti.*

Definicija 6.5. **$\mathcal{D}$-vložitvena verjetnost** *$P_{\mathcal{D},X}(n,m)$ je verjetnost, da dani binarni niz X dolžine n lahko $\mathcal{D}$-vložimo v naključen, enakomerno porazdeljen binaren niz Y dolžine m Če je $\mathcal{D} = [1, d+1]$, potem vložitveno verjetnost $P_{[1,d+1],X}(n, m(n))$ označimo s $P_{d,X}(n,m)$.*

Dolžina m niza Y je odvisna od dolžine n niza X in jo označimo z $m(n)$. Izbrana mora biti tako, da je verjetnost zgrešenega dogodka P_m dovolj majhna. Zgrešeni dogodek se zgodi takrat, ko med vsemi binarnimi nizi dolžine $m(n)$ ni takega, v katerega bi dani niz X dolžine n lahko $\mathcal{D}$-vložili. Očitno je $P_m = P\left(\sum_{i=1}^{n} d_i > m(n)\right)$.

6.2 Napad z vložitvami na $\{1,2\}$-kontroliran CCSR

Napad na $\{1,2\}$-kontroliran CCSR, ki ga bomo predstavili, temelji na rekonstrukciji začetnega stanja generatorja toka ključev **G**. Spomnimo se konstrukcije CCSR-generatorja iz 5. poglavja. Generator toka ključev **G** je sestavljen iz kontrolnega registra, ki ga označimo z G_0 in binarnega linearnega pomičnega registra (LFSR), ki je kontroliran z G_0. Naj bodo binarna zaporedja $\{a_i\}_{i=1}^{\infty}$, $\{b_i\}_{i=1}^{\infty}$ in $\{c_i\}_{i=1}^{\infty}$ zaporedoma izhodno zaporedje registra G_0, izhodno zaporedje, ki ga generira LFSR in tok ključev generatorja **G**. Izhodno zaporedje LFSR-generatorja je, kot vemo določeno z vrednostmi $b_1, b_2, \ldots, b_k$ in rekurzivno relacijo

$$b_n = \sum_{i=1}^{k} h_i b_{n-i}, \quad n > k, \tag{6.2}$$

kjer je k dolžina LFSR-generatorja, $h_1, h_2, \ldots, h_k \in GF(2)$ pa so njegovi povratni koeficienti. Izhodno zaporedje generatorja toka ključev **G** dobimo iz naslednje enakosti:

$$c_n = b_{r_n}, \quad n \geq 1 \tag{6.3}$$

kjer je $r_n = n + \sum_{i=1}^{n} a_i, \quad n \geq 1$, oziroma

$$r_{n+1} = r_m + 1 + a_{n+1}, \quad n \geq 1. \tag{6.4}$$

Algoritem, s katerim bomo rekonstruirali začetno stanje, je opisan v Živković [31] in ga po avtorju kasneje tudi imenujemo. Temeljil bo na iskanju podnizov v periodi izhodnega zaporedja LFSR-generatorja, v katere lahko nize $(c_i, c_{i+1}, \ldots, c_N), \quad 1 \leq i < N$ 1-vložimo, pri čemer predpostavljamo naslednje.

1. Znanih je prvih N členov $c_1, c_2 \ldots c_N$ izhodnega zaporedja generatorja toka ključev **G**, kjer je $N > 0$.

2. Če poznamo $K > 0$ členov zaporedja $a_1, a_2, \ldots a_n, \ldots$, lahko izračunamo začetno stanje kontrolnega registra G_0 (G_0 je lahko na primer LFSR).

Naj bo P perioda LFSR-generatorja, N dolžina znanega toka ključev in $B = P + N$. Naj bo niz $\boldsymbol{b}= (b_2, b_2, \ldots, b_B)$ del izhodnega zaporedja LFSR-generatorja, z začetnim stanjem $b_1 = b_2 = \ldots = b_k = 1$, glej (6.2). Algoritem za rekonstrukcijo začetnega stanja generatorja toka ključev je naslednji.

Algoritem 5.1. *Rekonstrukcija začetnega stanja generatorja toka ključev G.*

1. $i \leftarrow N,\ t \leftarrow 0$;

   for $j = 1$ **to** B **do** $p_j \leftarrow \delta_{c_N, b_j}$

   { $\delta_{i,j}$ *je Kroneckerjev simbol. Vektor* $\boldsymbol{p}$ *vsebuje informacijo o možnih položajih* c_i *v* $\boldsymbol{b}$*;* i *in* t *sta števca.*}

2. $i \leftarrow i - 1$;

   while $i > 0$ **do**

3. **for** $j = 1$ **to** B **do**

   if $c_j = b_j$ **and**

   $((j + 1 \leq B$ **and** $p_{j+1} = 1)$ **or** $(j + 2 \leq B$ **and** $p_{j+2} = 1))$

   then $q_j \leftarrow 1$;

   else $q_j \leftarrow 0$.

   {*Vektor* $\boldsymbol{q}$ *vsebuje informacijo o možnih položajih* c_i *v* $\boldsymbol{b}$.}

4. **if** $\sum_{i=1}^{B} q_j = 1$ **and** $q_s = 1$ **then**

   $t \leftarrow t + 1,\ u_t \leftarrow i,\ v_t \leftarrow s$.

   {*Bit* c_i *je dobljen iz* b_s *izhodnega zaporedja* G *in* $r_i = s$*, glej (6.3).*}

5. **for** $j = 1$ **to** B **do** $p_j \leftarrow q_j$;

6. $i \leftarrow i - 1$;

7. *Za vsak par* $(j, j + 1)$, $1 \leq j < t$, *za katerega je* $u_j = t$ *in* $u_{j+1} = t + 1$ *izračunaj* $a_{t+1} = v_{j+1} - v_j - 1$ *(členi kontrolnega zaporedja, glej (6.4).) Če je število takih parov dovolj veliko, lahko izračunamo začetno stanje* G_0*, tako da rešimo ustrezen sistem enačb.*

8. *Če smo našli začetno stanje* G_0*, lahko izračunamo začetno stanje LFSR-generatorja zopet z reševanjem sistema linearnih enačb. Druga možnost je, da začetno stanje določimo iz* k *zaporednih bitov zaporedja* $\{b_i\}_{i=1}^{B}$*, z začetkom na mestu, kjer je ustrezna koordinata vektorja* $\boldsymbol{p}$ *enaka 1. Kasneje bomo pokazali, da je takih mest malo, če je le* N *dovolj velik.*

Z indukcijo po i, kjer $i = N, N-1, \ldots, 1$ lahko dokažemo, da na $(N - i + 1)$-em koraku velja $q(t) = 1$ natanko tedaj, ko podzaporedje $(c_i, c_{i+1}, \ldots c_N)$ lahko vložimo v zaporedje $(b_t, b_{t+1}, ..., b_B)$ brez vstavljanja na začetku. Kot bomo videli kasneje, je verjetnost, da zaporedje $\{c\}$ vstavimo na napačno mesto v zaporedja $\{b\}$, majhna, če je le N dovolj velik.

Glavna omejitev tega algoritma je v časovni zahtevnosti, ki je reda $O(N2^k)$, saj predpostavljamo, da ima LFSR dolžine k maksimalno periodo, ki je pri izpolnjenih pogojih enaka $2^k - 1$. Rekonstrukcija začetnega stanja torej ni praktično izvedljiva, če je na primer dolžina LFSR-generatorja večja od 30, glej [31].

6.3 Napad z vložitvami na $\mathcal{D}$-kontroliran CCSR

V prejšnjem razdelku smo si ogledali napad na $\{1,2\}$-kontroliran CCSR. Zdaj bomo napad posplošili na $\mathcal{D}$-kontroliran CCSR, kjer $\mathcal{D} = [1, d+1]$. Nato si bomo ogledali še napad z neomejenimi vložitvami, kjer je $\mathcal{D} = \mathbb{Z}^+$. Namen napada je isti kot v prejšnjem razdelku, rekonstrukcija začetnega stanja CCSR, ki generira zaporedje Y dolžine m, v katerega lahko dano zaporedje X, dolžine n d-vložimo. Napad je v bistvu iskanje vseh začetnih stanj pomičnega proizvajalnega registra, ki regularno kontroliran, generira zaporedje, v katerega lahko d-vložimo znan del toka ključev. Uspešen je, če je takih začetnih stanj malo. To se zgodi natanko tedaj, ko je verjetnost, da dani niz X lahko d-vložimo v naključni niz Y, majhna. Opazimo tudi, da napad na $\mathcal{D}$-kontroliran CCSR lahko uporabimo za vsak $\mathcal{D}'$-kontroliran CCSR, kjer $\mathcal{D}' \subseteq \mathcal{D}$.

Oglejmo si zdaj zadostno dolžino m niza Y, da je verjetnost zgrešenega dogodka zelo majhna. Očitno je pri $m(n) = n(d+1)$ verjetnost $P_m = 0$, saj med nizi dolžine $n(d+1)$ obstaja vsaj eden, v katerega dani niz X dolžine n lahko d-vložimo.

Rekonstrukcija začetnega stanja je možna le, če se verjetnost vložitve $P_{\mathcal{D},Y}(n, m(n))$ bliža vrednosti 0, ko n narašča. Če verjetnost pada proti 0 eksponentno, potem je minimalna dolžina znanega toka ključev linearna glede na dolžino pomičnega registra. K problemu verjetnosti vložitve in posledicami te za učinkovitost napada, se bomo vrnili v naslednjem poglavju. Zaenkrat si samo oglejmo, kako tak napad izgleda.

Kot smo že omenili, je napad iskanje vseh začetnih stanj pomičnega registra, ki regularno kontroliran, generira niz x, v katerega lahko d-vložimo znan del toka ključev. Ostane nam še, vprašanje, kako za dani niz Y, preverimo, ali niz X lahko vanj d-vložimo. Za $d \in \{1,2\}$ smo si to ogledali v prejšnjem razdelku. Posplošitev Živkovićevega algoritma za ugotavljanje, ali X lahko $\mathcal{D}$-vložimo v Y, pri čemer je $\mathcal{D} = [1, d+1]$ pa izgleda takole.

Algoritem 5.2 *Posplošitev algoritma 5.1.*

1. $i \leftarrow n$
2. **while** $i > 0$ **do**
3. *Poiščemo vsa možna mesta, kjer se zadnji bit x_n niza X ujema z biti niza Y. Pri tem upoštevamo $\mathcal{D} = [1, d+1]$ (glej korak 3 v algoritmu 5.1.)*
4. *Izvedi koraka 4 in 5 algoritma 5.1.*
5. $i \leftarrow i - 1$;

Posplošitev algoritma 5.1 se razlikuje v koraku 3, kjer upoštevamo $\mathcal{D} = [1, d+1]$. Ostali koraki so identični, z izjemo reševanja ustreznega sistema enačb. Časovna zahtevnost zgornjega postopka je $O(nm)$, oziroma v primeru $\{1,2\}$-kontroliranega CCSR, $O(N2^k)$.

Drug postopek, ki ga lahko uporabimo za ugotovaljanje ali lahko niz X d-vložimo v niz Y temelji na omejeni Levensthein-ovi razdalji in je podrobneje opisan v [12]. Po tem algoritmu je vložitev možna natanko tedaj, ko je razdalja enaka minimalni vrednosti, ki je v bistvu razlika med dolžinama nizov. Računska zahtevnost tega algoritma je $O(n(m-n))$.

Oglejmo si zdaj napad z neomejeno vložitvijo. Napad je skoraj identičen napadu z omejeno vložitvijo, le da tu, če uporabljamo deli in vladaj algoritem, ni potrebno upoštevati omejitev, danih z $\mathcal{D}$, saj je $\mathcal{D} = \mathbb{Z}^+$. Druga modifikacija zgornjega algoritma je v tem, da začnemo preverjati najprej za prvi bit niza X, ker zaradi $\mathcal{D} = \mathbb{Z}^+$ ni omejitev za d. Časovna zahtevnost tega postopka je $O(nm)$. Drug pristop je uporaba algoritma, ki temelji na neomejeni Levensthein-ovi razdalji. Ta postopek je podrobneje opisan v [12].

Poglavje 7

KRIPTOANALIZA CCSR

V prejšnjem poglavju smo si ogledali napad na CCSR, ki temelji na algoritmu za rekonstrukcijo začetnega stanja CCSR-generatorja. Algoritem predstavljen v 6. poglavju je poiskal niz Y, v katerega smo dani niz X lahko vložili. Tu X predstavlja del toka ključev, katerega napadalec pozna, Y pa izhodno zaporedje proizvajalnega registra. Če najdemo tak niz Y oziroma začetno stanje proizvajalnega registra, ki ta niz generira, potem lahko rekonstruiramo cel tok ključev. Kot smo videli, je bila uspešnost napada pogojena s številom možnih kandidatov za niz Y. Če je takih kandidatov malo, je velika verjetnost, da so ti pravi izhod proizvajalnega registra. Malo kandidatov pa je natanko tedaj, ko je verjetnost $P_{d,X}(n,m)$, da vložimo niz X dolžine n v naključen niz Y dolžine m, majhna.

V tem poglavju bomo ocenili to verjetnost. Hkrati bomo navedli tudi pogoj za minimalno dolžino znanega toka ključev, ki je potrebna za uspešno rekonstrukcijo začetnega stanja. Poglavje razdelimo takole. Najprej si bomo ogledali analizo napada z neomejenimi vložitvami. Razlog je preprost, vložitveno verjetnost je v tem primeru lahko izračunati, kar pa za verjetnost vložitve pri $d = 1$ ne moremo trditi. S pomočjo te verjetnosti bomo pokazali, kdaj je napad uspešen. Temu bo sledila groba ocena zgornje meje za $d = 1$ in analitičen izračun te zgornje meje. Obe zgornji meji bomo uporabili pri analizi varnosti CCSR-generatorjev pred napadi z vložitvami in dolžine znanega toka ključev, potrebne za uspešno rekonstrukcijo začetnega stanja. Problem vložitve bomo nato razširili za $d \geq 1$. Tu bomo pri izračunu vložitvene verjetnosti uporabili teorijo končnih avtomatov, razvito v 3. poglavju. Na koncu bomo izračunali dolžino toka ključev, potrebno za uspešno rekonstrukcijo začetnega stanja.

Problem računanja vložitvene verjetnosti $P_{d,X}(n,m)$ je težak problem in zaenkrat je bila izračunana le verjetnost v primeru $d = 1$. Kljub temu pa so znane ocene dovolj, da pokažemo, da CCSR teoretično ni varen pred napadi z vložitvami, tudi če je d velik. Večji d zagotavlja večjo praktično varnost.

Napad z vložitvami je torej uspešen, če je možnih kandidatov za izhod proizvajalnega registra malo. To je ekvivalentno temu, da je vložitvena verjetnost $P_{\mathcal{D},X}(n,m)$ majhna. Uspešnost napada pa lahko merimo tudi z verjetnostjo napačnega alarma P_f. **Napačni alarm** se zgodi takrat, ko najdemo niz Y, v katerega lahko vložimo dani niz X, vendar pa niz Y ni izhodno zaporedje proizvajalnega registra. Vložitvena verjetnost in verjetnost napačnega alarma sta tesno povezani, glej [28]. Naj bo r dolžina proizvajalnega registra. Verjetnost P_f lahko dobro aproksimiramo z naslednjo enačbo:

$$P_f = 1 - (1 - P_{\mathcal{D},X}(n, m(n)))^{2^r - 1}. \tag{7.1}$$

Izpeljava zgornje enačbe zahteva pomembna orodja verjetnosti, kot na primer pogojno porazdelitev in pogojno verjetnost. Zaradi zapletenosti izpeljavo tu izpuščamo, bralec jo lahko najde

v [28]. Ker se izhodna zaporedja proizvajalnega registra obnašajo podobno kot naključna zaporedja, si zgornjo enačbo lahko razlagamo tudi takole. Izraz na desni $(1 - P_{\mathcal{D},X}(n,m))^{2^r-1}$ predstavlja verjetnost, da danega niza X ne moremo vložiti v niz dolžine m generiran s proizvajalnim registrom. Takih nizov je $2^r - 1$, toliko kot je začetnih neničelnih stanj. Če to vrednost odštejemo od 1, dobimo verjetnost, da smo dani niz vložili v niz Y dolžine m, ki pa ni izhodni niz proizvajalnega registra. Za P_f bi radi videli, da je blizu 0, tj. $P_f \approx 0$. To velja, če je izpolnjen naslednji pogoj:

$$2^r \cdot P_{\mathcal{D},X}(n, m(n)) \leq 1. \tag{7.2}$$

Iz pogoja (7.2) in ocene za vložitveno verjetnost lahko izračunamo dolžino n znanega dela toka ključev, potrebno za uspešno rekonstrukcijo začetnega stanja CCSR-generatorja.

7.1 Kriptoanaliza napada z neomejeno vložitvijo

V tem razdelku si bomo najprej ogledali vložitveno verjetnost $P_{\mathcal{D},X}(n, m(n))$, kjer je $\mathcal{D} = \mathbb{Z}^+$. Za lažji zapis naj $P_X(n,m)$ označuje verjetnost $P_{\mathbb{Z}^+,X}(n,m)$. Napad z neomejeno vložitvijo bo uspešen, če je $P_X(n,m)$ majhna. Velja naslednji izrek.

Izrek 7.1. *Naj bo X poljuben binaren niz dolžine n in naj bo*

$$P_+(n,m) = \sum_{k=0}^{m-n} \binom{n-1+k}{k} 2^{-n-k}. \tag{7.3}$$

Potem za neomejeno vložitveno verjetnost velja

$$P_X(n,m) = P_+(n,m) = 1 - 2^{-m} \sum_{k=0}^{n-1} \binom{m}{k}. \tag{7.4}$$

Dokaz. Predpostavimo da niz X dolžine n lahko $\mathbb{Z}^+$-vložimo v niz Y dolžine m. Dokazali bomo, da obstaja koračni niz $D^* = \{d_i^*\}_{i=1}^n$, ki je minimalen v smislu, da je vsak od njegovih elementov minimalen v množici vseh koračnih nizov za dani X in Y. Naj bo d_0^* enak najmanjšemu pozitivnemu celemu številu j, tako da je $x_1 = y_j$. Iterativno nadaljujemo, tj. za $2 \leq i \leq n$ naj bo d_i^* enak najmanjšemu pozitivnemu številu j, tako da je $x_i = y_s$, kjer je $s = \left(\sum_{k=1}^{i-1} d_k^* + j\right)$. Hitro se lahko prepričamo, da je tako dobljeni D^* minimalen. Poleg tega pa iz konstrukcije sledi, da je D^* enoličen.

Očitno velja, $P_X(n,m) = A_X(n,m)/2^m$, kjer je $A_X(n,m)$ število binarnih nizov Y dolžine m, v katere lahko niz X dolžine n $\mathbb{Z}^+$-vložimo. Enoličnost D^* zagotavlja, da različni D^* tvorijo različne nize Y. Vsak D^*, kjer velja $\sum_{i=1}^n d_i = m-k$ za $0 \leq k \leq m-n$, tvori natanko 2^k različnih nizov Y. Ker je za vsak $0 \leq k \leq m-n$ natanko $\binom{m-k-1}{n-1}$ takih D^*, neodvisno od X, enakost (7.3) sledi. Enakost (7.4) dobimo, če za X vzamemo konstanten niz. ■

Nadaljujmo zdaj z analizo asimptotičnih lastnosti verjetnosti $P_+(n,m)$. Naj bo $\{m(n)\}_{n=1}^\infty$ zaporedje pozitivnih celih števil, tako da velja

$$\lim_{n\to\infty} \frac{n}{m(n)} = 1 - \lambda, \tag{7.5}$$

kjer je $0 \leq \lambda \leq 1$. Če uporabimo naslednjo formulo (glej [30])

$$\frac{1}{\sqrt{8k(n-k)/n}} 2^{nH(k/n)} \leq \sum_{i=0}^{k} \binom{n}{i} \leq 2^{nH(k/n)}, \tag{7.6}$$

ki velja za $k \leq n/2$, kjer je $H(x) = -x\log_2 x - (1-x)\log_2(1-x)$ **binarna funkcija entropije**, potem iz (7.4) sledi naslednja posledica.

Posledica 7.2. *Če je $\{m(n)\}_{n=1}^{\infty}$ celoštevilsko zaporedje, ki zadošča pogoju (7.5), potem je*

$$\lim_{n\to\infty} -\frac{\log P_+(n,m(n))}{n} = \frac{1-H(\lambda)}{1-\lambda},\ 0 \leq \lambda < 0.5 \tag{7.7}$$

$$\lim_{n\to\infty} P_+(n,m(n)) = \begin{cases} 0.5 & \lambda = 0.5 \\ 1 & 0.5 < \lambda \leq 1, \end{cases} \tag{7.8}$$

kjer je $H(x)$ binarna funkcija entropije. ■

Naslednji izrek nam bo povedal, kdaj je napad z neomejenimi vložitvami uspešen.

Izrek 7.3. *Za vsak $\mathcal{D} \subseteq \mathbb{Z}^+$, je napad z neomejeno vložitvijo na $\mathcal{D}$-kontroliran CCSR s stopnjo brisanj p_d uspešen, če je $p_d < 0.5$ in če dolžina n opazovanega toka ključev zadošča pogoju*

$$n \geq r\frac{1-p_d}{1-H(p_d)}. \tag{7.9}$$

Če je $p_d \geq 0.5$, napad ni uspešen.

Dokaz. Oglejmo si napad na $\mathcal{D}$-kontroliran pomični register z neomejeno vložitvijo, kjer $\mathcal{D} \subseteq \mathbb{Z}^+$. Da bi bila verjetnost zgrešenega dogodka P_m manjša od 1, je potreben in zadosten pogoj, da je $\lambda \geq p_d$, kjer je p_d stopnja brisanja za $\mathcal{D}$-kontroliran pomični register, glej definicija 6.4. Če vzamemo $\lambda = p_d$, potem je $m(n) = 1/(1-p_d) + c\sqrt{n}$, kjer je konstanta c odvisna od verjetnosti P_m. Če pa je $\lambda < p_d$, tj. če P_m konvergira proti 1, potem je pravo začetno stanje neločljivo od ostalih. Če to združimo s posledico 7.2 in kriterijem (7.2), zgornja trditev sledi. ■

Izrek 7.3 nam pove, da je $\mathcal{D}$-kontroliran CCSR, za vsak $\mathcal{D} \subseteq \mathbb{Z}^+$, teoretično varen pred napadom z neomejeno vložitvijo, če je stopnja brisanja p_d večja ali enaka $1/2$.

7.2 Groba ocena vložitvene verjetnosti $P_{1,X}(n,m)$

Oglejmo si zdaj oceno zgornje meje vložitvene verjetnosti za $d = 1$. Kot smo že omenili pri vložitveni verjetnosti, glej 6.5, bomo za dolžino m niza Y, kamor želim 1-vložiti niz X, vzeli $m(n) = n(d+1)$. Pri $d = 1$ je torej $m = 2n$. Zaradi lažjega zapisa uvedemo naslednjo oznako:

$$P_{N,c} = P_{\{1,2\},c}(N, 2N).$$

$P_{N,c}$ označuje verjetnost, da niz c, dolžine N, lahko 1-vložimo v naključen niz B, dimenzije $2N$, katerega koordinate so neodvisne, enakomerno porazdeljene slučajne spremenljivke. Živkovićevo oceno zgornje meje za vključitveno verjetnost nam daje naslednji izrek.

Izrek 7.4. *Naj bo $\boldsymbol{c} = (c_1, c_2, \ldots, c_N)$ binaren vektor dolžine N in naj bo $\boldsymbol{B} = (B_1, B_2, \ldots, B_{2N})$ $2N$-dimenzionalen binaren vektor z neodvisnimi koordinatami in enakomerno porazdelitvijo. Naj $P_{N,c}$ označuje verjetnost, da lahko vektor $\boldsymbol{c}$ 1-vložimo v naključen vektor $\boldsymbol{B}$ in naj bo $P_N = \max_c P_{N,c}$. Potem obstajata konstanti $\alpha, \beta > 0$, $\alpha < 1$, tako da je*

$$P_N < \beta\alpha^N. \tag{7.10}$$

Dokaz. Naj bo $\Phi(\boldsymbol{c})$ množica vektorjev dolžine $2N$, v katere lahko 1-vložimo vektor $\boldsymbol{c}$. Naš cilj je najti zgornjo mejo za $|\Phi(\boldsymbol{c})|$. Potem je zgornja meja za P_N kvocient števila $|\Phi(\boldsymbol{c})|$ in 2^{2N}. Predpostavimo, da je $N = mM$, kjer sta m in M celi števili. Za $0 \leq \ell \leq m$ definiramo $U_{m,\ell}$ kot maksimalno število različnih vektorjev dolžine $m + \ell$, v katere lahko 1-vložimo poljubno m-terico, tako da vstavimo ℓ bitov (največ enega za vsakim elementom m-terice). Primeri števil $U_{m,\ell}$ za $2 \leq m \leq 7$ so dani v tabeli 7.1.

m/ℓ	0	1	2	3	4	5	6	7
2	1	3	4					
3	1	4	8	8				
4	1	5	13	20	16			
5	1	6	19	38	48	32		
6	1	7	26	63	104	112	64	
7	1	8	34	96	192	272	256	128

Tabela 7.1: Primeri števil $U_{m,\ell}$.

Definirajmo še naslednja simbola:

$$\alpha_m = \frac{1}{2}\left(\sum_{\ell=0}^{m} U_{m,\ell} \cdot 2^{-\ell}\right)^{\frac{1}{m}} \tag{7.11}$$

in p_m, ki je največje število $p \in \left[0, \frac{1}{2}\right)$, tako da je entropija navzgor omejena z $\log_2 \alpha_m$, tj.:

$$H(p) = -p \log_2 p - (1-p) \log_2 (1-p) \leq -\log_2 \alpha_m.$$

Primeri zgornjih vrednosti za α_m in p_m, kjer je $2 \leq m \leq 7$, so v tabeli 7.2.

m	α_m	$-1/\log_2 \alpha_m$	p_m
2	0.9354143	10.3817861	0.0123972
3	0.9085603	7.2282625	0.0194109
4	0.8946455	6.2261849	0.0235022
5	0.8862936	5.7423940	0.0260349
6	0.8807541	5.4588471	0.0277883
7	0.8768163	5.2727731	0.0291521

Tabela 7.2: Primeri števil α_m, $-1/\log_2 \alpha_m$ in p_m.

Za $m \geq 2$ velja $U_{0,m} = 1$ in $U_{m,m} = 2^m$. Iz naslednje neenakosti

$$U_{m,\ell} \leq \binom{m}{\ell} 2^\ell, \quad 0 \leq \ell \leq m, \tag{7.12}$$

in enakosti $U_{m,1} = m + 1$ sledi

$$\alpha_m < 1, \tag{7.13}$$

saj v neenakosti (7.12) za $\ell = 1$ velja strogi neenačaj.

Očitno je $\boldsymbol{b} \in \Phi\boldsymbol{c}$) natanko tedaj, ko $\boldsymbol{b}$ lahko razbijemo v $M + 1$ delov, tako da za $1 \leq i \leq M$ vektor $\boldsymbol{c}^i = (c_{mi-m+1}, \ldots, c_{mi})$ lahko 1-vložimo na i-ti del zgornje delitve vektorja $\boldsymbol{b}$. Predpostavimo, da smo fiksirali število vstavljanj v $\boldsymbol{c}^i$, kjer je $1 \leq i \leq M$ in naj bo I_ℓ število m-teric $\boldsymbol{c}^i$ z ℓ vstavljanji, kjer $0 \leq \ell \leq m$. Število takih vektorjev $\boldsymbol{b}$ je kvečjemu

$$\left(\prod_{\ell=0}^{m} U_\ell^{I_\ell}\right) \exp_2\left(2N - \sum_{\ell=0}^{m}(m+\ell) I_\ell\right) = 2^N \prod_{\ell=0}^{m} \left(U_\ell \cdot 2^{-\ell}\right)^{I_\ell},$$

ker je $\sum_{\ell=0}^{m} I_\ell = M$ in zadnjih $2N - \sum_{\ell=0}^{m}(m+\ell)I_\ell$ bitov vektorja $\boldsymbol{b}$ lahko zavzame vrednosti iz množice $\{0,1\}$ neodvisno. Za fiksirane $I_0, I_1, \ldots, I_m$ imamo $M!/(I_1! \cdot I_2! \cdots I_m!)$ možnosti izbire števil bitov, ki jih vstavljamo v m-terke $\boldsymbol{c}^i$, kjer je $1 \le i \le M$. Če to seštejemo čez vse particije M:

$$\left\{(I_0, I_1, \ldots, I_m) \;\mid\; I_0, I_1, \ldots, I_m \ge 0,\ \sum_{\ell=0}^{m} I_\ell = M\right\},$$

dobimo zgornjo mejo za kardinalnost množice $\Phi(\boldsymbol{c})$,

$$|\Phi(\mathbf{c})| \le 2^N \sum \frac{M!}{I_0! I_1! \cdots I_m!} \prod_{\ell=0}^{m} \left(U_\ell \cdot 2^{-\ell}\right)^{I_\ell} = 2^{2N} \alpha_m^N, \tag{7.14}$$

torej za $N = mM$ je $P_N \le \alpha_m^N$. Naj bo $\boldsymbol{c}$' vektor dobljen z vstavljanjem poljubnega bita na začetek vektorja $\boldsymbol{c}$. Potem je

$$|\Phi(\mathbf{c}')| \le 4|\Phi(\mathbf{c})| \le 2^{2N+2} \cdot P_N, \tag{7.15}$$

iz česar sledi

$$P_{N+1} \le P_N.$$

Če združimo neenakosti (7.14) in (7.15), dobimo naslednjo neenakost:

$$P_N \le \alpha_m^{m\lfloor \frac{N}{m} \rfloor} \le \alpha_m^{N-m+1}. \tag{7.16}$$

Če postavimo $\alpha = \alpha_m$ in $\beta = \alpha_m^{1-m}$, za $m \ge 2$, je zgornja neenakost ravno to, kar smo želeli dokazati. ■

Opomba. Vrednosti za α_m, $2 \le m \le 7$, so naštete v tabeli 7.2. Iz (7.16) in (7.13) vidimo, da dobimo najboljšo zgornjo mejo za P_N, če vzamemo najmanjši α_m, tj. za $m = 7$. Če upoštevamo neenakost (7.16), potem velja

$$P_N \le 0.8768163^{7\lfloor N/7 \rfloor}. \tag{7.17}$$

7.3 Zgornja meja vložitvene verjetnosti $P_{1,X}(n,m)$

Približek za zgornjo mejo vložitvene verjetnosti smo izračunali zgoraj. V nadaljevanju se bomo lotili istega problema analitično in tako izračunali boljšo zgornjo mejo. Pri tem sledimo Golićevemu članku [9].

Naš cilj je izračunati verjetnost, da dani binarni niz X dolžine N lahko 1-vložimo v enakomerno porazdeljen naključen binaren niz Y dolžine $2N$. Naj X_i označuje prefiks niza X, dolžine i. Naj $A_{n,k}(X)$, $0 \le k \le n$ označuje število binarnih nizov dolžine $n+k$, v katere lahko strogo 1-vložimo binarni niz X dolžine n. Naj $A_{n,k}$ označuje maksimum vseh $A_{n,k}(X)$ po vseh X dolžine n in naj bo

$$A_n = \sum_{k=0}^{n} 2^{n-k} A_{n,k}. \tag{7.18}$$

Očitno A_n predstavlja zgornjo mejo za število binarnih nizov dolžine $2n$ v katere lahko 1-vložimo poljuben binaren niz dolžine n. To je samo zgornja meja, kajti možno je, da dan binaren niz 1-vložimo v dva binarna niza različnih dolžin, kjer je eden prefiks drugega. Torej je verjetnost, da poljuben binaren niz dolžine n 1-vložimo v enakomerno porazdeljen binaren niz dolžine $2n$, navzgor omejena z

$$P_n = 2^{-2n} A_n. \tag{7.19}$$

V nadaljevanju bomo rešili linearno rekurzijo za A_n in tako izračunali zgornjo mejo za P_n. Pred tem vpeljimo še $A_{n,k,i}(X)$, $i = 0, 1$, ki predstavlja število binarnih nizov Y dolžine $n+k$, za katere lahko binaren niz X 1-vložimo na prefiks Y_{n+k-i} in ne na $Y_{n+k} = Y$ za $i = 1$. Podobno, naj $A_{n,k,i}$ označuje maksimum vseh $A_{n,k,i}(X)$ po vseh X dolžine n. Očitno za vsak X velja:

$$A_{n,k}(X) = A_{n,k,0}(X) + A_{n,k,1}(X). \tag{7.20}$$

Zdaj, ko smo definirali osnovne pojme in simbole, lahko izračunamo vložitveno verjetnost.

Izrek 7.5. *Za poljuben binaren niz X dolžine N in vsak $n \in \{2, \ldots, N\}$ in $k \in \{0, 1, \ldots, n\}$ velja:*

$$A_{n,k,0}(X_n) = A_{n-1,k,0}(X_{n-1}) + A_{n-1,k,1}(X_{n-1}), \tag{7.21}$$

$$A_{n,k,1}(X_n) \leq A_{n,k-1,0}(X_n) + A_{n-1,k-1,1}(X_{n-1}), \tag{7.22}$$

kjer vzamemo $A_{n,k,i} = 0$, če je $k < 0$. V neenakosti (7.22) velja enačaj, če in samo če $x_{n-1} \neq x_n$, z začetnimi vrednostmi $A_{1,0,0}(X_1) = 1$, $A_{1,0,1}(X_1) = 0$, $A_{1,1,0}(X_1) = 0$, $A_{1,1,1}(X_1) = 2$, neodvisnimi od X_1.

Dokaz. Začetni pogoji sledijo direktno iz definicije $A_{n,k,i}(X)$. Enakost (7.21) sledi iz definicije $A_{n,k,0}(X_n)$, saj X_n lahko 1-vložimo na Y_{n+k} natanko tedaj, ko $y_{n+k} = x_n$ in X_{n-1} lahko strogo 1-vložimo v Y_{n+k-1}.

Po drugi strani pa iz $A_{n,k,1}(X_n)$ sledi naslednje. Če $y_{n+k} \neq x_n$, potem lahko X_n 1-vložimo na Y_{n+k-1} in ne na Y_{n+k} natanko tedaj, ko X_n lahko 1-vložimo na Y_{n+k-1}, s čimer dobimo prvi člen na desni strani neenakosti (7.22). Če $y_{n+k} = x_n$, potem lahko X_n 1-vložimo na Y_{n+k-1} in ne na Y_{n+k} natanko tedaj, ko je $y_{n+k-1} = x_n$ in X_{n-1} lahko 1-vložimo na Y_{n+k-3} in ne na Y_{n+k-2}, saj če lahko X_{n-1} 1-vložimo na Y_{n+k-2}, potem iz $y_{n+k} = x_n$ sledi, da lahko X_n 1-vložimo na Y_{n+k}. S tem dobimo drugi člen na desni strani neenakosti (7.22).

Enakost je posledica dejstva, da je ta pogoj potreben, ne pa tudi zadosten. Natančneje, če $x_{n-1} = x_n$ in X_{n-2} lahko strogo 1-vložimo v Y_{n+k-2}, potem lahko X_n 1-vložimo na Y_{n+k}, saj $y_{n+k} = x_n$ in $y_{n+k-1} = x_n = x - n - 1$. Če pa $x_{n-1} \neq x_n$, potem pa to ni mogoče in pogoj postane zadosten. To pomeni, da iz $y_{n+k} = x_n$ sledi, da X_n lahko 1-vložimo na Y_{n+k-1} in ne na Y_{n+k} natanko tedaj, ko $y_{n+k-1} \neq x_n$ in X_n lahko 1-vložimo na Y_{n+k-3} in ne na Y_{n+k-2}. Torej, če $x_{n-1} \neq x_n$, potem v neenakosti (7.22) velja enačaj. ■

Naslednja trditev karakterizira primer, ko v (7.22) velja enakost za vsak $n \in \{2, \ldots, N\}$ in vsak $k \in \{0, 1, \ldots, n\}$.

Trditev 7.6. *Naj bo X binaren niz dolžine N. Potem (7.22) velja z enačajem za vsak $2 \leq n \leq N$ in $0 \leq k \leq n$ natanko tedaj, ko je X alternirajoč niz.*

Dokaz. Če je niz X alternirajoč, potem velja $x_{n-1} \neq x_n$ za vsak $2 \leq n \leq N$, iz česar pa po konstrukciji dokaza izreka (7.5) sledi enačaj v neenakosti (7.22). Obrat sledi iz dokaza izreka 7.5. ■

Posledično, za alternirajoče nize sistem rekurzivnih neenačb danih v izreku (7.5) postane sistem dveh linearnih rekurzivnih enačb. Še več, ker so koeficienti v (7.20), (7.21) in (7.22) pozitivni in začetne vrednosti $A_{1,k,i}(X_1)$ neodvisne od X_1, sledi naslednja trditev.

Posledica 7.7. *Za $n \geq 1$, $0 \leq k \leq n$ in $i = 0, 1$ so maksimalne vrednosti $A_{n,k,i}$ in $A_{n,k}$ $A_{n,k,i}(X_n)$ in $A_{n,k}(X_n)$ po vseh X_n dolžine n dosežene, če je X_n alternirajoči in so določene z linearnimi rekurzijami*

$$A_{n,k,0} = A_{n-1,k,0} + A_{n-1,k,1}, \tag{7.23}$$

$$A_{n,k,1} = A_{n,k-1,0} + A_{n-1,k-1,1}, \tag{7.24}$$

za vsak $n \geq 2$ *in* $0 \leq k \leq n$, *z začetnimi vrednostmi* $A_{1,0,0} = 1$, $A_{1,0,1} = 0$, $A_{1,1,0} = 0$, $A_{1,1,1} = 2$ *in predpostavko, da je* $A_{n,k,i} = 0$ *za* $k < 0$, *skupaj z linearno enačbo*

$$A_{n,k} = A_{n,k,0} + A_{n,k,1} \tag{7.25}$$

■

S pomočjo rekurzivnih enačb (7.23), (7.24), (7.25) lahko hitro dobimo linearno rekurzivno enačbo za $A_{n,k}$.

Posledica 7.8. *Za vsak* $n \geq 3$ *in* $0 \leq k \leq n$ *velja*

$$A_{n,k} = A_{n-1,k} + 2A_{n-1,k-1} - A_{n-2,k-1}, \tag{7.26}$$

z začetnimi vrednostmi $A_{1,0} = 1$, $A_{1,1} = 2$, $A_{2,0} = 1$, $A_{2,1} = 3$, $A_{2,2} = 4$ *in predpostavko, da je* $A_{n,k} = 0$, *če* $k < 0$ *ali* $k > n$. ■

Iz posledice (7.8) skupaj z (7.18) dobimo rekurzivno enačbo za A_n, ki je ni težko rešiti.

Posledica 7.9. *Za vsak* $n \geq 3$ *velja*

$$A_n = 4A_{n-1} - 2A_{n-2}, \tag{7.27}$$

z začetnimi vrednostmi $A_1 = 4$ *in* $A_2 = 14$. *Rešitev rekurzivne enačbe* (7.27) *je dana z*

$$A_n = \frac{\sqrt{2}+1}{2}\left(2+\sqrt{2}\right)^n - \frac{\sqrt{2}-1}{2}\left(2-\sqrt{2}\right)^n, \tag{7.28}$$

za $n \geq 1$.

Dokaz. Vzemimo nastavek $A_n = \lambda^n$. Iz enačbe (7.27) dobimo

$$\lambda^n = 4\lambda^{n-1} - 2\lambda^{n-2}.$$

Enačbo delimo z λ^{n-2} in dobimo kvadratno enačbo

$$\lambda^2 = 4\lambda - 2.$$

Rešitvi te kvadratne enačbe sta

$$\lambda_1 = 2 - \sqrt{2}, \quad \lambda_2 = 2 + \sqrt{2}.$$

Rešitev naše rekurzije bo torej imela obliko

$$A_n = B\left(2-\sqrt{2}\right)^n + C\left(2+\sqrt{2}\right)^n.$$

Če upoštevamo še začetni vrednosti za $n = 1, 2$ dobimo rešitev

$$A_n = \frac{\sqrt{2}+1}{2}\left(2+\sqrt{2}\right)^n - \frac{\sqrt{2}-1}{2}\left(2-\sqrt{2}\right)^n.$$

■

Posledica (7.9) in formula (7.19) določata željeno zgornjo mejo za P_n, to je za vložitveno verjetnost v primeru $d = 1$. Meja je eksponentna za n.

Posledica 7.10.

$$P_n = \frac{\sqrt{2}+1}{2}\left(\frac{2+\sqrt{2}}{4}\right)^n - \frac{\sqrt{2}-1}{2}\left(\frac{2-\sqrt{2}}{4}\right)^n, \tag{7.29}$$

za $n \geq 1$. ■

Primerjajmo zdaj zgornjo mejo v (7.29) z zgornjo mejo (7.16). V prejšnjem razdelku smo pokazali, da za vsak $1 \leq m \leq n$ velja

$$P_n \leq P_m^{\lfloor n/m \rfloor} \leq \left(\frac{A_m^{1/m}}{4}\right)^{n-m+1}, \tag{7.30}$$

Zgornja meja (7.16) je preprosta posledica (7.18) in dejstva, da lahko vsako strogo 1-vložitev kokatenacije binarnih nizov lahko predstavimo kot kokatenacijo strogih 1-vložitev posameznih binarnih nizov in obratno. Za vsak $P_m < 1$ sledi, da je zgornja meja za vključitveno verjetnost eksponentna za n. V dokazu izreka 7.4 smo pokazali, da je $P_m < 1$. Najboljša zgornja meja je bila dobljena za $m = 7$ in je enaka

$$P_n \leq 0.8768163^{7\lfloor n/7 \rfloor}.$$

Če to primerjamo z zgornjo mejo (7.30), vidimo, da je slednja boljša.

Zgornjo mejo za A_n lahko še izboljšamo, če namesto (7.18) definiramo A'_n kot:

$$A'_n = \sum_{k=0}^{n} 2^{n-k} A_{n,k,1}, \tag{7.31}$$

ter upoštevamo $A'_n = A_n - 2A_{n-1}$, za $n \geq 2$ in $A'_1 = 2$. Ustrezno zgornjo mejo za vložitveno verjetnost

$$P'_n = 2^{-2n} A'_n$$

izračunamo podobno kot v (7.28), in dobimo naslednji rezultat.

Posledica 7.11.

$$P'_n = \frac{1}{2}\left(\frac{2+\sqrt{2}}{4}\right)^n + \frac{1}{2}\left(\frac{2-\sqrt{2}}{4}\right)^n, \tag{7.32}$$

za $n \geq 1$. ■

7.4 Kriptoanaliza 1-kontroliranega CCSR-generatorja

Zdaj, ko imamo izračunano oceno zgornje meje (7.16) oziroma zgornjo mejo (7.29) za vložitveno verjetnost 1-kontroliranega CCSR-generatorja, si oglejmo, kako ta rezultat vpliva na varnost oziroma uspešnost napadov opisanih v prejšnjem poglavju.

Uspešnost Živkovićevega algoritma

Živkovićev algoritem za rekonstrukcijo začetnega stanja CCSR-generatorja, če je dan del toka ključa $c = (c_1, \ldots, c_N)$, je uspešen, če je pričakovano število možnih začetnih stanj LFSR-generatorja malo. V tem razdelku bomo pokazali, da pričakovano število možnih začetnih stanj pada eksponentno z dolžino znanega dela toka ključev.

Če si ogledamo Živkovićevo oceno zgornje meje (7.16) oziroma zgornjo mejo (7.29) vključitvene verjetnosti, vidimo, da je le-ta eksponentno majhna za velike N. Konkretno, pri Živkovićevem

algoritmu to pomeni, da je verjetnost vstavitve zaporedja **c** v napačna mesta periode zaporedja **b** majhna, za dovolj velik N. S pomočjo izreka (7.4) lahko tudi izračunamo potrebno dolžino N danega toka ključa. Verjetnost, da **c** ni mogoče vložiti na poljubna napačna mesta v periodi izhodnega zaporedja generiranega z LFSR-generatorjem dolžine k, je navzdol omejena približno z

$$\left(1-\beta\alpha^N\right)^{\frac{K}{2}}, \tag{7.33}$$

kjer je $K=2^k$. Zgornjo oceno dobimo tako, da izračunamo verjetnost, da niza X dolžine N ne moremo vložiti v izhodno zaporedje LFSR-generatorja. Pri tem je možnih izhodnih zaporedij 2^k-1, torej toliko kot neničelnih začetnih stanj. $K/2$ vzamemo zaradi lažjega ocenjevanja dolžine N, potrebne za rekonstrukcijo začetnega stanja. Če $\beta\alpha^N \times K/2 \ll 1$ potem je ta meja večja od $1/2$, če $N > -(k+\beta)/\log_2\alpha \simeq -k/\log_2\alpha$. Za $\alpha=\alpha_7$, glej tabelo 7.2, dobimo pogoj

$$N > 5,3\cdot k.$$

Torej za rekonstrukcijo začetnega stanja LFSR-generatorja potrebujemo izhodno zaporedje CCSR-generatorja, ki je približno petkrat daljše od dolžine LFSR-generatorja. Seveda je ta pogoj potreben le za izračun začetnega stanja LFSR, če želimo izračunati še začetno stanje kontrolnega registra, ponavadi potrebujemo večji del toka ključev. Podobno oceno dobimo, če uporabimo kriterij (7.2) in oceno zgornje meje (7.17).

S kriterijem (7.2) in zgornjo mejo (7.29) lahko podobno izračunamo dolžino N znanega toka ključev, potrebno za uspešno rekonstrukcijo začetnega stanja. Tako mora dolžina N potrebna za rekonstrukcijo začetnega stanja zadoščati pogoju

$$N > 4,5\cdot k.$$

Ta ocena je boljša, kar je glede na boljšo zgornjo mejo za vložitveno verjetnost tudi pričakovano.

7.5 Kriptoanaliza CCSR z večkratnimi koraki

Do zdaj smo si ogledali uspešnost napadov na CCSR, ki je $\{1,2\}$-kontroliran. Pokazali smo, da število kandidatov za začetna stanja proizvajalnega registra ki generirajo zaporedje, v katerega lahko vložimo dan tok ključev, relativno pada proti 1, ko dolžina danega toka ključev raste. To smo pokazali tako, da smo izračunali zgornjo mejo za vložitveno verjetnost, ki z naraščanjem dolžine toka ključev pada eksponentno proti 0. Zdaj želimo analizirati uspešnost napada z vložitvami, če za d dopuščamo poljubne pozitivne celoštevilske vrednosti. To je, želimo ugotoviti, ali je mogoče posplošiti izrek (7.4) oziroma oceno (7.10) in uporabiti napad z vložitvami na $[1,d+1]$-kontroliran CCSR, kjer $a_i \in \{1,2,\ldots,d+1\}$, za poljuben d. V tem primeru iz niza B zbrišemo največ d bitov, preden izberemo naslednji člen toka ključev. Pristop, ki smo ga uporabili pri izreku 7.4, ne moremo posplošiti za poljuben d, saj je le-ta temeljil na direktnemu štetju, ki pa za splošen d nima smisla. V nadaljevanju bomo sledili članku Golić, O'Connor [10].

Glavni namen analize bo pokazati, da je minimalna dolžina opazovanega toka ključev n potrebna za rekonstrukcijo začetnega stanja $[1,d+1]$-kontroliranega CCSR približno

$$n \geq r\cdot\ln 2\cdot\left(1-2^{-d-2(d_1)4^{d+1}+(d+2)2^{d+2}}\right)\cdot 2^{(d+2)(1+d^{d+2})}, \tag{7.34}$$

kjer je r dolžina proizvajalnega registra. Posledično bomo pokazali, da z večanjem d ne moremo zagotoviti teoretične varnosti pred napadom z vložitvami, lahko pa popravimo praktično varnost CCSR-generatorja. Podobno, kot v primeru $d=1$, tudi tu predpostavimo, da je verjetnost vključitve niza C v niz B dobro aproksimirana z verjetnostjo vključitve naključnega niza X dolžine n v naključen vektor Y dolžine $m=n(d+1)$.

Problem vložitve

Osnovne definicije in problem smo že predstavili, zato bomo tu dokazali samo nekaj splošnih rezulatov, pomembnih za našo analizo, in vpeljali potrebne oznake.

Naj $P_{d,X}(n)$ označuje verjetnost, da dani binarni niz X dolžine n lahko d-vložimo v naključen binaren niz Y dolžine $m = n(d+1)$. Naj $P_{d,X}(n,k)$ označuje verjetnost, da X lahko strogo d-vložimo v naključen binaren niz Y dolžine $n+k$, kjer $0 \leq k \leq nd$. Očitno velja naslednja neenakost:

$$P_{d,X}(n) \leq P^*_{d,X}(n) = \sum_{k=0}^{nd} P_{d,X}(n,k). \tag{7.35}$$

Pri tem je $P^*_{d,X}(n)$ lahko večji od 1, za majhne n. Pri $n=1$ je $P^*_{d,X}(1) = d + 1/2$. Ustrezna zgornja meja za niz X dolžine n je potem:

$$P_{d,X}(n) \leq P^*_{d}(n) = \sum_{k=0}^{nd} P_{d}(n,k), \tag{7.36}$$

kjer $P_d(n,k)$ označuje maksimum vseh $P_{d,X}(n,k)$, ko X teče po vseh binarnih nizih dolžine n. Izračun zgornje meje $P_d(n,k)$ za poljuben X je precej težak. Zaenkrat je ta zgornja meja eksaktno izračunana samo za primer $d=1$, rezultat in metodo smo si ogledali v razdelku 7.3. Naš namen je dobiti podobno zgornjo mejo za $P_d(k,n)$. Naslednja lema je zelo uporabna pri nadaljnem računanju zgornje meje.

Lema 7.12. *Naj $X = X_1 \parallel X_2$ označuje stik nizov X_1 in X_2 zaporedoma dolžine n_1 in n_2. Dolžina niza X je $n = n_1 + n_2$. Potem velja:*

$$P^*_{d,X}(n) \leq P^*_{d,X_1}(n_1) \cdot P^*_{d,X_2}(n_2). \tag{7.37}$$

Dokaz. Za strogo d-vložitev lahko hitro izpeljemo naslednjo neenakost:

$$P_{d,X}(n,k) \leq \sum_{k_1} \sum_{k_2} P_{d,X_1}(n_1,k_1) \cdot P_{d,X_2}(n_2,k_2),$$

kjer indeksa k_1 in k_2 v vsoti tečeta pa vseh $0 \leq k_1 \leq dn_1$ in $0 \leq k_2 \leq dn_2$, tako da je $k = k_1 + k_2$. Upoštevamo neenakost (7.35) in dobimo

$$P^*_{d,X}(n) \leq \sum_{k=0}^{nd} \sum_{k_1} \sum_{k_2} P_{d,X_1}(n_1,k_1) \cdot P_{d,X_2}(n_2,k_2) = P^*_{d,X_1}(n_1) \cdot P^*_{d,X_2}(n_2).$$

Zadnjo enakost dobimo, če vsote razpišemo in upoštevamo definicijo $P_{d,X}(n)$ v 7.35. ■

Analogni rezultat za $P_{d,X}(n)$ ne velja. Lema 7.12 nam torej omogoča preoblikovanje zgornje meje (7.35) v zgornjo mejo za nize oblike $X = X_1 \parallel X_2$. V nasprotju z zgornjo mejo za $P_{d,X}(n)$ pa lahko spodnjo mejo izračunamo dokaj enostavno.

Lema 7.13. *Za vsak X velja*

$$P_{d,X}(n) \geq \left(1 - \frac{1}{2^{d+1}}\right)^n. \tag{7.38}$$

■

Rešitev za poljuben d

V tem razdelku je prikazana teoretična metoda za reševanje splošnega problema vložitve z največ d brisanji. Glaven namen tu je ugotoviti, ali lahko deli in vladaj napad z omejeno vložitvijo preprečimo, če izberemo dovolj velik d. Ena možnost bi bila analitični izračun $P_d(n,k)$, podobno kot v primeru za $d=1$ in potem uporaba ocene (7.36). Ta metoda je precej zahtevna, saj je težko ugotoviti obnašanje $P_{d,X}(n,k)$ za poljuben niz X.

Metoda, ki je prikazana tu, je sestavljena iz dveh stopenj in temelji na konstantnih in alternirajočih nizih.

Definicija 7.14. *Niz X je* **konstanten**, *če je en bit ponovljen po celi dolžini niza X. Konstantne nize označujemo 0^n in 1^n, kjer je 0^n niz n-tih ničel, 1^n pa niz n-tih enic. Niz X je* **alternirajoč**, *če je vsak bit x_i komplement prejšnjega bita x_{i-1}.*

Naj $P_{d,a}(n,k)$ in $P_{d,c}(n,k)$ označujeta zaporedoma vložitveno verjetnost za alternirajoče in vložitveno verjetnost za konstantne nize. Najprej bomo izračunali zgornji meji za verjetnosti $P_{d,a}(n,k)$ in $P_{d,c}(n,k)$. V drugi stopnji bomo uporabili lemo 7.12 in neenakost (7.36) ob predpostavki, da poljuben niz X lahko razbijemo na konstantne in alternirajoče podnize. Tako bomo dobili eksponentno zgornjo mejo, ki velja za dovolj dolge nize z verjetnostjo skoraj 1.

Konstantni in alternirajoči nizi

Da bi dobili zgornji meji za $P_{d,a}(n,k)$ in $P_{d,c}(n,k)$, si najprej oglejmo naslednjo lastnost. Če konstanten niz $X=0^n$ lahko strogo d-vložimo v niz Y dožine $m \geq n$, potem Y ne vsebuje podniza 1^{d+1}. Analogno ista ugotovitev velja za niz $X=1^n$ in podniz 0^{d+1}. Če alternirajoči niz X dolžine n lahko strogo 1-vložimo v niz Y dolžine $m \geq n$, potem Y ne vsebuje nobenega od podnizov $0^{2(d+1)}$ ali $1^{2(d+1)}$. Če preštejemo vse binarne nize, ki imajo zgornje lastnosti, lahko izračunamo zgornjo mejo za $P_{d,a}(n,k)$ in $P_{d,c}(n,k)$. To bomo naredili z uporabo teorije formalnih jezikov in rodovnimi funkcijami.

Oglejmo si najprej konstantne nize. Množica binarnih nizov, ki se začnejo z 0 in ne vsebujejo niza 1^ℓ kot podniza, za $\ell = d+1$, je za fiksen d regularen jezik (3.3), ki ga označimo kot $\boldsymbol{L_\ell^c}$. Po izreku 3.11 obstaja deterministično končen avtomat (DKA), ki prepozna oziroma sprejme dani jezik. Čeprav tak DKA ni enolično določen, lahko za dani DKA iz jezika L_ℓ^c, ki ga le-ta sprejema, po izreku 3.11 konstruiramo regularni izraz. Od vseh regularnih izrazov, si oglejmo naslednjega:

$$L_\ell^c = (0 + 01 + \cdots + 0\underbrace{11\ldots11}_{\ell-1})^*. \tag{7.39}$$

Torej vsak niz $X \in L_\ell^c$ lahko dobimo s ponavljanjem stikov nizov iz množice $\{0\} \cup \{01\} \cup \cdots \cup \{011\ldots11\}$, saj operator * (iteracija) pomeni: " izberi ničkrat ali večkrat". Prazen niz ϵ dolžine nič je prav tako vključen. Pomembna lastnost tega je, da vsak niz $X \in L_\ell^c$ enolično razstavimo v podnize $0, 01, 011, \ldots, 011\ldots11$, ki definirajo regularni izraz (7.39) za jezik L_ℓ^c.

Primer. Oglejmo si naslednji niz: 01010010000100. Ta niz ne vsebuje podniza 11 in je torej element jezika L_2^c. Sestavljen je enolično iz regularnega izraza oblike (7.39) in sicer kot

$$01 \mid 01 \mid 0 \mid 01 \mid 0 \mid 0 \mid 0 \mid 01 \mid 0 \mid 0,$$

kjer je $\mid$ uporabljen za ločitev elementov iz regularnega izraza. •

Enolična dekompozicija nam omogoča, da jezik L_ℓ^c oštevilčimo z uporabo rodovne funkcije

$$\frac{1}{1-z} = \sum_{i \geq 0} z^i.$$

Najprej vpeljimo naslednjo oznako.

Definicija 7.15. *Naj bo $G(x) = \sum_i a_i z^i$ rodovna funkcija. Potem n-ti koeficient v razvoju rodovne funkcije $G(x)$ označimo z $[z^n]$.*

Primer. Naj bo

$$(1+x)^m = \sum_{k=0}^{m} \binom{m}{k} z^k$$

rodovna funkcija. Potem je koeficient $[z^n]$ rodovne funkcije $(1+x)^n$ enak

$$\binom{m}{n}.$$

•

Lema 7.16. *Naj $C_\ell(n)$ označuje število nizov dolžine $n \geq 0$, sestavljenih iz besed jezika L_ℓ^c. Potem je $C_\ell(n)$ enak koeficientu $[z^n]$ rodovne funkcije*

$$\mathcal{G}_\ell^c(z) = \frac{1}{1-(z+z^2+\cdots+z^{\ell-1}+z^\ell)} = \frac{1}{1-\sum_{i=1}^{\ell} z^i}. \tag{7.40}$$

Dokaz. Rodovna funkcija za $(0+01+\cdots+0\underbrace{11\ldots11}_{\ell-1})^i$, glede na dolžino je

$$(z+z^2+\cdots+z^l)^i.$$

Število nizov dolžine n sestavljenih iz besed L_ℓ^c je potem enako koeficientu $[z^n]$, vrste

$$\sum_{i\geq 0} (z+z^2+\cdots+z^\ell)^i = \frac{1}{1-\sum_{i=1}^{\ell} z^i}. \tag{7.41}$$

■

Oglejmo si zdaj alternirajoče nize. Množica binarnih nizov, ki se začnejo z 0 in ne vsebujejo nizov 1^ℓ in 0^ℓ kot podnizov, kjer $\ell = 2(d+1)$, je regularen jezik za fiksen d, ki ga označimo z L_ℓ^a. Jezik L_ℓ^a je generiran z naslednjim regularnim izrazom:

$$L_\ell^a = \left((0+00+\cdots+\underbrace{00\ldots00}_{\ell-1})(1+11+\cdots+\underbrace{11\ldots11}_{\ell-1})\right)^* (\epsilon+0+00+\cdots+\underbrace{00\ldots00}_{\ell-1}). \tag{7.42}$$

Podobno kot pri L_ℓ^c tudi tu lahko hitro preverimo, da ima ta regularen izraz za L_ℓ^a enolično dekompozicijo, kar nam omogoča direktno številčenje elementov.

Lema 7.17. *Naj $A_\ell(n)$ označuje število nizov dolžine $n \geq 0$, sestavljenih iz besed jezika L_ℓ^a. Potem je $A_\ell(n)$ enak koeficientu $[z^n]$ rodovne funkcije*

$$\mathcal{G}_\ell^a(z) = \frac{1}{1-\sum_{i=1}^{\ell-1} z^i}. \tag{7.43}$$

Dokaz. Rodovna funkcija glede na dolžino niza definiranega z

$$\left((0+\cdots+\underbrace{00\ldots00}_{\ell-1})(1+\cdots+\underbrace{11\ldots11}_{\ell-1})\right)^{i}(\epsilon+0+\cdots+\underbrace{00\ldots00}_{\ell-1})$$

je

$$\left(\sum_{j=1}^{\ell} z^j\right)^{2i} \cdot \sum_{j=0}^{\ell} z^j. \tag{7.44}$$

Število nizov dolžine n sestavljenih iz besed L_ℓ^a je potem enako koeficientu $[z^n]$, vrste

$$\sum_{j\geq 0}\left(\sum_{j=1}^{\ell} z^j\right)^{2i} \cdot \sum_{j=0}^{\ell} z^j = \frac{1+\sum_{j=1}^{\ell} z^j}{1-\left(\sum_{j=1}^{\ell} z^j\right)^2} = \frac{1}{1-\sum_{i=1}^{\ell-1} z^i}.$$

■

Vidimo, da imata rodovni funkciji za L_ℓ^c in L_ℓ^a isto obliko, razlikujeta se le v polinomu imenovalca. To lastnost lahko uporabimo v naslednjem izreku, ki nam bo dal zgornji meji za verjetnosti $P_{d,a}(n,k)$ in $P_{d,c}(n,k)$. Več o rodovnih funkcijah bralec lahko najde v [25].

Izrek 7.18. *Naj bo $C_m(n)$ koeficient $[z^n]$ v razvoju rodovne funkcije*

$$G_m(z) = \frac{1}{1-\sum_{i=1}^{m} z^i}.$$

Za vložitveni verjetnosti $P_{d,c}(n,k)$ in $P_{d,a}(n,k)$ velja:

$$P_{d,c}(n,k) \leq 2^{-(n+k)} \cdot C_{d+1}(n+k), \tag{7.45}$$

$$P_{d,a}(n,k) \leq 2^{-(n+k)} \cdot C_{2d+1}(n+k). \tag{7.46}$$

Dokaz. Vložitvena verjetnost za konstantne nize je enako kvocientu števila binarnih nizov dolžine $n+k$, v katere lahko d-vložimo konstantni niz X, dolžine n in števila vseh binarnih nizov dolžine $n+k$. Naj bo $\ell = d+1$. Z uporabo leme 7.16 dobimo prvo neenakost v (7.45). Podobno naredimo za (7.46). Naj bo zdaj $\ell = 2(d+1)$. Z uporabo leme 7.17 dobimo drugo neenakost in izrek je dokazan. ■

Zdaj nam preostane le še ocena števila $C_m(n)$, tj. koeficienta $[z^n]$ v razvoju rodovne funkcije

$$G_m(z) = \frac{1}{1-\sum_{i=1}^{m} z^i},$$

ki jo poda naslednji izrek.

Izrek 7.19. *Naj bo $C_m(n)$ koeficient $[z^n]$ v razvoju rodovne funkcije*

$$G_m(z) = \frac{1}{1-\sum_{i=1}^{m} z^i}.$$

Potem za vsak $n \geq 0$ in $m \geq 2$ velja

$$\frac{C_m(n)}{2^n} < \left(1-\frac{1}{2^{m+1}}\right)^{n-m+1}. \tag{7.47}$$

Skica dokaza. Izreka ne bomo dokazali v celoti. V dokazu uporabimo sredstva funkcionalne analize, ki so na voljo v [26].

Naj bo $P_m(z)$ recipročni polinom rodovne funkcije

$$G_m(z) = \frac{1}{1 - \sum_{i=1}^{m} z^i},$$

tj.

$$P_m(z) = z^m - \sum_{i=0}^{m-1} z^i.$$

$C_m(n)$ lahko eksplicitno izrazimo s koreni $\alpha_1, \ldots, \alpha_m$ recipročnega polinoma $P_m(z)$, glej [25]. V primeru $m = 2$, $P_2(z)$ lahko reduciramo na recipročni polinom, ki ustreza rodovni funkciji za Fibonacci-jeva števila, ki je

$$f(z) = \frac{1}{1 - z - z^2}.$$

Če pa je $m > 2$, potem ni mogoče dobiti analitičnega izraza za korene polinoma $P_m(z)$, zato moramo uporabiti numerično aproksimacijo. Asimptotično obnašanje $C_m(n)$ je dominirano s koreni največje stopnje, kar pa v našem primeru ni zadovoljivo, saj nas zanima aproksimacija zgornje meje za $C_m(n)$, ki bo veljala za vse vrednosti n. S funkcionalno analizo se da pokazati, da ima za $m \geq 2$ polinom $P_m(z)$ pozitivne realne korene β_m, tako da velja

$$1 < \beta_m < 2 - \frac{1}{2^m}. \tag{7.48}$$

Torej je zaporedje $\{c_m \beta_m^n\}_{n=0}^{\infty}$, kjer je c_m poljubna pozitivna konstanta, rešitev linearne rekurzije, določene s polinomom $P_m(z)$, ki je

$$x_n = \sum_{i=1}^{m} x_{n-i}, \tag{7.49}$$

za $n \geq m$. Zaporedje $\{C_m(n)\}_{n=0}^{\infty}$, ki je tudi pozitivna rešitev rekurzije (7.49), ima začetne vrednosti $C_m(0) = 1$ in $C_m(n) = 2^{n-1}$, za $1 \leq n \leq m-1$, saj številči nize sestavljene iz besed jezika L_m^c. Če je izbrana konstanta c_m dovolj velika, da zadošča pogoju

$$C_m(n) \leq c_m \beta_m^n,$$

za $0 \leq n \leq m-1$, potem velja $C_m(n) \leq c_m \beta_m^n$ za vse n, ker ima rekurzija (7.49) pozitivne koeficiente. Zgornjemu pogoju je zadoščeno, če za konstanto c_m izberemo

$$c_m = \left(\frac{2}{\beta_m}\right)^{m-1}.$$

Torej za vsak $n \geq 0$ in $m \geq 2$ velja

$$\frac{C_m(n)}{2^n} \leq \left(\frac{\beta_m}{2}\right)^{n-m+1} < \left(1 - \frac{1}{2^{m+1}}\right)^{n-m+1}. \tag{7.50}$$

■

Zdaj, ko imamo oceni za $P_{d,c}(n,k)$ in $P_{d,a}(n,k)$, lahko ocenimo $P_{d,c}^*(n)$ in $P_{d,a}^*(n)$. Velja naslednji izrek.

Izrek 7.20. *Naj $P^*_{d,c}(n)$ in $P^*_{d,a}(n)$ zaporedoma označujeta zgornji meji vključitvene verjetnosti za konstantni in alternirajoči niz, definirani v (7.35). Potem za vsak $n \geq 1$ velja:*

$$P^*_{d,c}(n) < 2^{d+2}\left(1 - \frac{1}{2^{d+2}}\right)^{n-d} \tag{7.51}$$

$$P^*_{d,a}(n) < 4^{d+1}\left(1 - \frac{1}{4^{d+1}}\right)^{n-2d}. \tag{7.52}$$

Dokaz. Glede na definicijo $P^*_{d,c}(n)$ v (7.35) lahko zapišemo

$$P^*_{d,c}(n) = \sum_{k=0}^{nd} P_{d,c}(n,k).$$

Če zdaj združimo neenakost (7.45), izrek 7.19 in seštejemo po k, neenakost (7.51) očitno sledi. Podobno ravnamo v primeru alternirajočega niza, da dobimo neenakost (7.52). ■

Obe zgornji meji, (7.51) in (7.52) sta za majhne vrednosti n, večji od ena, odvisno od d. Za velike n pa konvergirata proti nič eksponentno. Zgornja meja v (7.51) je relativno blizu spodnji meji, izračunani v lemi 7.13, medtem ko je zgornja meja (7.52) za alternirajoče nize precej večja od obeh.

Naključni nizi

Predpostavimo, da smo pokazali, da je alternirajoči niz najbolj verjeten kandidat za vključevanje v naključen niz. Eksperimentalni rezultati, dobljeni z direktnim štetjem, potrjujejo to dejstvo. Natančneje, za $d = 2$ in za $2 \leq n \leq 7$ in $0 \leq k \leq 2n$, so maksimalne vrednosti $P_2(n,k)$ stroge vključitvene verjetnosti $P_{2,X}(n,k)$ dobljene za alternirajoč niz X, glej Golić, O'Connor [10]. To nas pripelje do domneve, da alternirajoči nizi maksimizirajo strogo vključitveno verjetnost za poljubne d, n in k. Ta domneva za splošen d ni dokazana, v primeru $d = 1$ smo jo pokazali v razdelku 7.3. Če to domnevo sprejmemo, potem zgornja meja (7.52) velja za vse nize X. Če uporabimo kriterij (7.2), dobimo naslednjo neenakost:

$$2^r \cdot 4^{d+1}\left(1 - \frac{1}{4^{d+1}}\right)^{n-2d} \leq 1. \tag{7.53}$$

Iz neenakosti (7.53) tako dobimo približno oceno potrebne dolžine n

$$n \geq (r + 2(d+1)) \cdot \ln 2 \cdot 4^{d+1} \tag{7.54}$$

opazovanega izhodnega zaporedja za uspešno rekonstrukcijo začetnega stanja. To lahko primerjamo s spodnjo mejo dobljeno v lemi 7.13. Po tej lemi velja, če dolžina n opazovanega izhodnega zaporedja približno zadošča neenakosti

$$n \leq r \cdot \ln 2 \cdot 2^{d+1}, \tag{7.55}$$

potem uspešna rekonstrukcija začetnega stanja ni mogoča.

Seveda pa ocena (7.54) temelji na domnevi, da je zgornja meja za alternirajoče nize tudi zgornja meja za ostale nize, kar pa se zdi teoretično težko dokazljivo. Zato raje uporabimo dejstvo, da lahko vsak binaren niz razdelimo na konstantne in alternirajoče podnize. Tako lahko uporabimo izrek 7.20 v luči leme 7.12. Ker sta zgornji meji v (7.51) in (7.52) za majhne n

večji od ena in zato neuporabni, želimo najti vse zadosti dolge konstantne in alternirajoče nize. Najprej želimo najti minimalne vrednosti za n, tako da bosta desni strani neenakosti (7.51) in (7.52) manjši od ena. Če za $m = d + 2$ in $m = 2(d + 1)$ velja

$$n \geq k_m = m2^m, \tag{7.56}$$

potem ni težko preveriti, da sta meji v (7.51) in (7.52) zaporedoma manjši od ena. Zdaj lahko formuliramo in dokažemo naslednji izrek.

Izrek 7.21. *Naj bo X naključen niz dolžine n in $k_m = m2^m$. Potem za dovolj velik n, z verjetnostjo blizu 1, velja*

$$P^*_{d,X}(n) < \left[\left(1 - \frac{1}{2^{d+2}}\right)^{2^{-k_{d+2}}} \cdot \left(1 - \frac{1}{4^{d+1}}\right)^{2^{-k_{2d+2}}}\right]^n. \tag{7.57}$$

Dokaz. Razdelimo binaren niz X dolžine n v konstantne podnize samih ničel in enic. Zaporedne podnize dolžine ena združimo v alternirajoče podnize. Taka delitev binarnega niza je očitno enolična. Zaradi (7.56) štejejo samo konstantni podnizi dolžine vsaj $(d + 2)2^{d+2}$ in alternirajoči podnizi dolžine vsaj $2(d + 1)4^{d+1}$. Če je niz X poljuben, potem so možni vsi primeri. Če pa je X naključen niz in n dovolj velik, potem je z verjetnostjo blizu ena približno $n/2^i$ konstantnih podnizov dolžine vsaj i, $i \geq 1$, in $n/(2^{i+2})$ alternirajočih podnizov dolžine vsaj i, $i \geq 2$. Torej sledi, da je število bitov vsebovanih v konstantnih podnizih dolžine vsaj i, $i \geq 1$, enako $(i+1)/2^i$, medtem ko je število bitov vsebovanih v alternirajočih podnizih dolžine vsaj i, $i \geq 2$, enako $(i + 1)/2^{i+1}$. Z uporabo leme 7.12 in izreka 7.20, trditev sledi. ■

Izrek 7.21 nam pove, da za dani naključni niz X, verjetnost, da X lahko d-vložimo v naključen niz Y, eksponentno pada proti nič z dolžino niza. Ker se izhodno zaporedje CCSR-generatorja obnaša kot naključno zaporedje, lahko ta rezultat uporabimo v "deli in vladaj" napadu z omejeno vložitvijo. Če uporabimo kriterij (7.2), potem lahko hitro izračunamo, dolžino n znanega toka ključev, potrebno za uspešno rekonstrukcijo začetnega stanja, ki je enaka

$$n \geq r \cdot \ln 2 \cdot \left(1 - 2^{-d-2(d+1)4^{d+1}+(d+2)^{d+2}}\right) \cdot 2^{(d+2)(1+2^{d+2})}. \tag{7.58}$$

Vidimo, da je minimalna potrebna dolžina izhodnega zaporedja CCSR-generatorja linearna v dolžini registra r in supereksponentna v d. To je posledica teoretičnega pristopa. V praksi pa se izkaže, da se minimalna dolžina opazovanega izhodnega zaporedja obnaša podobno, kot smo to izračunali za alternirajoče nize v (7.54), kjer je spodnja meja linearna v r in eksponentna v d, glej Golić, O'Connor [10].

7.6 Kriptoanaliza d-kontroliranega CCSR-generatorja

Če posplošimo Živkovićev algoritem za poljuben d, kot smo to naredili v prejšnjem poglavju in uporabimo izračunane zgornje meje za vložitveno verjetnost, izračunane zgoraj (7.51), (7.52), (7.57), in spodnji meji za dolžino opazovanega izhodnega zaporedja (7.54), (7.58), ki sta linearni v dolžini registra r in eksponentni oziroma superekspononenti v d, vidimo, da z večanjem d-ja ne moremo zagotoviti teoretične varnosti d-kontroliranega CCSR-generatorja pred napadom z omejeno vložitvijo. Lahko pa izboljšamo praktično varnost. Če upoštevamo rezultate dobljene v praksi, ki potrjujejo domnevo, da zgornja meja za alternirajoče nize velja tudi za ostale, potem je spodnja meja za dolžino opazovanega toka ključev linearna v r in eksponentna v d, kar pomeni, da je napad z omejeno vložitvijo smiselen, če le d ni prevelik. Torej je za varnost pred takim napadom, velikost števil d pomembna, saj nam le-ta zagotavlja praktično varnost.

Poglavje 8

ZAKLJUČEK

To delo je obravnavalo simetrično kriptografijo, natančneje tokovne šifre in njihove osnovne gradnike. Pri LFSR-generatorju smo si ogledali osnovne lastnosti pomičnih registrov. Pri tem smo največ pozornosti namenili periodi in linearni zahtevnosti. Pokazali smo, da je perioda najdaljša, če je povratni polinom primitiven. Opisali smo algoritem za konstrukcijo najkrajšega LFSR-generatorja, ki generira dano zaporedje. Zaradi linearnosti LFSR-generatorja smo opisali CCSR, ki je s kriptografskega stališča varnejši. Tu smo s podobnimi metodami izračunali periodo in linearno zahtevnost. Nadaljevali smo z možnimi napadi na CCSR, kjer smo podrobneje predstavili napad z vložitvami. Nato smo analizirali varnost CCSR-generatorjev pred takimi napadi. Tu smo največ pozornosti namenili računanju vložitvene verjetnosti. Pri tem smo omenili, da analitična zgornja meja za splošen d ni izračunana, torej je to še odprt problem. Kljub temu pa smo z aproksimacijo zgornje meje za vložitveno verjetnost pokazali, da je dolžina znanega ključa, potrebna za rekonstrukcijo začetnega stanja, linearna v dolžini registra r in superekseponentna v d. Rezultati v praksi so pokazali, da je le-ta linearna v r in eksponentna v d, kar bi bilo res, če bi zgornjo mejo maksimizirali alternirajoči nizi. Za $d = 1$ to velja in smo tudi dokazali, za $d > 1$ pa to še ni dokazano. Zaradi tega je analitičen izračun vložitvene verjetnosti v primeru $d = 2$ pomemben. Dokaz, ki bi maksimiziral zgornjo mejo za alternirajoče nize, bi namreč okrepil domnevo, da to velja za vsak d.

Literatura

[1] E.R. Berlekamp, *Algebraic Coding Theory*, McGraw-Hill Book Company, 1968.

[2] T. Beth, F.C. Piper, *The stop-and-go generator*, Advances in cryptology: Proc. EUROCRYPT '84, *Lecture Notes in Computer Science*, **209** (1985), str.88-92.

[3] W. G. Chambers, *Clock-controlled shift registers in binary sequence generators*, IEE Proceedings E., **135** (1988), str.17-24.

[4] W. G. Chambers, D. Gollman, *Lock-in effect in cascades of clock-controlled shift registers*, Advances in Cryptology, EUROCRYPT '87, *Lecture Notes in Computer Science*, **330** (1988), str.331-342.

[5] T. W. Cusick, C. Ding, A. Renvall, *Stream ciphers and number theory*, Elsevier Science B.V., Amsterdam, 1998.

[6] D. W. Davies, *The Siemens and Halske T52e cipher machine*, Cryptologia, **10** (1986), str.289-307.

[7] C. Ding, *The Differential Cryptanalysis and Design of Natural Stream Ciphers*, Fast software encryption, *Lecture Notes in Computer Science*, **809** (1994), str.101-115.

[8] C. Ding, G. Xiao, W. Shan, *The Stability Theory of Stream Ciphers*, Lecture Notes in Computer Science **561**1991

[9] J. Dj. Golić, *Constrained embedding probability for two binary strings*, SIAM Journal on Discrete Mathematics, **9/3** (1996), str.360-364.

[10] J. Dj. Golić, L. O'Connor, *A Cryptanalysis of Clock-Controlled Shift Registers with Multiple Steps*, Cryptography: policy and algorithms (Brisbane, 1995) *Lecture Notes in Computer Science*, **1029** (1996), str.174-185.

[11] J. Dj. Golić, L. O'Connor, *Embedding and probabilistic Correlation Attacks on Clock-controlled Shift Registers*, Advances in Cryptology-EUROCRYPT '94, *Lecture Notes in Computer Science*, **950** (1995), str.230-243.

[12] J. Dj. Golić, M. J. Mihaljević, *A generalized correlation attack on a class of stream ciphers based on the Levensthein distance*, Journal of Cryptology, **3(3)** (1991), str.201-212.

[13] J. Dj. Golić, *On the Linear Complexity of Functions of Periodic GF(q) Sequences*, IEEE Transactions on Information Theory, **35** (1989), str.69-75.

[14] J. Dj. Golić, M. J. Mihaljević, *A fast iterative algorithm for shift register initial state reconstruction given the noisy output sequences*, Advances in Cryptology - AUSCRYPT '90, *Lectur Notes in Computer Science*, **453** (1990), str.165-175.

[15] D. GOLLMAN, W. G. CHAMBERS, *Clock controlled shift register: a review*, IEEE Journal on Selected Areas in Communications, **7(4)** (1989), str.525-533.

[16] D. GOLLMAN, *Linear recursions of cascaded sequences*, Contributions to General Algebra 3, Proceedings of the Vienna Conference June 1984. Verlag Holder-Pichler-Tempsky, Wien 1985

[17] G. R. GRIMMETT, D. R. STIRZAKER, *Probability and Random Processes*, 2.nd ed., Clarendon Press, Oxford, 1992.

[18] C. G. GÜNTHER, *Alternating step generators controlled by de Bruijn sequences*, Proceedings of EUROCRYPT '87, *Lecture Notes in Computer Science*, **309** (1988), str.5-14.

[19] J. E. HOPCROFT, J.D.ULLMAN, *Introduction to automata, theory, languages and computation*, Addison-Wesley Publishnig, 1979.

[20] A. JURIŠIĆ, A. MENEZES, *Eliptic Curves and Cryptography*, Dr. Dobb's Journal, **264** (1997), str.26-36.

[21] R. LIDL, H.NIEDERREITER, *Introduction to finite fields and their applications*, Cambridge University Press, 1986.

[22] J. C. LAGARIAS, *Pseudorandom number generators in cryptography and number theory*, Cryptography and Computational Number Theory; *Proceeding of Symposia in Applied Mathematics*, **42** (1990), str.115-143.

[23] J. L. MASSEY, *Shift-Register Synthesis and BCH Decoding*, IEEE Transactions on Information Theory, **IT-15, No.1** (1969), str.122-127.

[24] A. J. MENEZES, P. C. VAN OORSCHOT, S. A. VANSTONE, *Handbook of Applied Cryptography*, CRC Press LLC, 1997.

[25] F. ROBERTS, *Applied Combinatorics*, Englewood Cliffs, NJ: Prentice Hall, 1984.

[26] W.RUDIN, *Functional Analysis*, McGraw-Hill,1991.

[27] R. A. RUEPPEL, *Analysis and Design of Stream Ciphers*, Heidelberg: Springer-verlag, 1986.

[28] T. SIEGENTHALER, *Decrypting a Class of Stream Ciphers Using Ciphertext Only*, IEEE Transactions on Computers, **c-34** (1985), str.81-85.

[29] D. R. STINSON, *Cryptography: Theory and Practice*, CRC Press, 1995.

[30] D. WELSH, *Codes and Cryptography*, Okford Science Publications, 1988.

[31] M. V. ŽIVKOVIĆ, *An algorithm for the initial state reconstruction of the clock-controlled shift register*, IEEE Transactions on Information Theory, **37/6** (1991), str.1488-1490.

[32] http://csrc.nist.gov/encryption/aes/round2/r2report.pdf