SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Evidentné číslo: FEI-5382-5941

**KONEČNÉ AUTOMATY A LATIN ARRAYS**

**BAKALÁRSKA PRÁCA**

2013 Juraj Charvát

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

**KONEČNÉ AUTOMATY A LATIN ARRAYS**

**BAKALÁRSKA PRÁCA**

FEI-5382-5941

Študijný program: aplikovaná informatika

Číslo študijného odboru: 2511

Názov študijného odboru: 9.2.9 aplikovaná informatika

Školiace pracovisko: Ústav informatiky a matematiky

Vedúci záverečnej práce/školiteľ: prof. RNDr Otakar Grošek, PhD.

|  |  |  |
| --- | --- | --- |
| **Slovenská technická univerzita v Bratislave** **Ústav informatiky a matematiky** |  | **Fakulta elektrotechniky a informatiky** **2012/2013** |
|  | | |
| http://is.stuba.sk/img.pl?unid=157787 | | |
|  | | |
| **ZADANIE BAKALÁRSKEJ PRÁCE** | | |

|  |  |  |
| --- | --- | --- |
| Evidenčné číslo: | | FEI-5382-5941 |
| ID študenta: | | 5941 |
| Autor práce: | | Juraj Charvát (5941) |
| Študijný program: | | Aplikovaná informatika |
| Študijný odbor: | | 9.2.9 aplikovaná informatika |
|  | | |
| Vedúci práce: | | prof. RNDr. Otokar Grošek, PhD. |
|  | | |
| Miesto vypracovania: | | Ústav informatiky a matematiky |
|  | | |
| Názov témy: | **Konečné automaty a Latin arrays** | |

|  |  |  |
| --- | --- | --- |
| Špecifikácia zadania: | | |
|  | Symetrické šifry obsahujú aj algoritmus na rozvoj kľúča, ktorý je často zdrojom informácie o správaní sa šifry. Preto boli navrhnuté symetrické šifry na báze Latin arrays, ktoré nepotrebujú takýto algoritmus. Cieľom práce je naštudovať túto problematiku a zrealizovať takéto šifrovanie.  Úlohy: 1. Naštudujte problematiku invertovateľných automatov na báze Latin arrays. 2. Naštudujte problematiku bezpečnosti takejto šifry. 3. Vytvorte počítačovú aplikáciu na šifrovanie a dešifrovanie pre takýto šifrátor. |  |
|  | | |
| Literatúra: | | |
| 1, A. Menezes, P. van Oorschot, S. Vanstone: Handbook of Applied Cryptography.  2, Renji Tao: Finite Automata and Application to Cryptography. | | |

Dátum zadania bakalárskej práce: 24. 09. 2012

Termín odovzdania bakalárskej práce: 24. 05. 2013

Juraj Charvát

študent

prof. RNDr. Otokar Grošek, PhD. prof. RNDr. Gabriel Juhás, PhD.

vedúci pracoviska garant študijného programu

SLOVENSKÁ TECHNICKÁ UNIVERZITA v Bratislave

FAKULTA ELEKTROTECNIKY A INFORMATIKY

**Abstrakt**

Názov školy: Slovenská technická univerzita

Názov ústavu: Ústav informatiky a matematiky

Študijný program: Aplikovaná informatika

Autor: Juraj Charvát

Názov bakalárskej práce: Konečné automaty a latin arrays

Kľúčové slová:

SLOVAK UNIVERSITY OF TECHNOLOGY  BRATISLAVA

FACULTY OF ELECTRICAL ENGINEERING AND INFORMATION TECHNOLOGY

**Abstract**

School name: Slovak University of Technology

Faculty name: Department of Applied informatics and Information Technology

Study program: Applied informatics

Author: Juraj Charvát

Final thesis topic: Finite Automata and Latin arrays

Keywords:

**Čestné prehlásenie**

Podpisom čestne prehlasujem, že som Bakalársku prácu Konečné automaty a latin arrays vypracoval samostatne na základe poznatkov získaných počas štúdia a informácií, ktoré som získal z dostupnej literatúry uvedenej v práci. Uvedenú prácu som vypracoval pod vedením pána doc. prof. RNDr Otakara Grošeka, PhD.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

V Bratislave, dňa Podpis

# Poďakovanie

Ďakujem veducému Bakalárskej práce pánovi doc. prof. RNDr Otakarovi Grošekovi, PhD, vedúcemu Katedry informatiky a matematikyza za veľmi užitočnú metodickú pomoc a cenné rady pri spracovaní Bakalárskej práce.

Ďakujem ďalej svojim rodičom za naprosto nenahraditeľnú podporu v priebehu tvorby tejto práce.

### Obsah

Obsah

[Poďakovanie 7](#_Toc344508129)

[Obsah 8](#_Toc344508130)

[Úvod 10](#_Toc344508131)

[1. Konečné automaty(KA) 11](#_Toc344508132)

[1.1 Definícia 11](#_Toc344508133)

[1.2 Základné pojmy konečných automatov 11](#_Toc344508134)

[1.3 Deterministický konečný automat 11](#_Toc344508135)

[1.4 Nedeterministický konečný automat 12](#_Toc344508136)

[1.5 Rozdiely medzi nedeterministickým a deterministickým konečným automatom 12](#_Toc344508137)

[1.6 Koncepcia invertibility konečných automatov 12](#_Toc344508138)

[1.6.1 Teorémy 13](#_Toc344508139)

[1.6.2 Dôsledky 13](#_Toc344508140)

[1.7 Šírenie chýb a dopredná inverzia 14](#_Toc344508141)

[1.7.1 Teorémy 14](#_Toc344508142)

[1.7.2 Dôsledky 14](#_Toc344508143)

[2. Symetrický kryptosystém a latin arrays 15](#_Toc344508144)

[2.1 Kanonická forma pre konečné automaty symetrických kryptografických systémov 15](#_Toc344508145)

[2.2 Grafické znázornenie kódera a dekódera 17](#_Toc344508146)

[2.3 Latin array 18](#_Toc344508147)

[2.3.1. Definície 18](#_Toc344508148)

[2.3.2 (n, k, r) Latin arrays 19](#_Toc344508149)

[2.3.3 Invariant 20](#_Toc344508150)

[2.3.4 Autotopism Group 22](#_Toc344508151)

[2.3.5. n = 2, 3 22](#_Toc344508152)

[2.3.6 n= 4, k ≤ 4 23](#_Toc344508153)

[2.3.7 (4, 3)-Latin Arrays 24](#_Toc344508154)

[2.3.8 (4, 4)-Latin Arrays 25](#_Toc344508155)

[2.3.9. Aplikácia (4, 4)-Latin array v kryptografii 28](#_Toc344508156)

[2.4 Lineárne závislé latin arrays 29](#_Toc344508157)

[2.4.1. Latin arrays a inverzné funkcie 29](#_Toc344508158)

[2.4.2 Generovanie lineárne nezávislých permutácií 29](#_Toc344508159)

[3. Konečné automaty a asymetrický kryptosystém 35](#_Toc344508160)

[3.1 Teorémy 35](#_Toc344508161)

[3.2 Šifrovacie algoritmy 36](#_Toc344508162)

[3.3 Šifrovanie 37](#_Toc344508163)

[3.4 Dešifrovanie 37](#_Toc344508164)

[3.5 Slabé kľúče 38](#_Toc344508165)

[3.5.1 Test lineárnej transformácie Ra, Rb 38](#_Toc344508166)

[3.5.2 Útok redukovaním stupňovitej matice 38](#_Toc344508167)

[3.5.3 Útok kanonicky diagonálnou maticou polynómu 39](#_Toc344508168)

[3.6 Bezpečnosť 39](#_Toc344508169)

[3.6.1 Inversion by a General Method 40](#_Toc344508170)

[3.6.2 Exhausting Search and Stochastic Search 40](#_Toc344508171)

# Úvod

NA STRANU

# 1. Konečné automaty(KA)

## 1.1 Definícia

Konečné automaty sú matematické abstrakcie diskrétnych a digitálnych systémov s konečnou "pamäťou". Zo štrukturálneho hľadiska má tento systém vstup, výstup a vnútorné stavy. Z funkčného hľadiska konečné automaty transformujú vstupné sekvencie na výstupné sekvencie rovnakej dĺžky. Z časového hľadiska sa jedná o diskrétny systém.

V každom okamihu môže byť vstup zhotovený iba z obmedzeného množstva hodnôt(vstup, výstup, vnútorný stav). V konkrétnom čase je vstup a vnútorný stav jednoznačne určený.

Vstupnými postupnosťami môžeme kedykoľvek získať výstupné postupnosti. Systém je invertibilný. Možno ho preto použiť na dekódovanie šifier a na opravu chýb. Konečné automaty sú jedným zo základných prostriedkov na popis regulárnych jazykov. Existujú dva základné modely automatov- deterministický konečný automat a nedeterministický konečný automat. Ku každému nedeterministickému konečnému automatu A existuje deterministický konečný automat B taký, že L(B) = L(A), kde L(A)(resp. L(B) ) predstavuje jazyk akceptovaný konečným automatom. Deterministický automat je špeciálny prípad nedeterministického automatu.

## 1.2 Základné pojmy konečných automatov

Pre každú množinu A, spojenie prvkov a0a1 . . .al-1 z A nazývame slovo (konečnú postupnosť) nad A, l označuje dĺžku slova. V prípade, že dĺžka je nulová, a0a1 . . .al-1 je označovaná ako prázdna postupnosť(neobsahuje žiadne prvky). Takáto prázdna postupnosť sa nazýva prázdne slovo, označované ԑ. Dĺžka slova α sa označuje |α|. Množina všetkých slov nad A, vrátane prázdneho slova, sa označuje A\*. Ak a0,a1, ..., an, ... sú prvky množiny A, tak spojenie nekonečne veľa prvkov a0a1..an... sa nazýva nekonečne dlhé slovo alebo ω (slovo alebo nekonečná postupnosť) nad A. Na označenie množiny všetkých nekonečne dlhých slov nad A sa používa Aω. An označuje všetky slová nad A dĺžky n, pričom n musí byť nezáporné. Nech α= a0a1...am-1 a β=b0b1...bn-1 sú dve slová v A\*. Spojenie slov α a β je a0a1...am-1b0b1...bn -1, slovo z A\* dĺžky m+n. Takéto slovo značíme ako α.β (αβ), pričom platí α. ԑ = ԑ. α = α. Podobne platí, že ak α = a0a1...am -1 je z A\* a β = b0b1...bn -1... z Aω, potom spojenie α. a β je prvok a0a1...am-1b0b1...bn-1... z Aω, ktorý tiež označujeme α.β. Samozrejme, ԑ. β = β. β sa nazýva preffix α, ak existuje také ϒ, že α= β. ϒ, β je suffix α, ak existuje také ϒ, že α= ϒ. β. Pre každé U, V A\*, spojenie U a V je množina { αβ | α β }, táto množina sa označuje UV.

## 1.3 Deterministický konečný automat

Deterministický konečný automat je pätica(Σ, K , q0, δ, F), kde Σ je vstupná abeceda(neprázdna konečná množina symbolov), K je konečná množina stavov, q0 je počiatočný stav, pričom platí q0  K, δ je prechodová funkcia δ: K Σ K, čiže funkcia, ktorá na základe stavu a symbolu zo vstupnej abecedy vráti nový stav. F je množina akceptačných stavov, ktorá je ľubovoľná (môže byť aj prázdna) podmnožina K. Hovoríme, že deterministický konečný automat akceptuje slovo , ak výpočet na tomto slove skončí v niektorom z akceptačných stavov.

## 1.4 Nedeterministický konečný automat

Nedeterministický konečný automat je pätica(Σ, K, q0, δ, F), kde Σ je vstupná abeceda(neprázdna konečná množina symbolov). K je konečná množina stavov, q0 je počiatočný stav, pričom platí q0  K. δ je prechodová funkcia δ: K ( ) 2K, čiže funkcia, ktorá na základe stavu a symbolu zo vstupnej abecedy vytvorí množinu nových stavov. F je množina akceptačných stavov, je to ľubovoľná (môže byť aj prázdna) podmnožina K. Hovoríme, že nedeterministický konečný automat akceptuje slovo , ak výpočet na tomto slove skončí v niektorom z akceptačných stavov.

## 1.5 Rozdiely medzi nedeterministickým a deterministickým konečným automatom

Za hlavné rozdiely medzi automatmi možno pokladať nasledovné:

* Nedeterministické KA povoľujú prechody na \varepsilon
* Pri nedeterministickom automate nie je nový stav pre každý prechod určený jednoznačne. Prechodová funkcia vracia celú množinu stavov (KA môže postupovať do ľubovoľného z nich). Táto množina môže byť dokonca i prázdna.

Zdroj: http://sk.wikipedia.org/wiki/Kone%C4%8Dn%C3%BD\_automat

## 1.6 Koncepcia invertibility konečných automatov

Automat *M = <X,Y,S, δ, λ >* je invertibilný práve vtedy, ak pre každý stav *s*, *s´* v *S*a ľubovoľnú postupnosť *α*, *α´* v *Xω* platí vzťah *λ(s, α) = λ(s, α´),*  *α = α´*. Teda *M* je invertibilný práve vtedy, ak pre každý stav *s* v *S*a ľubovoľnú postupnosť *α* v *Xω,* môže byť automat M jednoznačne určený výstupnou funkciou *λ(s, α*.*)*. Ak *M* je invertibilný, tak *M´´= <X,Y,S´´, δ´´, λ´´>* je invertibilný vtedy, ak *M´´≺ M* alebo *M´´⩽ M*.

Konečný automat *M = <X,Y,S, δ, λ >* je invertibilný s oneskorením *τ*, kde *τ* je nezáporné celé číslo, práve vtedy, ak pre každý stav *s* v *S* a ľubovoľné *xi* v *X*, *i = 0, 1,... , τ* môže byť jednoznačne určený výstupnou funkciou *λ(s, x0x1 ...xτ)*. To znamená, že pre každé *s*, *s´* v *S* a ľubovoľné *xi*, *xi´* v *X*, *i = 0, 1, ..., τ*, platí *λ(s, x0x1....xτ) = λ(s, x0´x1´... xτ, )* aplatí *x0 = x0´*.

Platí teda, že ak je *M* invertibilný s oneskorením *τ*, potom *M´´= <X,Y,S´´, δ´´, λ´´>* je invertibilný s oneskorením *τ* v prípade, ak *M´´≺ M* alebo *M´´⩽ M*. Ak *M* je invertibilný s oneskorením *τ*, potom *M* je invertibilný. Táto veta platí aj obrátene.

### 1.6.1 Teorémy

1, *M* je invertibilný práve vtedy, ak *GM* nemá kruh. Navyše, ak *GM* nemá kruh, vzdialenosť od vrcholugrafu *GM* je *ρ – 1,* potom *M* je invertibilný s oneskorením *ρ +1 a* nie je invertibilný s oneskorením *τ* pre každé *τ ⩽ ρ*.

2, Ak *M* je invertibilný s oneskorením *τ* , potom existuje konečný automat *M´* so vstupnou pamäťou *τ-*rádu, taký, že *M´* je inverzný s oneskorením *τ*.

3, *M* je slabo invertibilný práve vtedy, ak *G´M* nemá kruh. Navyše, ak *GM* nemá kruh, vzdialenosť od vrcholu grafu *G´M* je *ρ – 1,* potom *M* je slabo invertibilný s oneskorením *ρ +1* anie je slabo invertibilný s oneskorením *τ* pre každé *τ ⩽ ρ*.

4, Ak *M* je slabo invertibilný s oneskorením *τ*, potom existuje konečný automat *M´* taký, že *M´* je slabo inverzný s oneskorením *τ*  automatu *M*.

5, Nech *M =* *<X,Y,S, δ, λ>* je konečný automat. Ak *M* je slabo invertibilný s oneskorením *τ* , potom platí rovnosť |*X*| ⩽ |*Y*|.

6, Nech *M =* *<X,Y,S, δ, λ>* je konečný automat. Ak |*X*| = |*Y*| a *M* je slabo invertibilný s oneskorením *τ* , potom vzťah *λ(S, Xr) (* *{ λ(S, α)*|*s S, α Xr }) = Yr* platí pre všetky nezáporné celé čísla *r*.

### 1.6.2 Dôsledky

1, Ak je *M* invertibilný, potom existuje *τ ⩽ n (n – 1)/2* také, že *M* je invertibilný s oneskorením *τ*, kde *n* je číslo prvku v stavovej abecede automatu *M*.

2, *M* je invertibilný s oneskorením *τ* práve vetdy, ak existuje konečný automat *M´* taký, že *M´* je inverzný s oneskorením *τ* automatu *M*.

3, Ak je *M* slabo invertibilný, potom existuje *τ ⩽ n (n – 1)/2* také, že *M* je slabo invertibilný s oneskorením *τ*, kde *n* je číslo prvku v abecede stavov automatu *M*.

4, *M* je slabo invertibilný s oneskorením *τ* práve vtedy, ak existuje konečný automat *M´* taký, že *M´* je slabo inverzný s oneskorením *τ*  automatu.

*M´* nazývame inverzný (inverzia) s oneskorením *τ* automatu *M*, ak pre každé *s* v *S* a ľubovoľné *s´* v *S´*, je *(s´, s)* zodpovedajúci pár s oneskorením *τ*. *M* voláme pôvodná inverzia s oneskorením *τ* automatu *M´* , ak *M´* je inverzný s oneskorením *τ* automatu *M*. *M´* voláme inverzný s oneskorením *τ,* ak *M´* je inverzný z oneskorením *τ* nejakého konečného automatu. *M´* nazývame inverzný, ak *M´* je inverzný s oneskorením *τ* pre nejaké *τ*.

## 1.7 Šírenie chýb a dopredná inverzia

Nech *M´=<X,Y,S´, δ´, λ´>* je inverzný konečný automat s oneskorením *τ* automatu *M =<X,Y,S, δ, λ>.* Pre každú konečnú vstupnú postupnosť *α´ Xω* a každý stav *s S*, nech *β = λ(s, α)*. Potom pre každý stav *s´ S* automatu *M* existuje *α0*  *X\** dĺžky *τ* , také, že *λ´(s´, β) = α0α*. Predpokladajme, že *β = ββ2 a β´ = ββ*, pričom |*β1*| = |*β*|. Nech *α = αα*, pričom |*α*| = |*β*|. Potom *λ´(s´, β´) = λ´(s´, β) λ´(δ´(s´, β), β)= λ´(s´, β) λ´(δ´(s´, β), λ(δ´(s, α), α))= λ´(s´, β)α α*,pre postupnosti *α* dlžky *τ* v *X\**. Pretože |*αα*| = |*λ´(s´, β)α* |, *i* –tý znak výstupnej funkcie *λ´(s´, β)* sa rovná *i* –tému znaku výstupnej funkcie *λ´(s´, β´)* ak *i >*|*β*| + *τ.* To znamená, že šírenie dešifrovacích chýb inverzie *M´* na *M* je najviac *τ* znakov.

Nech *M´ =* *<X,Y,S´, δ´, λ´>* je slabo inverzný konečný automat s oneskorením *τ* automatu *M =* *<X,Y,S, δ, λ>*.Pre každé *s* v *S* a akékoľvek *s´* v *S´*, ak ak *s´* *τ*-zodpovedá stavu *s* a ak pre každú postupnosť *α* v *X\** a každú postupnosť *β* v *Y\**, kde |*α*|=|*β*| a ľubovoľné *k*, *0⩽ k ⩽* |*β*| *–c*, *λ (s, α) =k β* znamená *λ´(s´, λ(s, α)) =k* + c *λ´(s´, β)*, pár *(s, s´)* nazývame *(τ, c) –*zodpovedajúci pár. Pre každý stav *s* v *S* existuje *s´* v *S´* také, že *(s, s´)* je *(τ, c) –*zodpovedajúci pár. Hovoríme, že šírenie slabo dešifrovacích chýb automatu M´na M je ohraničené dľžkou šírenia chýb ⩽*c*. Minimálne nezáporné celé číslo *c*, ktoré spĺňa nasledujúcu podmienku sa nazýva dľžka šírenia chýb.

### 1.7.1 Teorémy

1, Nech *M´ =* *<X,Y,S´, δ´, λ´>* je slabo inverzný konečný automat s oneskorením *τ* automatu *M=* *<X,Y,S, δ, λ>*. Predpokladajme, že šírenie slabo dekódovacích chýb automatu *M´* na *M* je ohraničené dĺžkou šírenia chýb ⩽*c*, kde *c* ⩾ *τ*. Potom môžeme skonštruovať konečný automat SIM *(M´´, f)* so polo vstupnou pamäťou *c* –rádu taký, že SIM *(M´´, f)* je slabo inverzný automat s oneskorením *τ* automatu *M*.

2, Konečný automat *M* je dopredne invertibilný s oneskorením *τ*, práve vtedy, ak existuje konečný automat *M´* taký, že *M´* je slabo inverzný s oneskorením *τ* automatu *M*. Šírenie slabo dekódovacích chýb automatu *M´* na *M* je ohraničené.

### 1.7.2 Dôsledky

1,Ak *M* je lineárny konečný automat nad GF(*q*) a *S* *= {δ(0, α)*| *α X\*}*, potom SIM *(M´´, f)* môže byť ekvivalentný s lineárnym konečným automatom so vstupnou pamäťou *c*-rádu.

2, Nech *M´* je slabo inverzný automat s oneskorením *τ* konečného lineárneho automatu *M* nad GF(*q*). Predpokladajme, že šírenie slabo dekódovacích chýb automatu *M´* na *M* je ohraničené dĺžkou šírenia chýb ⩽*c*, kde *c* ⩾ *τ*. Potom môžeme skonštruovať lineárny konečný automat so polo vstupnou pamäťou c-rádu SIM *(M´´, f)* nadGF(*q*), taký že SIM *(M´´, f)* je slabo inverzný konečný automat s oneskorením *τ* automatu *M.*

*3,* Nech *M* je lineárny konečný automat. Potom *M* je dopredne invertibilný s oneskorením *τ* práve vtedy, ak existuje lineárny konečný automat s polo vstupnou pamäťou konečného rádu, ktorý je slabo inverzný s oneskorením *τ* automatu *M*.

# 2. Symetrický kryptosystém a latin arrays

## 2.1 Kanonická forma pre konečné automaty symetrických kryptografických systémov

Z matematického hľadiska predstavuje kryptografický systém (šifrovanie) rodinu transformácií {fk, k ∈ K}, ktoré závisia od parametra k tj. kľúča, kde K reprezentuje množinu kľúčov. Šifrovacia transformácia fk je injektívnym zobrazením z množiny P (otvorený text priestoru) do množiny C (priestor zašifrovaného textu). Ak chce odosielateľ poslať správu otvoreného textu α pomocou nezabezpečeného kanála, ktorý môže byť odpočúvaný nepriateľom, odosielateľ najskôr zakóduje správu α aplikovaním transformácie fk. Následne môže po kanáli poslať výslednú transformáciu fk(α). Táto transformácia reprezentuje zašifrovaný text. Legitímny príjemca po prijatí správy dešifruje zašifrovaný text fk(α) aplikovaním inverznej transformácie f k transformácií fk.

Ak odosielateľ a príjemca zdieľajú spoločný kľúč, je takýto kryptosystém nazývaný jedno kľúčový kryptografický systém.

Príklad kryptografického systému je prúdová šifra, ktorej kľúčový reťazec

je pseudonáhodná postupnosť generovaná binárne posuvným registrom vzhľadom na n. Kľúčový priestor je vektorový priestor dimenzie n nad GF(2), priestor otvoreného textu a priestor zašifrovaného textu pozostáva zo všetkých slov nad GF(2) danej dĺžky.

Kryptografická transformácia fk je definovaná rovnicou fk(x0x1...xl-1) = y0y1...yl-1, kde yi = si ⊕ xi, i = 0, 1,..., l- 1, a kľúčový reťazec s0s1...sl-1 je prvých l číslic výstupu posuvného registra pre počiatočný stav k.

Hoci sú v prúdových šifrách posuvné registre dôležité sekvenčné generátory, v podstate sú len špeciálny druh autonómneho konečného automatu. Konečné automaty sú považované za matematický model kryptografického systému z implementačného hľadiska, kde priestor otvoreného textu a priestor zašifrovaného textu pozostáva zo všetkých slov nad niektoré konečné množiny a šifrovacie transformácie fk(α) rovnajúce sa λ (k, α), kde λ je výstupná funkcia niektorých slabo invertibilných konečných automatoch. Kľúčový priestor je množina slabo invertibilných konečných automatov a ich počiatočné stavy.

Predpokladajme, že konečný automat M = <X, Y, S, δ, λ> je vybraný ako kodér na implementáciu šifrovania. Potom M musí spĺňať určité podmienky invertibilnosti. Ak M je invertibilný s oneskorením τ, môžno zvoliť jeho inverzný konečný automat s oneskorením τ, povedzme τ- order input memory konečný automat M = <X, Y, S‘, δ‘, λ‘> , ktorý zodpovedá dekodéru na realizáciu dešifrovania.

Na zašifrovanie otvoreného textu x0...xl-1 v X\* najskôr rozšírime náhodne τ písmená xl,..., xl+τ−1 v X na koniec postupnosti, zvolíme náhodne stav s v M. Následne vypočítame y0...yl+τ−1 = λ(s, x0...xl+τ−1), kde y0...yl+τ−1 je zašifrovaný text z textu x0...xl−1. Pri dešifrovaní porovnávame pre každý stav s’ v M´ rovnosť x‘0...x‘l+τ−1 = λ‘(s’, y0...yl+τ−1), následne porovnávame, či otvorený text x0...xl−1 je rovný x‘τ...x‘l+τ−1. Kľúč je v tomto prípade štruktúra z M. V prípade, ak M je slabo invertibilné s oneskorením τ, môžeme zvoliť jeho slabý inverzný konečný automat M‘ = <Y,X,S‘, δ‘, λ‘> s oneskorením τ ako vhodný dekóder. Na zašifrovanie otvoreného textu x0...xl−1 z X\*, najprv rozšírime X o náhodné τ písmená xl,..., xl+τ−1 , ktoré pridáme na koniec. Potom vypočítame y0...yl+τ−1 = λ(s, x0...xl+τ−1), kde y0...yl+τ−1 je zašifrovaný text z x0...xl−1. Na rozšifrovanie vypočítame x0...x‘l+τ−1 = λ‘(s‘, y0... yl+τ−1), kde s‘ sú stavy z M‘. Následne sa otvorený text x0...xl−1 rovná x‘τ...x’l+τ−1. Kľúč je stav s z M, v prípade ak štruktúra M je pevná.

V invertibilnom prípade, kde M‘ je vstupno pamäťový konečný automat, chyba slova v šifre spôsobí najviac τ + 1 chýb písmen v dešifrovaní. V slabo invertibilnom prípade niekedy chyba v písmene v šifre môže spôsobiť nekonečne veľa chýb v dešifrovaní.

Napríklad, nech X a Y є Z2, *M* je konečný automat so vstupnou pamäťou 1- rádu Mf, kfe f je zobrazenie z X2 do Y, *f(x0, x–1)* = x0⊕*x–1*. Nech *M´* je konečný automat s pamäťou *(0, 1)*-rádu *Mg*, kde *g* je zobrazenie z *X* x *Y* do *X a g(x – 1,* *y0) = x–1*⊕*x0*.

Preto na zaručenie ohraničeného šírenia chýb kódery a dekodéry musia byť spätnoväznovo invertibilné.

Ak je M = <X, Y, S, δ, λ> kóder, potom platí |Y | |X|. Na reprezentáciu všetkých zašifrovaných textov zo všetkých otvorených textov dĺžky l (l log2|X| bitov) je potrebných (l+τ) log2|Y | bitov. Potom l\*log2|X| (l+ τ) log2|Y|. Preto neexistuje rozšírenie otvoreného textu práve vtedy ak l\*log2|X| = (l+ τ) log2|Y|, práve vtedy ak |Y|= |X| a dĺžka omeškanie τ = 0.

Pre jedno kľúčové kryptografické systémy, ktoré sú reprezentované konečným automatom bez rozšírenia otvoreného textu a s ohraničeným šírením kódovacích chýb, dekódery môžu byť slabo inverzné semi-input memory konečný automat s oneskorením 0, v ktorom má vstupná a výstupné abeceda konečného automatu rovnakú veľkosť.

Dekóder M‘ = <Y,X,S‘, δ‘, λ‘> je c-order semi-input-memory konečný automat SIM(Ma, f), kde X=Y, Ma= <Ya, Sa, δa, λa> je autonómny konečný automat f s mapovaním z Yc+1×λa(Sa) do X s |f(Y,yc−1,...,y0,λa(sa))| = |X| pre každé sa ∈ Sa a každé y0,...,yc−1 ∈ Y . Pre každé ya ∈ λa(Sa) a všetky y0,...,yc−1 ∈ Y, nech fyc−1,...,y0,ya je jedno hodnotové zobrazenie z Y do X definované podľa fyc−1,...,y0,ya(yc)=f(yc,...,y0, ya), yc ∈ Y. Je evidentné, že fyc−1,...,y0,ya je permutácia na Y (alebo X). Potom existuje single valued mapping h z Yc×λa(Sa) do W, čiže f(yc, yc−1,..., y0, ya)=g−1h(yc−1,...,y0,ya)(yc), ya ∈ λa(Sa), y0,..., yc ∈ Y pre konečnú množinu W, kde gw-1 je bijekcia z Y do X. Pre každé w z W. Pre každý vstupný stav s‘0=<y−1,..., y−c, sa0> a každú vstupnú postupnosť (zašifrovaný text) y0...yl−1 z M’, výstupná postupnosť (otvorený text) x0...xl−1 z M‘ sa môže vypočítať pomocou sa,i+1=δa(sai), ti=λa(sai), wi=h(yi−1,..., yi−c, ti), xi= gwi-1(yi), i = 0, 1,..., l−1.

Medzi iným môže byť ako zodpovedajúci kóder zvolený konečný automat M = <X, Y, Yc × Sa, δ, λ>, ktorý možno bližšie definovať nasledovne:

δ(<y−1,..., y-c, sa>, x0) = <y0, y−1,..., y-c+1, δa(sa)>, λ(<y−1,..., y−c, sa>, x0)=y0, w0=h(y−1,..., y−c, λa(sa)), y0 = gw0 (x0), <y−1,..., y−c, sa> ∈ Yc× Sa, x0 ∈ X.

Pre každý vstupný stav s0=<y−1,..., y−c, sa0> a pre každú vstupnú postupnosť(otvorený text) x0...xl−1 z M a výstupný text(zašifrovaný text) y0...yl−1 z M platí: sa,i+1 = δa(sai), ti = λa(sai), wi = h(yi−1,..., yi−c, ti), yi = gwi(xi), i = 0, 1,..., l−1.

Špeciálny prípad nastáva ak c = 0. Kanonická forma pre jedno kľúčový kryptografický systém reprezentovaný konečným automatom bez rozšírenia otvoreného textu a bez šírenia chýb dekódovania je nasledovná.

Dekodér M‘ =< Y, X, Sa, δ‘, λ ‘> je 0-order semi-input-memory konečný

automat SIM (Ma, g‘), kde X = Y,

δ‘(sa, y) = δa(sa), λ‘(sa, y) = gw−1(y), w = λa(sa), sa ∈ Sa, y ∈ Y,

Ma = <W, Sa, δa, λa> je autonómny konečný automat, gw−1 je bijekcia z Y do X pre všetky w z W, a gw−1(y)=g’(y, w). Pre každý vstupný stav sa0 a každú výstupnú sekciu(zašifrovaný text) y0...yl−1 z M’, výstupná postupnosť(otvorený text) x0...xl−1 z M‘ môže byť počítané podľa:

sa,i+1 = δa(sai), wi = λa(sai), xi = gwi−1(yi), i = 0, 1,..., l−1.

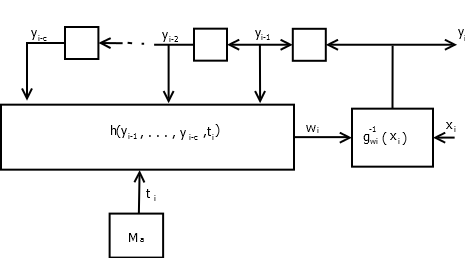
Odpovedajúci kóder môže byť konečný automat M =<X,Y, Sa, δ, λ> Kde X = Y,

δ(sa, x) = δa(sa), λ(sa, x) = gw(x), w = λa(sa), sa ∈ Sa, x ∈ X.

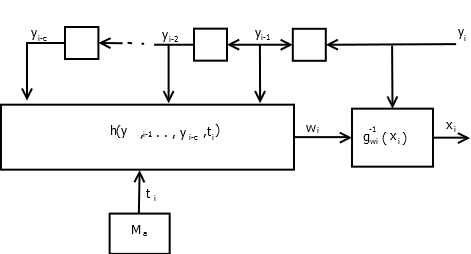
To znamená, že M =< X, Y, So, δ, λ> je tiež -order semi-input-memory konečný automat SIM (Ma, g), kde g (x, w) = gw(x). Pre každý pôvodný stavu sa0 a akúkoľvek vstupnú postupnosť(otvorený text) x0...xl-1 z M, výstupnú postupnosť (zašifrovaný text) Y0. . . yl-1 M môže byť počítaný podľa:

sa,i+1 = δa(sai), wi = λa(sai), yi = gwi (xi), i = 0, 1, . . . , l − 1.

## 2.2 Grafické znázornenie kódera a dekódera



Obrázok 1 Kóder

****

Obrázok 2 Dekóder

## 2.3 Latin array

## 2.3.1. Definície

Problém vytvorenia jednokľúčových kryptografických systémov predstavovaných konečným automatom bez rozšírenia otvoreného textu a bez obmedzeného rozšírenia dekódovacích chýb môže byť redukovaný výberom vhodných parametrov. Tieto parametre sú veľkosť abecedy, dĺžka c v šifrovacom priebehu a vytvorenie troch komponentov kanonickej formy- autonómny konečný automat Ma, transformáciu h a permutačnú rodinu {gw, w in W}. Takýto systém je efektívny a zároveň bezpečný.

Predpokladajme, že rozloženie prvkov v odvodenej kľúčovej postupnosti w0 w1... je vo vyššie uvedenej kanonickej forme jednotná. Nech {gw, w vo W} je rodina permutácií na X. Vzhľadom na odolnosť voči plaintext útoku je nižšie uvedená vlastnosť 1 veľmi dôležitá.

**Vlastnosť 1**. Pre každé x, y v X, |{w|w v W, gw(x) = y}| = konštanta.

**Vlastnosť 2**. Pre každé w’ v W, |{w|w v W, gw = gw’ }| = konštanta.

Prvky X sú x1,..., xn a prvky W definujeme ako w1,..., wm. Nech A je n × m matica, z ktorých prvok v riadku i a stĺpci j je qwj(xi). Potom každý stĺpec v matici A je permutácia prvkov v X.

Nech A je n × nk matica nad N={1,. . . , N}. Ak sa každý prvok N vyskytuje práve raz v každom stĺpci A a k- krát v každom riadku, potom (n, k) voláme latin array.

Nech A je (n, k)- latin array. Nech sa každý stĺpec A nachádza presne r- krát v stĺpcoch A, potom A sa nazýva (n, k, r)- latin array.

(n, 1)-Latin arrays sú n-order Latin squares. Nech A a B sú n x m matice nad N. Keď maticu B získame z matice A po preskupovaním riadkov, po preskupovaním stĺpcov a premenovaním elementov, potom A a B sú izotopy a transformácia <α, β, γ> sa nazýva ISOTOPISM z A do B, kde α, β a γ je popreskupovanie riadkov, premenovanie a popreskupovanie stĺpcov. Izotopná relácia je reflexívna, symetrická a tranzitívna.

Ak A je (n, k) latin array a ak A a B sú izotopy, potom B je (n, k) latin array. Podobne, ak A je (n, k, r) latin array a ak A a B sú izotopy, potom je B (n, k, r) latin array. Pre (n, k) latin arrays alebo (n, k, r) latin arrays sa každá ekvivalencia triedy izotopov nazýva trieda izotopu.

### 2.3.2 (n, k, r) Latin arrays

Počet všetkých (n, k) latin arrays sa označujú U(n,k). U(n, k, r) reprezentuje počet všetkých (n, k, r) latin arrays.

I(n, k) je počet všetkých izotopných tried (n, k) latin arrays. I(n, k, r) používame na označenie počtu všetkých izotopných tried (n, k, r) latin arrays.

Nech Ai je matica n x mi, kde i=1,...,t. Matica [A1,…,At], označená n x (m1+...+ mt) sa nazýva zreťazenie A1,…,At. Zreťazenie z T identickej matice A sa nazýva t- fold zreťazenie z A, označená A(t).

Je zrejmé, že zreťazenie [A1,…,At] z (n, ki) latin array Ai, kde i =1,...,t je (n, k1 +...+kt)latin array

Ak A a B sú n × xm matice nad N. Ak B možno označiť ako A preskupením stĺpcov A, potom A a B sú stĺpcovo ekvivalentné.

Je zrejme, že relácia stĺpcovej ekvivalencie je reflexívna, symetrická a tranzitívna. Pre (n, k)latin arrays alebo (n, k, r)latin arrays ekvivalentné triedy relácie stĺpcovej ekvivalencie sa nazývajú stĺpcovo- ekvivalentné triedy. Pre každú maticu A používame b(A) na označenie matice získanej z A zmazaním rovnakých stĺpcov.

#### 2.3.2.1 Príklady

1,Nech je daná latin array() A= , nasledovné latin arrays sú navzájom izotopné.

(a) , ktorá vznikla preskupením stĺpcov

(b), ktorá vznikla preskupením riadkov

(c) , ktorá vznikla preskupením riadkov a stĺpcov

(d) , ktorá vznikla preskupením riadkov, stĺpcov a premenovaním premenných

2, Nech je daná latin array B= , potom b(B)=

#### 2.3.2.2 Lémy

1, Nech A je (n, k, 1)latin array, potom

1. A(r) je (n, k, r)latin array
2. r|k , b(A) je (n, k/r, 1)latin array a A a r-fold zreťazenie z b(A) sú izotopy a tiež sú stĺpcovo ekvivalentné.

2,Nech A a B sú dve (n, k, r)latin arrays

1. A a B sú izotopy práve vtedy ak b(A) a b(B) sú izotopy
2. A a B sú stĺpcovo-ekvivalentné práve vtedy, ak b(A) a b(B) sú stĺpcovo-ekvivalentné

3,I(n, k, r)=I(n, k/r, 1)

4,U(n, k, r)=U(n, k/r, 1)(nk)!/((nk/r)!(r!)nk/r)

5,Nech Ai je (n, k, 1)latin array a Ai´ je komplement Ai i=1, 2

1. A1 a A2 sú izotopy práve vtedy ak A1´ a A2´ sú izotopy
2. A1 a A2 sú stĺpcovo závislé práve vtedy ak A1´ a A2´ sú stĺpcovo závislé

6,Nech 1≤k<(n-1)!

1. I(n, k, 1)=I(n, (n-1)! –k, 1)
2. U(n, (n − 1)! − k, 1) = U(n, k, 1)(n! – nk)!/(nk)!
3. I(n, (n − 1)!, 1) = 1, U(n, (n − 1)!, 1) = (n!)!.

### 2.3.3 Invariant

Nech A je (n, k)-Latin array. Násobnosť každého stĺpca matice vyjadruje počet výskytov stĺpca v matici. Používame ci na označenie počtu odlišných stĺpcov s početnosťou i, pre i = 1,..., k. ck ,ck-1, ..., c2 sa nazýva charakteristická hodnota stĺpca matice A.

Pre ľubovoľné poradie (x1, ..., xk), kde xi je hodnota z ľubovoľného súboru s n-1 prvkami, nknk-1...n2 sa nazýva typ postupnosti, kde nj je počet vzdialeností xi s početnosťou j.

V prípade ak k = 2, možné typy sú 1 a 0, označujú sa ako twins a all diferent. Ak k = 3, možné typy sú 10, 01 a 00, ktoré sú označované trio, twins and all different.

V prípade ak k = 4, možných typy sú 100, 010, 002 a 001, ktoré sú označované quad, trio, double twins a twins,

Pre akékoľvek iné i, j používame A(i, j, a) na označenie j- tého riadku submatice zloženej zo stĺpcov A, ktorej prvky v riadku i sú a.

Nech ct je počet z a, 1 ≤ a ≤ n, rovnako aj typ z A(i, j, a) je t, potom T1(i, j)=Σt\*ct, kde t obsahuje všetky typy. Označme Σct =n, každé ch môže byť určené iným ct. Zafixovaním permutácie všetkých typov, označme ich tr,...,t1,T1(i, j) je reprezentované ako ctr, ctr-1,..., ct2. Napríklad, v prípade (4, 2)-latin array, permutujeme typy ako 1, 0, a predstavujú T1(i, j) z c1, v prípade, že (4, 3)-latin array permutujeme typy ako 10, 01, 00, ktoré reprezentujú T1 (i, j) z c10c01.

Pre rôzne i a j, kde i≠j a pre každé a, 1 ≤ a ≤ n, keď v type nk,nk-1,...n2 z A(i, j, a ) nenulového nk smaximálnym indexom h má hodnotu 1, potom definujeme π(a) ako prvok v A(i, j, a) s maximálnou početnosťou.

Ak mapovanie π je bijektívne, π sa nazýva odvodená permutácia z riadku i do riadku j, označená ako π(i, j). Odvodená permutácia môže byť vyjadrená ako výsledok disjunktných cyklov s dĺžkou > 1. Rozdelenia dĺžok týchto cyklov sa nazývajú

typy odvodených permutácií, označené T2(i, j). Rozdelenie týchto dĺžok cykov sa nazýva typ derivačnej permutácie. Ak odvodená permutácia neexistuje a v prípade, že maximálna násobnosť r prvkov vyskytujúcich sa v A(i, j, 1),..., (I, j, n), je väčšia ako k / 2, | I ∩ J | sa nazýva prienik čísel z riadku i do riadku j označený ako T3 (i, j), kde I = {a | a ∈ N, maximálna násobnosť prvkov v A(i, j, a) je r}, a J = {b | tu je a ∈ I, takže násobnosť b v A(i, j, a) je r}

Nech T (i, j) = (T1 (i, j), T2 (i, j), T3 (i, j)); T2 (i, j) a T3 (i, j), môžu byť nedefinované. Množina pozostávajúca z T (i, j), i, j = 1,. . . , n, i ≠ j (opakovanie prípustné) sa nazýva

riadková charakteristika množiny A. Nech GRA je (orientovaný) graf s množinou vrcholov N a množinou oblúkov (N x N) \ {(i, i), i ∈ N}, z ktorých každý oblúk, povedzme (i, j), je označený T (i, j); GRA sa nazýva riadkovo charakteristický graf A. GRA je symetrický, ak T (i, j) = T (j, i), i ≠ j. Graf je neorientovaný ak dva oblúky (i, j) a (j, i) sú spojené okrajom s koncovými bodmi i a j.

#### 2.3.3.1Teorémy

1,Nech A a B sú dve (n, k)latin array. Keď A a B sú izotopy, potom

1. stĺpcové charakteristické hodnoty A a B sú rovnaké,
2. riadkové charakteristické grafy A a B sú izomorfné,
3. riadkovo charakteristické množiny A a B sú rovnaké.

2, Pre každé dva (n, k)latin array platí, že ak stĺpcovo charakteristické hodnoty riadkovo charakteristických množín sú odlišné, potom latin arrays nie sú izotopné.

3. Pre n=4, 2 ≤ k ≤ 4 riadkovo charakteristické grafy (n, k)latin array sú symetrické.

#### 2.3.3.2 Príklad

Nech je daná (5,2)latin array A= , , nech A(3, 2, 3) je,

ct je číslo a-3 (kde 1≤ 3(a)≤ 5), t obsahuje typy poradia x1x2 a n2 je typ postupnosti. Keďže k=2, možné typy sú 1 alebo 0. V prípade latin array A(3, 2, 3) , teda (33) je to číslo 1.

T1(3,2)=1\*3=3

### 2.3.4 Autotopism Group

Pre každé n a k, skupina skladajúca sa zo všetkých izotopov z (n, k)-latinského poľa s rovnakými operáciami na množine sa nazýva izotopová skupina z (n, k)-latinského poľa, označená G. Je zrejmé, že G je reprezentovaná ako Sn × Sn × SNK, kde Sr predstavuje symetrickú skupinu stupňa r pre akékoľvek kladné číslo r. Označme prvok G z <> , kde (preskupenie riadkov), (premenovanie) a (preskupenie stĺpcov) sú permutácie z čísla n, n a nk sú kladné čísla.

Pre každé (n, k)-latin array A , izotopia z A do seba samej sa nazýva autotopisms na A do seba. Všetky autotopisms vytvárajú podskupinu G. GA označujeautotopism grupy A.

Vezmime ľubovoľnú (n, k)latin array A, nech

GA´´´= { <α, β, γ> GA | α=(1), β=(1) },

GA´´= { <α, β, γ> GA | α=(1) },

GA´= { <α, β, γ> GA | α(1)=1 },

Je zrejmé, že GA´´´ ≤GA´´≤ GA´≤ GA, kde symbol ≤ predstavuje podskupinu

SIr predstavuje symetrickú grupu na Ir. Z definícií je zrejmé, že GA´´´ je izomorfnés SI1SI2...SIr

#### 2.3.4.1. Teorémy a lémy

1,Nech <αi, βi, γi>  GA, i=1,2.

1. Ak α1 =α2 a β1 = β2 potom γ1 γ2 (mod SI1SI2...SIr)
2. Ak α1 =α2 a γ1 γ2 (mod SI1SI2...SIr), potom β1 = β2
3. Ak riadky A sú navzájom odlišné, β1 = β2 a γ1 γ2 (mod SI1SI2...SIr), potom α1 =α2

2, Nech G1 je podskupina G2 a GA. Pre každé g v G2, (pravý) koset gG1 ⊆ GA práve vtedy ak g GA.

1. Nech gi=<αi, βi, γi>, i=1,2. Ak g1, g2 GA´´, potom g1GA´´´= g2GA´´´ práve vtedy ak β1 = β2.
2. Ak g1, g2 G´A, potom g1GA´´= g2GA´´ práve vtedy ak α1 =α2
3. Ak g1, g2 GA, potom g1GA´= g2GA´ práve vtedy ak α1(1)=α2(1)

Predpokladajme, že (n, k)-latin arrays A a B sú kanonické a že A môže byť premenená na B prostredníctvom izotopu <(1), β, γ>. Potom pre každé i, 1≤ i≤ n, typy stĺpcov(riadkov) blokov Ai a Bβ(i) sú zhodné. Navyše, v prípade, že blok A, označme ho Ah ,čiže pár stĺpcového a riadkového typu Ah je odlišný od jedného z ostatných blokov a pre každé kladné celé číslo r je iba jeden stĺpec Ah s násobnosť r, potom β môže byť určený ako stĺpec.

### 2.3.5. n = 2, 3

Každá (n, k)latin array je aj (2, k, k)latin array. Preto, ak A je (2, k, r)latin array, potom r=k.

#### 2.3.5.1Teorémy

1, Pre každé kladné číslo k platí I(2, k)=1, U(2, k)=

2, Pre každé kladné číslo k platí:

1. I(3, k)=
2. U(3, k)=

3,I(3, k, 1) = U(3, k, 1) = 0 ak k > 2,

4,I(3, 2, 1) = 1, U(3, 2, 1) = 6!, I(3, 1, 1) = 1, U(3, 1, 1) = 3!2!.

### 2.3.6 n= 4, k ≤ 4

Z vyššie uvedených definícií vyplýva, že počet izotopov triedy latinských štvorcov of order 4 is 2 a počet latinských štvorcov of order 4 je (4!)2 .

Latin array (4,1) a latin array (4,1,1) sú zhodné s latinským štvorcom of order 4, pričom I(4,1) =I(4,1,1)=2 a U(4,1)=U(4,1,1)=(4!)2 .

Nech A142,..., A1142 sú nasledovné (4,2)-Latin arrays:

A142 A242 A342 A442 A542 A642

A742 A842 A942 A1042 A1142

#### 2.3.6.1Lémy

1,

1. A142, ..., A1142 nie sú navzájom izotopné
2. Každá (4, 2)latin array je izotopná s jednou latin array z A142, ..., A1142
3. Každá (4,2,1) latin array je izotopná s jednou latin array z A142, ..., A642
4. Každá (4,2,2) latin array je izotopná s jednou latin array z A142, ..., A942

2, I(4, 2) = 11, I(4, 2, 1) = 6, I(4, 2, 2) = 2.

3, U(4, 2) = 12640320; U(4, 2, 1) = 10281600; U(4, 2, 2) = 60480.

4, I(4, 1) = I(4, 1, 1) = 2; U(4, 1) = U(4, 1, 1) = (4!)2.

#### 2.3.6.2 Príklady

1, Nech je daná latin array B= . Táto latin array B je izotopná s A142, preskupením druhého a štvrtého riadku. <(24),(1),(1)>

2, Latin array C= je izotopná s latin array A542 preskupením prvého stĺpca za piaty a preskupením siedmeho stĺpca za ôsmy. <(1),(1),(15)(78)>

3, Latin array D= je izotopná s A1042 preskupením piateho a šiesteho stĺpca, preskupením prvého a posledného riadku, premenovaním prvkov 4 a 2. <(14),(24),(56)>

### 2.3.7 (4, 3)-Latin Arrays

Nech nasledujúce latin arrays majú označenie A143, A243,..., A4643

#### 2.3.7.1Lémy

1, A143, ..., A4643 nie sú navzájom izotopné

2, Každá (4, 3)-Latin array je izotopná s jednou z latin array A143, ..., A4643

3, Každá (4, 3, 1)-Latin array je izotopná s jednou latin array z A3643, ..., A4643

4, I(4, 3) = 46, I(4, 3, 1) = 11

5, U(4, 3, 1) = 306561024000, U(4, 3) = 805929062400

#### 2.3.7.2 Príklady

1, Latin array B= je izotopná s latin array A143 preskupením druhého a 7 stĺpca. <(1),(1),(27)>

2, Latin array C= je izotopná s latin array A143 preskupením tretieho a štvrtého riadku. <(34),(1),(1)>

3, Latin array D= je izotopná s latin array A243 premenovaním čísla 2 za 3. <(1),(23),(1)>

### 2.3.8 (4, 4)-Latin Arrays

Nech prvý riadok každej kanonickej (4, 4)Latin array je 1111222233334444. Keďže každý stĺpec kanonickej (4, 4)latin array je permutácia z 1, 2, 3 a 4, každá kanonická (4, 4)Latin array môže byť označená jej riadkami 2 a 3. Napríklad, ak riadky 2 a 3 kanonickej (4, 4) -Latinskej polia sú 2222111144443333 3333444411112222 a, respektíve, potom máme

A=

Nech (4,4​​)-latin array Ax44 je vo formáte: "x: druhý riadok Ax44, tretí riadok Ax44 ". Potom A144, A244, ..., A20144 je nasledovných 201 (4,4​​)-latin arrays.

1 : 2222111144443333, 3333444411112222

2 : 2222111144443333, 3333444422221111

3 : 2222111144443333, 3333444411122221

4 : 2222111144443333, 3333444422211112

5 : 2222111144443333, 3333444411222211

6 : 2222111144443333, 3334444311122221

7 : 2222111144443333, 3334444322211112

8 : 2222111144443333, 3334444311222211

9 : 2222111144443333, 3344443311222211

10 :2222333344441111, 3334444111122223

11 : 2222333344441111, 3344441111223322

12 : 3333444421111222, 4444333112222113

13 : 3333444422211112, 4444333111122223

14 : 3333444422111122, 4444333111222213

15 : 3333444422112211, 4444331111223322

16 : 3333444411122221, 4442333122241113

17 : 3333444411222211, 4442333122141123

18 : 3333444412222111, 4442333121141223

19 : 3333444421111222, 4442333112242113

20 : 3333444422111122, 4442333111242213

21 : 3333444422211112, 4442333111142223

22 : 3333444411121222, 4442111322243331

23 : 3333444421121221, 4442111312243332

24 : 3333444412221211, 4442331121143322

25 : 3333444412121212, 4442331121243321

26 : 3333444411121222, 4442331122243311

27 : 3333444412212211, 4442331121143322

28 : 3333444411212221, 4442331122143312

29 : 3333444411122122, 2244113344211233

30 : 3333444411221122, 2244113344112233

31 : 3333444412122121, 2244113344211233

32 : 3333444412221121, 2244113344112233

33 : 2222333444411113, 3333444121142221

34 : 2222113444413331, 3333444111242212

35 : 2222113444413331, 3333444112242112

36 : 2222331444413311, 3333444122141122

37 : 2222443311441133, 3333114444222211

38 : 2222333444411113, 3334444111222231

39 : 2222333444411113, 3334444111122232

40 : 2222333444411113, 3334444311122221

41 : 2222333444411113, 3334441112242231

42 : 2222333444411113, 3334441111242232

43 : 2222113444413331, 3334434122241112

44 : 2222113444413331, 3334434122141122

45 : 2222113444413331, 3334434121141222

46 : 2222113444413331, 3334434111142222

47 : 2222113444413331, 3334444322121112

48 : 2222113444413331, 3334444321121122

49 : 2222113444413331, 3334444311121222

50 : 2222113444413331, 3334441322141122

51 : 2222113444413331, 3334441321141222

52 : 2222113444413331, 3334441311142222

53 : 2222133444411133, 3334344122142211

54 : 2222133444411133, 3334344121142212

55 : 2222133444411133, 3334344111142222

56 : 2222133444411133, 3334414312142221

57 : 2222133444411133, 3334414322142211

58 : 2222133444411133, 3334414121142322

59 : 2222133444411133, 3334414121242321

60 : 2222133444411133, 3334414122242311

61 : 2222133444411133, 3334444311122212

62 : 2222133444411133, 3334444321122211

63 : 2222133444411133, 3334444111122322

64 : 2222334444111122, 3334413121442221

65 : 2222334444111133, 3334413122442211

66 : 2222334444111133, 3334443111422221

67 : 2222334444111133, 3334443121422211

68 : 2222334444111133, 3334441111423222

69 : 2222333444411113, 3344411122143322

70 : 2222333444411113, 3344411122243321

71 : 2222333444411113, 3344411321142322

72 : 2222113444413331, 3344334121141222

73 : 2222113444413331, 3344334121241212

74 : 2222113444413331, 3344334122241112

75 : 2222113444413331, 3344431121242213

76 : 2222113444413331, 3344434111221223

77 : 2222133444411133, 3344344321122211

78 : 2222133444411133, 3344314321142221

79 : 2222133444411133, 3344314322142211

80 : 2222133444411133, 3344314121142322

81 : 2222133444411133, 3344314122142312

82 : 2222133444411133, 3344314122242311

83 : 2222334444111133, 3344443311222211

84 : 2222334444111133, 3344413311422221

85 : 2222334444111133, 3344413111422322

86 : 2222334444111133, 3344413121422312

87 : 2222334444111133, 3344441111223322

88 : 2222334444111133, 3344441112223312

89 : 2223111444413332, 3334444311122221

90 : 2223111444413332, 3334444311222211

91 : 2223111444413332, 3334444312222111

92 : 2223111444413332, 4332443321242111

93 : 2223111444413332, 4332443321142121

94 : 2223111444423331, 4332443121242113

95 : 2223111444413332, 4332443121242113

96 : 2223111444413332, 4332443121142123

97 : 2223111444423331, 4332443121142123

98 : 2223111444413332, 3344443321121221

99 : 2223333444411112, 3334441122143221

100 : 2224333144421113, 3332441411243221

101 : 2223333444411112, 3444411112222333

102 : 2223333444411112, 3444411321123223

103 : 2223333444411112, 3444411312222331

104 : 2224333144421113, 4333411412143222

105 : 2224333144421113, 4333411412243221

106 : 2224333144421113, 4433114421213232

107 : 2224333144421113, 4433414321212231

108 : 2224333144411123, 3332144421142312

109 : 2224333144411123, 3332144422142311

110 : 2224333144411123, 3342444311222311

111 : 2224333144411123, 3343444311222211

112 : 2224333144411123, 3442411412123332

113 : 2224333144411123, 3442411412223331

114 : 2224333144411123, 3442414312123231

115 : 2224333144411123, 3442411312143232

116 : 2224333144411123, 3442411312243231

117 : 2224333144411123, 3442414312123312

118 : 2224333144411123, 3442414312223311

119 : 2224333144411123, 3443411312242231

120 : 2224333144411123, 3443411312242312

121 : 2224333144411123, 3432414312142231

122 : 2224333144411123, 3432411412142332

123 : 2224333444111123, 3332441122443211

124 : 2224333444111123, 3332444121423211

125 : 2224333444111123, 4333411321442212

126 : 2224333444111123, 4333411322442211

127 : 2224333444111123, 4333411122442312

128 : 2224333444111123, 4333411122442231

129 : 2224333444111123, 4333441321422211

130 : 2224333444111123, 4333441121422312

131 : 2224333444111123, 4333441122422311

132 : 2224333444111123, 4333441112422231

133 : 2224333444111123, 4433144312222311

134 : 2224333444111123, 3342144311242231

135 : 2224333444111123, 3342144312243211

136 : 2224333444111123, 3342114321443212

137 : 2224333444111123, 3342441122243311

138 : 2223311444411332, 3334444311222121

139 : 2223311444411332, 3334444111223221

140 : 2223311444411332, 4332434121142123

141 : 2223311444411332, 4332434321142121

142 : 2223311444411332, 4332434121143221

143 : 2223311444411332, 4332144321143221

144 : 2223311444411332, 4332434122142113

145 : 2223311444411332, 4332434322142111

146 : 2223311444411332, 4332434122143211

147 : 2223311444411332, 4332144122143213

148 : 2223311444411332, 4332444311223211

149 : 2223311444411332, 3344144311223221

150 : 2223311444411332, 3344434311222121

151 : 2223331444411132, 3334144122143221

152 : 2223331444411132, 4332144321142213

153 : 2223331444411132, 4332144121243321

154 : 2223331444411132, 4332144321143221

155 : 2223331444411132, 4332144321243211

156 : 2223331444411132, 4332144121242313

157 : 2223331444411132, 4332443121242311

158 : 2223331444411132, 4332443121142213

159 : 2223331444411132, 4332444121123321

160 : 2223331444411132, 3344443311222211

161 : 2223331444411132, 3344414311222321

162 : 2223331444411132, 3344143122142321

163 : 2223113444421133, 4332434112142321

164 : 2223113444421133, 4332441112143322

165 : 2223113444421133, 4332444112113322

166 : 2223113444421133, 3344434321112212

167 : 2223113444421133, 3344441321112322

168 : 2223113444421133, 3344341311242212

169 : 2224113344421133, 4333434421212211

170 : 2224113344421133, 4333434121242211

171 : 2224113344421133, 3342441111243322

172 : 2224113344421133, 3342434111243212

173 : 2223314444123311, 4332141321441232

174 : 2223314444123311, 4332441321411232

175 : 2223314444123311, 4332441311241223

176 : 2223314444123311, 4332431121441232

177 : 2223314444123311, 3344431112241232

178 : 2223314444123311, 3344143311241222

179 : 2223314444123311, 3344141311242232

180 : 2223314444123311, 3344141312242132

181 : 2223314444111332, 3334441312242211

182 : 2223314444111332, 4332431122442113

183 : 2223314444111332, 3344443311222211

184 : 2223314444111332, 3344441311223221

185 : 2223314444111332, 3344441312223211

186 : 2223314444111332, 3344433111422221

187 : 2223314444111332, 3344433121422211

188 : 2223314444111332, 3344141321423221

189 : 2223334444113211, 3344141321422132

190 : 2233441111442233, 3344334422111122

191 : 2233314444211123, 3344141312423212

192 : 2233314444211123, 3442133112442231

193 : 2233314444211123, 3442433112142231

194 : 2234334144121123, 3342413411242312

195 : 2234114344123321, 3342443111242213

196 : 3344334411221122, 2423141324142313

197 : 3344114411222233, 2423341324141312

198 : 3344134411221322, 2423411324143213

199 : 3344113412241223, 2423341141423132

200 : 3344113412241223, 2423341341422131

201 : 2344133412241123, 4223314141422331

#### 2.3.8.1Lémy

1,A144, A244, ..., A20144 nie sú navzájom izotopné,

2,Každá (4, 4)-Latin array je izotopná z jednou latin array z A144, ..., A20144,

3,Každá (4, 4, 1)-Latin array je izotopná s jednou latin array z A19644, ..., A20144.

4,I(4, 4) = 201, I(4, 4, 1) = 6.

5,U(4, 4, 1) = 5335311421440000, U(4, 4) = 80306439693480000

Vypočítať high order latinských array nie je jednoduché. Niekoľko užitočných permutačných rodín, ktoré zodpovedajú (2r, 2r)-latin array sú

gww(x) = φ (w1−(w2⊕(w1− φ(x)))), gww(x) = φ(w1⊕(w2−(w1⊕φ(x)))),

gww(x) = w1 ⊕ φ(w2 − φ (w1 ⊕ x)), gww(x) = w1 − φ(w2 ⊕ φ(w1 − x)), kde φ je bijektívne zobrazenie. V prípade, ak φ je involúcia( φ-1 = φ), tak aj qw sú ivolúcie.

#### 2.3.8.2 Príklady

1,Nech latin array B= je izotopná s latin array A4444 preskupením prvého, druhého riadku a štvrtého a siedmeho stĺpca. <(12),(1),(47)>

2,Nech latin array B= je izotopná s latin array A4444 preskupením prvého, druhého riadku a štvrtého, siedmeho a ôsmeho a dvanásteho stĺpca. <(12),(1),(47)(812)>

3,Nech latin array B= je izotopná s latin array A4444 preskupením prvého, druhého a tretieho, štvrtého riadku a štvrtého, siedmeho a ôsmeho, dvanásteho stĺpca, výmenou čísla 1 a 4. <(12)(34),(14),(47)(812)>

### 2.3.9. Aplikácia (4, 4)-Latin array v kryptografii

Nasledovná binárna prúdová šifra nie je bezpečná: Je daný otvorený text x0...xl-1, ktorý je šifrovaný do textu y0...yl-1 podľa pravidla yi = xi⊕wi, i= 0,..., l−1. Kľúčom je reťazec w0...wl-1, ktorý je generovaný binárne lineárnym posuvným registrom.

V skutočnosti, takáto šifra nie je odolná voči plain-chosen útoku.

Predpokladajme, že kľúčový reťazec spĺňa rovnicu wi = a1wi−1⊕...⊕anwi−n, i = 0,..., l−1. Kľúč šifry je a1,...,an, w-n,..., w-1. Ak je možné získať časť otvoreného textu dĺžky 2n, povedzme xj...xj+2n-1, potom wj...wj+2n-1 môže byť počítané ako wi = xi ⊕ yi, i = j,..., j + 2n−1. Preto riešením rovnice wi = a1wi−1 ⊕...⊕ anwi−n, kde i=j+n,..., j+2n−1, koeficienty a1,...,an sú ľahko nájdené. Kľúčové bity wj+2,...,wl-1 môžu byt vypočítané z wj+n,..wj+2n-1 a bity w0,..,wj-1 môžu byť vypočítané z wj...wj+n-1 . Ak opíšeme šifru pomocou vyššie uvedených obrázkov na šifrovanie a dešifrovanie(obrázok 1, obrázok 2), potom Ma  je binárny lineárny posuvný register a permutačná rodina , ktorá zodpovedá (2, 2)latin array. Táto permutačná rodina je však príliš malá. Preto treba zvoliť viac komplexný posuvný register alebo zvoliť permutačnú rodinu zodpovedajúcu latin array vyššej triedy.

V nasledovnom príklade je preto využitá latin array (4, 4).

#### 2.3.9.1 Príklad

Je daný maximálny lineárny posuvný register a (4, 4)latin array. Nech X a Y sú GF(22) = {00, 01, 10, 11}. Nech Ma(obrázok 1) je generátor n-order maximálny lineárny posuvný register so 4 bitovým výstupom. Permutácia qw na GF(22) odpovedá (4, 4)latin array A: pre všetky w = w3w2w1w0 ∈ GF(24), ak prvok z A v stĺpci (w323 + w222 + w12 + w0) + 1 a riadku (x12 + x0)+1 je(y12 + y0)+1, potom gw(x1x0)= y1y0. Kľúč pozostáva z qw, Ma a jeho vstupného stavu. Množstvo kľúčov je počet (4,4)latin arrays, počet primitívnych polynómov stupňa n nad GF(2) a 2n-1. Takáto šifra odolá plain-chosen útoku.

Ak použijeme latin array vyššieho rádu a ak skupina permutácií zodpovedajúcej latin array spĺňa nasledujúce podmienky: pre každé x a y, prvky w, kde gw(x) = y nemôžu vyhovieť žiadnej lineárne rovnici. Kde je w reprezentované ako vektor dimenzie nad GF(2), potom pre kópiu šifry v prípade (n, k)latin array, charakteristický polynóm Ma môže byť vybratý z kľúča, čiže štruktúra Ma môže byť konštantná a kľúč môže byť zostavený z latin array a vstupného stavu Ma.

## 2.4 Lineárne závislé latin arrays

### 2.4.1. Latin arrays a inverzné funkcie

Nech n a k sú dve kladné celé čísla. Označme r= a N={1,...., N}. Nech A je (n, k)latin array. Vektor [u1,..., ur] nad GF(2) sa nazýva označenie stĺpca (u12r-1 + u22r-2 +···+ ur)+1 z A. Nech x, y ∈ N. Ak komponenty označenia stĺpca A, ktorého prvky na riadku x sú y a tie zodpovedajú určitému nenulovému polynómu r variabilného stupňa ≤ c nad CF(2), potom A je c- závislá s rešpektovaním (x, y),

A je c- závislé s rešpektovaním (x, y) pre každé x, y ∈ N, potom A je c- závislé. Ak A je c- nezávislé s ohľadom na (x, y) pre každé x, y ∈ N, potom A je c- nezávislé. Ak A je c- závislé a nie je (c - 1)- závislé, potom sa c nazýva stupeň závislosti na A, označované cA. Ak A je c- nezávislé a nie je (c +1)- nezávislé, potom sa c nazýva stupeň nezávislosti na A, označované IA. Platí vzťahy cA ≥ IA+1.

Používame Rna označenie priestoru riadkového vektora o rozmere r nad GF(2). Pre každé kladné číslo m, nech fm je jednoprvkové mapovanie z Rna {0,1,..., 2m-1} definované podľa fm(x1,..., xm) = x12m-1 + x22m-2 +...+ xm. Nech φ1 a φ2 sú dve permutácie na R, a φ je zobrazenie na R.Označme Φ = (φ1, φ, φ2). Zostrojme 2r×22r maticu AΦ nad {1,..., 2r} nasledovne: Pre každé x, w1, w2 ∈ R, prvok na riadku fr(x)+1 a stĺpec f2r(w1, w2)+1 je fr(φ1(w1)⊕φ(φ2(w2)⊕x))+1, kde ⊕ je vektorové sčítanie nad GF(2).

φ je permutácia a AΦ je (2r, 2r)-latinský pole z Φ. Nech φ je zobrazenie na Rs prvkami funkcie φ1,..., φ r. Pre každé nezáporné celé číslo c, ak je nenulový polynóm h stupeňa ≤c v 2r premenných nad GF(2) takých, že h(x1,..., xr, φ1(x1, ..., xr),..., φr(x1,..., xr))=0, x1,...,xr ∈ GF(2),φ by malo byť c- závislé a h závislý polynóm z φ. Ak φ nie je c- závislé, potom φ je c- nezávislé. Ak φ je c- závislé a (c-1)- nezávislé, c nazýva závislý stupeň z φ, ktorý označujeme cφ, c-1 sa nazýva nezávislý stupeň φ, označovaný ako Iφ. V prípade, ak φ je 1-závislé, φ je lineárne závislé. Ak je φ1-nezávislé, potom φ je lineárne nezávislé.

Afinné transformácia na Roznačuje zobrazenie xC⊕b, kde C je r×r matica nad GF(2), b je riadkový vektor dimenzie r nad GF(2).

#### 2.4.1.1Tvrdenia a lémy

1,Nech A je (n, k)latin array a r = , potom A

2, AΦ je (2r, 2r)latin array vtedy a len vtedy ak φ je permutácia.

3, Nech φ je zobrazenie na R, p a q sú dve invertibilné afinné transformácie na R. Nech φ´(x)= p(φ(q(x))), x R. Potom cφ=cφ´ a Iφ=Iφ´.

4, Nech φ je transpozícia na Ra φ1, φ2 sú dve invertibilné afinné transformácie na . Nech Φ=( φ1, φ, φ2) AΦ je (2r, 2r)latin array z Φ. Potom cAΦ= cφ, IAΦ= Iφ a cAΦ= IAΦ+1.

5, Pre každú transformáciu φ na R platí cφ ≤c(r), kde c(r) = minc [1++...+>2r].

6,Nech r≥3, potom exituje permutácia φ na R taká že cφ≥2.

7, Pre každé r, 1 ≤ r ≤ 6 existuje permutácia φ na R, kde cφ=c(r).

#### 2.4.1.2 Príklad

1, Nech je daná (4,8)latin array. Potom r = sa rovná 5. Označme vektor u ako [1, 0, 1, 1, 0]. Tento vektor nazývame column label stĺpca 23. Preto prvky na riadku x sú y, ktoré zodpovedajú nenulovému polynómu r- 5 premenných ≤c nad CF(2). Napríklad(01011). V tomto príklade je cA 5 a IA 4.

### 2.4.2 Generovanie lineárne nezávislých permutácií

#### 2.4.2.1 Truth table

Je dané r > 0, nech φ je transformácia na R. Nech Wi je ×r matica nad GF (2), ktorej riadky sa skladajú zo všetkých odlišných vektorov dimenzie r s váhou i, i=0, 1,..., r. Používame It na označenie stĺpcového vektora dimenzie , ktorých všetky komponenty sú 1.Pre všetky i, 0 ≤ i ≤ r, definujeme ×r maticu Ui nad GF (2), ktorej riadok j je hodnota φ na riadku j z Wi, 1 ≤ j ≤ . Definujme 2r×(1+2r) maticu Φ,

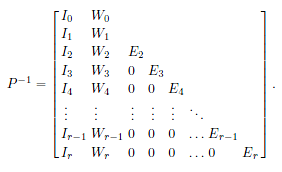
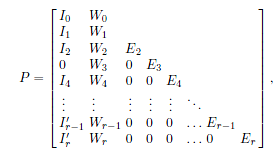
= Φ.

Označme Φ ako **truth table** z φ a označujeme submaticu posledných r stĺpcov z Φ ako Uφ. Uveďme, že W0 =0. Pre uľahčenie usporiadajme riadky W1, takže vznikne identická matica. Z definície vyplýva nasledovné:

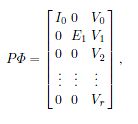
(a)cφ >1 práve vtedy ak stĺpce Φ sú lineárne nezávislé

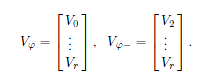
(b) φ je invertibilný práve vtedy ak riadky Uφ sú odlišné

Nech Et je identická matica . Nech 2r2r je matica P. Kde I=Ij ak j je párne, I=0 inak. P je regulárna matica a P-1 :



Prvých 1+r stĺpcov P-1 a Φ sú identické, preto platí:

 kde Vo= U0, Vi je je matica, i=1,...,r.Nech Vφ, Vφ-1 sú



P je regulárne. Stĺpce Φ sú lineárne nezávislé práve vtedy, keď stĺpce PΦ sú lineárne nezávislé. Stĺpce PΦ sú lineárne nezávislé práve vtedy, keď stĺpce Vφ- sú lineárne nezávislé. Platia nasledujúce tvrdenia:

1, cφ > 1 práve vtedy ak stĺpce Vφ- sú lineárne nezávislé.

2, Pre každé i, 1 ≤ i ≤ r a každú r × r permutačnú maticu Q nad GF(2) existuje jedinečná permutačná matica PiQ, taká že PiQWi = WiQ.

3, Predpokladajme, že V<DQ,δ,C>=V´. Potom P-1V´ =DQ(P-1V)C ⊕ .

4, Predpokladajme, že dve 2r r matice V a V´ sú ekvivalentné.

1. Stĺpce V\_ sú lineárne nezávislé práve vtedy ak stĺpce V\_´ sú lineárne nezávislé
2. Riadky P-1V sú odlišné práve vtedy ak riadky P-1V´ sú odlišné

5, Nech φ a φ´ sú dve transformácie na Ra Vφ a Vφ´ sú submatice posledných r stĺpcov z PΦ a PΦ´, teda kde Φ a Φ´ sú TRUTH TABLES z φ a φ´. Ak Vφ a Vφ´ sú ekvivalentné, potom podmienka cφ >1 a φ je invertibilné platí práve vtedy ak cφ´ >1 a φ´ je invertibilné.

#### 2.4.2.2 S(V0,V1)

Pre každý riadok vektora V0 dimenzie r nad GF(2) a akékoľvek r × r matice V1 nad GF(2), používame S(V0, V1) na označenie množiny 2r × r matíc nad GF(2) takých, že V ∈ S(V0, V1) práve vtedy ak pre podmienky platí: prvý riadok V je V0, submatica riadkov 2 až 1+r z V je V1, stĺpce V\_ sú lineárne nezávislé a riadky P-1V sú rôzne.

Pre akékoľvek transformáciu φ na R, nech Vφ je posledných r stĺpcov PΦ, kde Φ je TRUTH TABLE of φ. Označme prvý riadok z Vφ ako Vφ0, a submaticu pozostávajúcej z riadkov 2 až r + 1 z Vφ ako Vφ1. Ak cφ> 1 a φ je invertibilné, potom Vφ ∈ S(Vφ0, Vφ1). Pre akékoľvek V ∈ S (V0, V1), ak φ je transformácia na R s Vφ = V, potom cφ> 1 a φ je invertibilné.

##### 2.4.2.2.1Teorémy

Nech δ je riadkový vektor dimenzie r nad GF(2), Q je r × r permutácia matice nad GF(2), C je r × r regulárna matica nad GF(2). Potom

S((V0 ⊕ δ)C,QV1C)= {V | V ∈ S(V0, V1)} a |S((V0 ⊕ δ)C,QV1C)| = |S(V0, V1)|.

Pre každé kladné celé číslo r, značme ho Gr´´ = {< Q, C> | Q je r × r permutačná matica cez GF(2), C je r × r regulárna matica cez GF (2}

Nech · je operácia na Gr´´ definovaná <Q, C> ·< Q´, C´> =< QQ´, C´C>. Je zrejmé, že <Gr´´,·> je grupa.

Pre každú r × r maticu V1 nad GF(2) a akékoľvek<Q,C> v Gr´´, označme V1<Q,C> = QV1C. V1 a V1<Q,C> sú ekvivalentné podľa Gr´´. Každá trieda ekvivalencie podľa relácie ekvivalencie nad Gr´´ obsahuje r×r matice vo forme , kde E je jednotková matica, stĺpce B sú v zostupnom poradí v niektorých usporiadaniach.

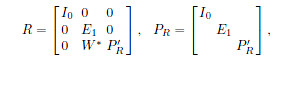
Najmenej jedno usporiadanie sa nazýva kánonická forma triedy ekvivalencie podľa Gr´´.

Riadky V1 sú nenulové vo všetkých prípadoch ekvivalencie. Je zrejmé, S(0, V1) ≠ ∅ znamená, že riadky v V1 sú nenulové a odlišné. Z vyššie uvedených definícií vyplýva, že generácia lineárne nezávislých permutácií môže byť znížená na generovanie S (0, V1), kde V1 má ohraničenie kanonickej formy podľa Gr´´, riadky V1 sú nenulové a odlišné. Napríklad, ak r=4 existujú iba 3 alternatívy:

, , .

##### 2.4.2.2.2 Lémy

1,Nech

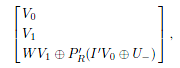


,kde PR´ je (2r-1-r) × (2r-1-r) permutačná matica nad CF(2) a W\* =W⊕P´RW, W=. Potom PRP-1 =P-1R a R zabezpečuje, že splnenie tejto rovnice je jednoznačne určené PR.

Pre všetky 2r × r matice V a V´ nad GF(2) platí:

1. prvých r+1 riadkov v P-1V a P-1V´ je rovnakých, a riadky z P-1V´ sú permutácie riadkov z P-1V,
2. prvých r + 1 riadkov vo V a V´ je rovnakých a existuje (2r-1-r) x (2r-1-r) permutačná matica PR´ taká, že V´\_ = (E´ ⊕ P´R) WV1 ⊕ P´RV\_, kde V\_ a V´\_ sú submatice pozostávajúce z posledného 2r - 1 - r riadkov z V a V´, respektíve, V1 je submatica pozostávajúca z riadkov 2 až r+ 1 z V, E´ je (2r–1-r) × (2r-1-r) jednotková matica, a W=

2, Nech V0 a V1 sú 1×r a r × r matice nad GF(2), Predpokladajme, že riadky V1 sú odlišné a nenulové. Nech U\_ je (2r - 1 - r) x r matica nad GF(2), ktorej riadky sa skladajú zo všetkých odlišných riadkových vektorov dimenzie r vrátane V0 a riadkov I1V0⊕V1. Potom S (V0,V1) je množinu všetkých

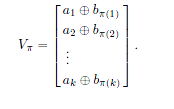


kde PR´ je ohraničená nad (2r-1-r) x (2r-1-r) permutačných matíc takých, že stĺpce WV1⊕P´R(I´V0⊕U\_) sú lineárne nezávislé, kde I1 a I´ sú stĺpcové vektory dimenzie r a (2r - 1 – r), ktorých každý komponent je 1.

#### 2.4.2.3 Problem P(a1,...,ak, b1,...,bk)

Nech je daný vektor V0 dimenzie r nad GF(2) a r × r matica V1 nad GF(2) s odlišnými a nenulovými riadkami. Musíme opraviť (2r-1-r) × r matice U\_ nad GF(2) také, že riadky V0, I1V0 ⊕ V1 a U\_ sú všetky riadkové vektory dimenzie r. Označme A=WV1 a B=I´V0⊕U\_. Z vyššie uvedeneých definícií vyplýva, že problém vytvárania S(V0, V1) je redukovaný na výber (2r-1-r) ×(2r-1-r) permutačných matíc PR´ takých, že stĺpce z A⊕PR´B sú lineárne nezávislé. Toto je zobrazené v nasledujúcom probléme P(a1,...,ak, b1,...,bk).

Problém P(a1,...,ak, b1,..., bk):nech sú dané riadky vektora a1,...,ak, b1, ..., bk dimenzie r nad GF(2). Hľadáme všetky permutácie π na , čiže stĺpce z Vπ sú lineárne nezávislé, kde



Ak k<r, potom problém P(a1,...,ak, b1,...,bk) nemá riešenie. Ďalej teda predpokladáme, že k≥ r.

Pre každú postupnosť (c1,...,cr) a (d1,...,dr), ak existuje i, 1 ≤i ≤ r, také že c1=d1,...,ci-1=di-1 a ci<di, (c1,...,cr) je menšia ako (d1,...,dr). Pre každé riešenie π Problému P(a1,...,ak, b1,...,bk), požadovaná množina všetkých postupností (c1,...,cr), čiže 1≤ c1 <c2 <…< cr ≤ k, riadky c1,…,cr z Vπ sú lineárne nezávislé. Minimálna postupnosť v množine sa nazýva rank- spectrum z π. Potom stĺpce Vπ sú lineárne nezávislé a rank-spectrum z π je definované.

Označme množinu všetkých riešení Problém P(a1,...,ak, b1,...,bk) s rank-spectrum (c1,...,cr) z V(c1,..., cr).

Nech V(c1,...,cr; d1,..., dr)= {π | π ∈ V (c1, . . . , cr), π(c1) = d1, . . . , π(cr) = dr}.

Nech π ∈ V (c1, . . . , cr; d1, . . . , dr). Potom a1 ⊕ bπ(1) = . . . = ac1−1 ⊕ bπ(c1−1) = 0 and ac1⊕ bd1 ≠ 0. Preto ak pre nejaké i<c1, ai je odlišné od každej bj, j=1,..., k , potom V(c1,. . , cr) = ø.

Používame R(c1,...,ci,d1,...,di) na označenie vektorového priestoru generovaného ac1⊕ bd1 ,..., aci ⊕bdi, ktorý je 0-dimenzionálny priestor {0} v prípade ak i=0. Pre každé I ⊆ {1, . . ., k}, nech Π(I, h, c1, ..., ci, d1, ..., di)= {j | j ∈ {1,..., k} \ I, bj ∈ ah ⊕ R(c1,..., ci, d1,..., di)}.

Zvoľme c1,..., cr and d1,..., dr, také že 1 ≤ c1 < c2 < ···< cr ≤ k a d1,..., dr sú odlišné prvky v {1,..., k} označené c0 = 0, cr+1 = k+1.

Nech Hi = {h | ci < h < ci+1}, i = 0, 1,..., r.

Definujme reláciu ~i na Hi ,h ~i h´ ⇔ ah ⊕ ah´ ∈ R(c1,..., ci, d1,..., di), 0 ≤ i ≤ r. Relácia ~i je reflexívna, symetrická a tranzitívna. Používame Hi1, Hi2,..., Hiti na označenie všetkých tried ekvivalencie z ~i na Hi.

##### 2.4.2.3.1 Lémy

1,Predpokladajme, že 1≤ c1<c2<…<cr≤ k a že d1,..., dr sú rôzne prvky v {1,..., k}. Nech I ⊆ {1,..., k}, potom pre každé i, 0 ≤ i ≤ r, a každé h, h´ Hi máme

(a) ak platí h ~i h´, potom Π(I, h, c1,..., ci, d1,..., di)=Π(I, h´, c1,..., ci, d1,...,di)

(b) ak neplatí h ~i h´, potom Π(I, h, c1, . . . , ci, d1, . . . , di) a (I, h´, c1, . . . , ci, d1, . . . ,di) sú disjunktné.

2, Predpokladajme, že 1≤ c1<c2<…<cr≤ k a že d1,..., dr sú odlišné prvky v {1,..., k}. Nech I ⊆ {1,..., k}. Potom pre každé i a i´, 0≤ i´ < i ≤ r, každé h Hi´a každé h´ Hi´, Π(I, h´, c1,..., ci´, d1,..., di´) je podmnožina Π(I, h, c1,..., ci, d1,..., di) alebo sú disjunktné.

3, Predpokladajme, že 1≤ c1<c2<…<cr ≤ k a že d1,..., dr {1, . . ., k} sú odlišné. Nech I={d1, . . ., dr}. Potom pre každú transformáciu π na {1, . . ., k}, π je vo V(c1,...,cr; d1,...,dr) práve vtedy ak platia nasledovné podmienky

1. π(cj) =dj, j=1,...,r
2. ac1 ⊕ bd1,...,acr ⊕ bdr sú lieárne nezávislé

každé i, j, 0 ≤ i ≤ r, 1 ≤ j ≤ ti, | π(Hij) | Hij | a π(Hij) ⊆ Π(I, hij, c1, ... , ci, d1,... , di) \

SUMA SLAJD 347

Kde hij je ľubovoľný prvok z Hij.

Riešenia π a π´ z problému P(a1,...,ak, b1,...,bk) sú ekvivalentné ak rank- spectrum z π a π´ sú zhodné, hovoríme (c1,...,cr) a π(ci) = π´(ci) pre i=1,...,r, π(Hij) = π´(Hij) pre i=0,1,..., r a j=1,...,ti. Táto relácia je reflexívna, symetrická a tranzitívna.

##### 2.4.2.3.2Dôsledky

1, Ak V(c1,..., cr, d1,..., dr) ≠ø, potom počet riešení v každej ekvivalentnej triede je a ekvivalentnú triedu obsahujúcu π možno získať obmedzením π na Hij na bijekciu z Hij na π(Hij) pre i=0,1,...,r a j=1,...,ti.

2,Predpokladajme, že 1≤ c1<c2<…<cr ≤ k a že d1,..., dr {1, . . ., k} sú odlišné. Nech I={d1,..., dr}. Potom existuje i, j, 0≤ i< r, 1 ≤ j ≤ti , také že + >

, kde hij Hij , potom V(c1,..., cr; d1,..., dr) =ø.

Hij and Π(I, hij, c1, . . . , ci, d1, . . . , di) nezávisí na parametroch ci+2, . . . , cr, di+1, . . . , dr, preto tvrdenie môže byť zovšeobecnené nasledovne:

3, Nech 0 ≤ i < r. Predpokladajme, že 1 ≤ c1 < c2 < · · · < ci+1≤ k − r + i + 1 a že d1, . . . , di ∈ {1, . . . , k} sú odlišné. Nech I = {d1, . . .,di }. Ak existuje j, 1 ≤ j ≤ ti také že |Hij | +

> , kde hij ∈ Hij , potom pre každé ci+2, . . ., cr, di+1, . . ., dr máme V(c1, . . ., cr; d1,. . ., dr) = ø.

3, Predpokladajme, že 1≤ c1<c2<…<cr ≤ k a že d1,..., dr {1, . . ., k} sú odlišné. Nech I={d1,..., dr}. Pre každé i, j, 1≤ i< r, 1 ≤ j ≤ti , nech qij= |Hij| a sij = -

,kde h je ľubovoľný prvok z Hij

1. V(c1, . . . , cr; d1, . . . , dr) ≠ø práve vtedy ak ac1⊕bd1 ,..., acr ⊕bdr sú lieárne nezávislé a (∀i)(∀j)[0 ≤ i ≤ r & 1 ≤ j ≤ ti → qij ≤ sij ].
2. Pre neprázdnu V(c1, . . . , cr; d1, . . . , dr) počet ekvivalentných tried je SUMA 349 a počet prvkov je .

# 3. Konečné automaty a asymetrický kryptosystém

Každé celé číslo i, kladné celé číslo k a akékoľvek symbol reťazca, na označenie symbolu reťazca zi-1,..., zi-k+1 budeme používať z(i, k). Pre každý (r, t)-order memory konečný automat M≤ X, Y, S, δ, λ>, akákoľvek (r´, t´)-order memory konečný automat M´=< X, Y, S´, δ´, λ´> a každé nezáporné celé číslo τ, používame PI (M, M´, τ). Platia nasledovné podmienky:

Pre každý stav s =< y(-1, t), x(-1, r)> z M a akejkoľvek stav s´ =<x(-1, t‘), x(τ -1, r‘)> z M‘, a akékoľvek x0, x1,...∈ X, yτ, yτ+1,...∈ Y, ak y0y1...= λ(s, x0x1...), potom x0x1...=λ‘(s‘, yτyτ+1...).

Pre každé i, 0 ≤ i ≤ n, nech Xi je stĺpcový vektor nad GF(q) z dimenzie li. Nech Y je stĺpcový vektor nad GF(q) dimenzie m a X = Xn.

Pre každé i, 1 ≤ i ≤ n, nech Mi =<Xi, Xi-1, X, δi, λi> je ri-order input memory konečný automat, M =<Xi-1, Xi, X×X, δ, λ> je (τi, ri)-order memory konečný automat, a τi ≤ ri.

Nech M0 =<X0, Y, Yt0 ×X, δ0, λ0>je (r0, t0)-order memory konečný automat, M =<Y, X0, X\*×Yr0\*, δ, λ> je (r, t)-order memory konečný automat, a τ0≤ r0.

## 3.1 Teorémy

1, Predpokladajme, že M , Mi a τi zodpovedajú PI(M ,Mi, τi), i = 0,1,..., n. Nech s=<x(i)(−bi−1 −1, ri), (i−1)(−bi−1 − 1, τi)> je stav z M , i = 1,..., n, a nech x ,..., x∈ X0

x ... x = λ (s, x...x)

s = δ(s,x ... x

i = 1,..., n, kde b0 = r0 −τ0, bi = bi−1 +ri −τi, i = 1,..., n.

Nech s0 =< x(0)(−1, t),y(−1, r∗0)> je stav z M.

Ak x0(0)x1(0)... = λ (s0(0)\*, y0y1...),

X0(i)x1(i)= λ∗i (s0(i)∗, x0(i−1) x1(i−1)...), i = 1,..., n,

y0y1...= λ‘(s‘, xτ(n) x...),kde λ‘ výstupná funkcia z C´(Mn,..., M1, M0), s´ = <y(−1, t0), x(n)(τ − 1, r)>, r = r0 +...+rn, a τ = τ0 +...+ τn.

2, Predpokladajme, že PI( Mi, M, τi), i=0, 1, ..., n. Nech s´ = <y(−1, t0), x(n)(−1, r)> je stav z C´(Mn, ,..., M1, M0) s výstupnej funkcie λ‘, kde r=r0+...+rn. Nech x ...x= λi(<x(i)(−r0 −···− ri−1 − 1, ri)>, x ...x), pre i=n, n-1,..., 1 a y0y1...=λ‘(s‘, x...). Ak x0 x...= λ(<x(0)(−1, t), y(τ0 − 1, r)>, yτ0yτ0+1...), pre i=1,..., n-1, potom xx...= λ(<x(n)(1-,rn),x(n-1)(τn − 1, τn)>, x x ...).

## 3.2 Šifrovacie algoritmy

Bežné šifrovacie systémy(jedno kľúčový šifrovací systém) je rodina dvojice šifrovacích a dešifrovacích algoritmov. Každý algoritmus v akýkoľvek dvojici je indexovaný jeho kľúčom. Kľúče šifrovacieho algoritmu a kľúče zodpovedajúce dešifrovaciemu algoritmu sú rovnaké, alebo nasledujúci kľúč možno ľahko odvodiť z predchádzajúceho. Bežné šifrovacie systémy vyžadujú aby medzi sebou príjemca a odosielateľ udržovali kľúč v tajnosti.

Verejné kľúčové šifrovacie systémy sú rodina dvojíc algoritmov, označená {(Ek, Dk), k ∈ K}, spĺňajúce podmienky:

1. pre každé k ∈ K, Dk je inverzný k Ek (pre dôverné aplikácie), a / alebo je Ek inverzný k Dk (pre overovacie aplikácie),
2. pre akékoľvek k ∈ K, Ek a Dk sú ľahko vypočítateľné,
3. pre takmer každé k ∈ K, je nemožné ľahko odvodiť ekvivalentne zodpovedajúce Dk z Ek
4. v každé k ∈ K, je možné vypočítať pár Ek a Dk.

V aplikáciách šifrovacích systémov verejného kľúča si každý užívateľ zvolí pár Ek, užívateľský verejný kľúč, a Dk, užívateľský súkromný kľúč, používateľ zanechá Ek verejnosti a DK udržuje v tajnosti.

Šifrovací systém s s verejným kľúčom môže byť navrhovaný pre dôverné a overovacie aplikácie. Je požadované aby m = l0 =···= ln. Stručne budeme označovať Xi ako X. Vyberme všeobecné q a m pre všetkých užívateľov. Nech obe abecedy X a Y sú priestor stĺpcového vektora nad GF(q) dimenzie m. Otvorený textový priestor X\* a priestor zašifrovaného textu Y\* sú zhodné. Používateľ, označme ho A si vyberá vlastný verejný a privátny kľúč.

1. Zostrojme (r, t)-order memory konečný automat M= <Y , X, S, δ, λ> a (r0, t0)-order memory konečný automat M0= <Y , X, S0, δ0, λ0> spĺňajúce podmienky PI(M, M0, τ0) a PI(M0,M,τ0).
2. Pre každé i, 1 ≤ i ≤ n, zostrojme ri-order input konečný automat Mi = <X, X, Si, δi, λi> a (τi, ri)-order memory konečný automat M = <X, X, S , δ , λ> spĺňajúci podmienky PI(M, Mi, τi) a PI(Mi,M, τi).
3. Zostrojme konečný automat C´(Mn,..., M1, M0)= <X, Y, S, δ, λ> z M0,..., Mn.
4. Nech b0 = r0−τ0, bi = bi−1+ri−τi, i = 1,..., n. Predpokladajme, že b0 =... = bc−1 = 0, napríklad., rj = τj , j = 0, . . . , c−1, pre ľubovoľné c, 0 ≤ c ≤ n. Vyberme ľubovoľné y−1,... , y−t0 ∈ Y, x,..., x∈ X. Pre každé i, c+1 ≤ i ≤ n, vyberme ľubovoľný stav s= <x,..., x,..., ,..., z M.
5. Počítajme x ...x = λs, x ...x) a  s0 = δ (s, x ...x) pre i=c+1,..., n. Vezmime s = <y−1,..., y−min(t0,r)>,

s = <x ,..., x>,

s= s, i= c+1,..., n, s=<y-1,...,y-to>, s s =<x,..., x>

Zvoľme ľubovoľné y−1,..., y−r +r0 ∈ Y a x,..., x ∈ X, kde r´ = r+ r1 +... + rn, r= min(r0, t). Počítajme x, ..., x=λi(<x,..., x>,x,..., x), i= n, n-1,..., 1.

Vezmime s=<y-1,..., y –min(r-τ, t), s = <x,..., x>, s = <x,..., x>, i=1,..., n.

1. Verejný kľúč používateľa A je C´(Mn,..., M1, M0), s, s τ0 +.... +τn. Tajný kľúč užívateľa A je M,...,M, ss, ss, ss..., , ss sdsdsdsdτ0,..., τn.

## 3.3 Šifrovanie

Užívateľ (B) chce poslať otvorený text x0,..., xl užívateľovi A. B najskôr indexuje každé τ0 + ... + τn číslo, povedzme xl+1 ...xl+τ0 +···+τn. Potom používateľov(A) verejný kľúč C´(Mn ,..., M1, M0), s=<y-1,…, y-min(r− τ0, t0)> a s=<x,…, x>.

Používateľ B počíta zašifrovaný text y0...yn+τ0+···+τn nasledovne:

y0...yn+τ0+···+τn= λ´(s´, x0... xl+τ0+···+τn), kde s´= <y−1,..., y−t0, x, ..., x>, x,..., x >, x,..., xje náhodne zvolený z X, kde t< r0 a y-r + 0 -1 ,...,y-t0 sú náhodne zvolené Y, kde r- 0 < t0

### 3.4 Dešifrovanie

Zo zašifrovaného textu y0...yn+τ0+···+τn, používateľ A môžeopäť získať otvorený text nasledovne: Použijeme M,..., M, s=< x,…, x> , s==<y-1,…, yr+ τ0)>, s=<x,…, x>, i= 1,..., n v priv8tnom kľúči používateľa, A vypočíta:

xx... x=λ(<x,..., x,yτ0 -1,..., ,y>, y y... y, xx... x = λ(<x,..., x, x ,..., x>, x x... x,i=1,..., n, kde x,...,x môže byť ľubovoľné ak r0 < *t*. Otvorený text x0...xl je rovný x.

Kryptosystémy s verejným kľúčom založené na vyššie uvedených konečných automatov označujeme skratkou FAPKC. Jednému otvorenému textu môže zodpovedať niekoľko zašifrovaných textov. Pri šifrovaní môžu mať niektoré číslice pôvodného stavu(stavov) ľubovoľné hodnoty. Počet týchto písmen je označovaný ako voľnosť, ktorej hodnota závisí od zvolených parametrov. Napríklad pri voľbe FAPKC3, v špeciálnom prípade kde n=1, rt0+ 0 a *t=* r0, potom je voľnosť pre šifrovanie rovná nule, pri voľbe FAKP je hodnota voľnosti 20.

Majme konečné automaty Mi a M spĺňajúce podmienku PI(M, Mi, i) PI(Mi, M, i). Nech p=-1, potom M je (r, t)-order memory konečný automat Mf definovaný podľa

yi = f(yi−1, ..., yi−t, xi,..., xi−r), i = 0, 1,...

Predpokladajme, že eq0(i) −yi + f(yi−1,..., yi−t, xi,..., xi−r) je rovné 0 a že

eqk(i) eqk‘(i), eqk‘ eqk+1(i), k = 0, 1,..., τ−1 je Ra Rb transformácia postupnosti. Nech f je jednohodnotové mapovanie z Xr ×Yr+t+1 do X a M\* =<Y,X, Xr ×Yr+t+1, δ , λ\* > je konečný automat M, definovaný podľa vzťahu

xi = f(xi−1,..., xi−r, yi´,..., y´i−τ−t), i = 0, 1,...

Ak eqτ (i) má riešenie f, napríklad pre každý parameter xi−1,..., xi−r, yi+τ,...,

yi−t, eqτ(i) má riešenie xi = f(xi−1,..., xi−r, yi+τ,..., yi−t), potom je PI(M\*, M, τ). Ak eqτ má najviac jedno riešenie f, napríklad pre každé xi,..., xi−r, yi+τ,..., yi−t, eqτ obsahuje xi = f(xi−1,..., xi−r, yi+τ,..., yi−t), potom dostaneme PI(M, M\*, τ). Práve preto, ak eqτ(i) označuje jednohodnotové mapovanie f, potom dostaneme PI(M\*, M, τ) a PI(M, M\*, τ).

## 3.5 Slabé kľúče

### 3.5.1 Test lineárnej transformácie Ra, Rb

C´(Mn, ..., M0) je vo verejnom kľúči FAPKC (r0 + ... + rn, t0) –order memory konečný automat. Nech C´(Mn,..., M0) pre Mf platí, že r = r0+...+ rn a t = t0. Nech Ra Rb je lieárna transformačná postupnosť, kde τ = τ0+·..+τn. Ak pre všetky parametre xi-1,..., Xi-r, yi+τ,..., Yi-t, rovnica eqτ(i) má (alebo má najviac jedno) riešenie xi. Potom (τ+t,r)-order memory konečný automat môže byť konštruovaný z eqτ(i) tak, že C´(Mn,..., M0) je slabo inverzný s oneskorením τ z . Preto by kontrolný proces mal zahrňovať kľúčový generátor z FAPKC na triedenie C´(Mn,..., M0), z ktorých slabá inverzia môže byť získaná z lineárnej Ra, Rb transformačnej metódy. Ak je lineárna Ra Rb transformácia spĺňajúca vyššie uvedenú podmienku, potom C´(Mn,..., M0) musíme vyhodiť. Hoci je počet tých lineárnych Ra Rb transformácií obrovský, len jednu lineárnu Ra Rb transformačnú postupnosť je potrebné skontrolovať vďaka vete: ak pre lineárnu Ra, Rb transformačnú postupnosť platí podmienka, potom pre všetky lineárne Ra, Rb transformačné postupnosti táto podmienka platí.

Kľúče z FAPKC, z ktorých C´(Mn,..., M0) sú vyhodené pre lineárnu Ra, Rb transformačnú metódu sa nazývajú slabé kľúče. Pre n=1 nasledujúce prípady sú slabé kľúče: lineárne M0, kvázi lineárne M1; lineárne M0, nelineárne 01, ktorých slabá inverzia je označovaná lineárnpou Ra Rb transformačnou metódou.

### 3.5.2 Útok redukovaním stupňovitej matice

Inverzné metódy redukovaním stupňovitej matice sú založené na injektívnosti a surjektivite D22ψ(x(i, ν+1)), ktoré sú aplikované pri konečnom automate M. Je to vlastne lineárna transformačná metóda Ra Rb, kde M = <X, Y, Yk ×Xh+ν, δ, λ>, kde

yi = φout(y(i − 1, k)) + [C0,..., Ch]ψ(x, i), i=0,1,...

Len slabé kľúče z FAPKC môžu byť zlomené použitím redukovanej stupňovitej matice.

### 3.5.3 Útok kanonicky diagonálnou maticou polynómu

Nech M = <X, Y, Yk ×Xh+ν, δ, λ> je konečný automat. Nech C(z)=.

Nech zj ψ(xi,...,xi-v) na označenie ψ(xi-j,...,xi-j-v), a zjx na označenie x. Potom [C0,..., Ch]ψ(x, i)= = = C(z) ψ(xi,..., xi-v).

Predpokladajme, že C(z)=D(z)F(z). Nech M1 je konečný automat definovaný podľa x =F(z)(xi,..., xi-v), i=0,1,... a M0 je konečný automat definovaný yi= φout(y(i-1,k))+D(z)x , i=0,1,...

Je zrejmé, že M=C´(M1,M0). Pretože M0 je kvázi lineárny, slabá inverzia môže byť ľahko konštruovaná. Existuje teda realizovateľná inverzná metóda pre konečný automat M, keď existuje inverzná metóda pre M1. Preto, ak pre každý parametre x-1,..., X-ν, F(0)(x0,..., x-ν) je injektívna funkcia premennej x0, potom M1 je slabo invertibilný s oneskorením 0 a slabú inverziu s oneskorením 0 z M1 je možné skonštruovať. Môžeme nájsť 0≤ a1≤ · · ·≤ ar, fj (z), j = 1,..., r a dva Invertibilné maticové polynómy P (z) a Q (z) také, že C(z) = P(z)DIAm,m(za1f1(z),..., zarfr(z), 0,..., 0)Q(z), ft(z) | fj+1(z) pre j = 1,..., r−1 a fj(0) ≠0 pre j= 1,..., r. Vezmime D(z) =P(z)DIAm,r(za1,..., zar) a F(z) = DIAr,m(f1(z),..., fr(z))Q(z). Potom C(z)=D(z)F(z) je označená ako derivačná faktorizácia typu 2 kanonickej diagonálnej matice mnohočlena.

Podobne, C(z) = D´(z)F´(z) sa označuje ako derivačná faktorizácia typu 1 kanonickej diagonálnej matice mnohočlena, kde D´(z) = P(z)DIAm,r (za1f1(z),..., zarfr (z)) a F(z) =

DIAr, m(1, ..., 1)Q(z). Ak C(z)= D(z)F(z) je derivačná faktorizácia typu a alebo 2 kanonickej diagonálnej matice mnohočlena a F(0)(X0,..., x-ν) ako funkcia premennej x0, ktorá je injekciou pre všetky parametre x-1,..., X-ν, potom slabá inverznia M môže byť konštruovaná vyššie uvedenými pravidlami. Na druhej strane, potom existuje Ra Rb transformačná postupnosť

Ck(z)C(z), C(z)Ck+1(z), k = 0, 1,..., τ´−1, kde C0(z)= C(z). Nech (z) je submatica prvých r riadkov C(z). Ak F(0)(x0,..., x-ν) je bijektívna funkcia premennej x0 pre všetky x-1,..., x-v, potom (0)(X0,..., x-ν) funkcia premennej x0 je injektívna pre každé x-1,..., x-v, Preto môže byť slabo inverzný automat M konštruovaný pomocou lineárnej Ra Rb transformačnej metódy. Kľúče z FAPKC ktoré môže byť rozbité na kanonické diagonálne matice polynomickej metódy sú slabé kľúče. Preto nie je potrebné zahrnúť kontrolný proces založený na obmedzovaní kánonickej diagonálnej matice polynómu v kľúčovom generátore z FAPKC.

## 3.6 Bezpečnosť

### 3.6.1 Inversion by a General Method

Ak dokážeme nájsť konečný automat M\*, ktorý je slabo inverzný k C´´(Mn,..., M0) s oneskorením τ0+···+τn, potom môžeme získať otvorený text z textu zašifrovaného.

Pre stav, výpočtové množstvo je O (qm (τ0 + · · · + τn)) (O (2160) pre n = 1, q = 2, m = 8, τ0 + τ1 = 20).

Preto táto metóda je nepraktické pre stredné oneskorenie τ0 +···+ τn.

Ak môžeme nájsť konečný automat M\*, z ktorého C´(Mn,..., M0) je slabo inverzný s oneskorením τ0 + ···+ τn, potom možno sfalšovať podpisy na správy. Pre všeobecný nelineárny konečný automat dokážeme použiť inverzný algoritmus, ktorý vyžaduje výpočet vstupný -strom s úrovňou τ0 +···+τn pre každý štát a každý výstup dĺžky τ0 +···+τn+1 Preto táto meróda zaberie veľa výpočtového času a úložného priestoru.

Ďaľšia nevyhovujúca metóda je inverzia dekompozície konečných automatov, kde často nie je možné nájsť faktorizáciu matice polynómu. Pri útoku otvoreným textom je problém redukovaný na problém nájdenia nelineárnej sústavy rovníc nad GF(q). Nájdenie takýchto automatov je však tažký problém.

### 3.6.2 Exhausting Search and Stochastic Search

#### 3.6.2.1 Exhausting Search Attack

Algoritmus šifrovania je verejne známy, ak sa niekomu podarí dešifrovať zašifrovaný text, vravíme že získal virtuálny otvorený text. Predpokladajme, že verejný kľúč kryptosystému založeného na konečných automatoch je sekvenčný. Jeho bloková dĺžka m je vzhľadom na malú dĺžku kľúča. Malá dĺžka bloku spôsobí nestabilnosť kryptografického systému pri pokusoch o útok. V skutočnosti môže proces redukovať na hádanie časti otvoreného textu o dĺžke τ0 + ···+ τn + 1 a rozhodovanie o jeho prvej číslice. Hadáme teda hodnotu prvých τ0 +···+ τn + 1 číslic otvoreného textu. Následne ho zašifrovať pomocou verejného kľúča a porovnať výsledok s prvými τ0 + ···+ τn +1 číslicami virtuálneho zašifrovaného textu. Ak sa zhodujú, potom prvé číslice otvoreného textu sú v skutočnosti prvé číslice z virtuálneho textu. Tento proces opakujeme dokedy nezískame všetky číslice otvoreného textu.

#### 3.6.2.2Stochastic search attack

Vyššie uvedený **Exhausting** útok je deterministický algoritmus. Tento útok môže byť upravený na algoritmus stochastického typu.

1,vstupy: stavy M =<X,Y,S, δ, λ> , výstupná postupnosť y0y1...yl ∈ W

2, výstupy: vstupná postupnosť x0x1...xl ∈ I.

y0y1...yl,s.

3, postup

1. Nastavme i=0.
2. Nastavme X={x|x ∈ X, yi = λ(δ(s, x x),x)} v prípade ak i>0, alebo { x|x ∈ X, yi = λ(s, x)} inak
3. Ak X ≠ ø, potom vyberme prvok v tom ako x , pripočítajme i o 1 prejdime na krok 4, inak inak vyzve zlyhanie informácie a zastavenie
4. Ak i>l, potom x ...x je vstup x0,...,xl a zastav, inak chod na krok 2

Nech p je pravdepodobnosť úspešného vybratia x ...x v algoritme. Nech pr(x,s,y0...yl), i=0, 1, ...,l, kde p =.

X a Y obmedzíme na Fm a automat M má formu C (0, DX,r1, 1, DX,r 2, 2,..., τ-1, DX, r τ, τ) kde 0≤ r1≤ r2≤...≤ rτ≤ m, i je slabo invertibilný automat s oneskorením 0 pre i = 0, 1,. . . , Τ, a i je (m - ri+1) preservable pre i = 1,. . . , Τ - 1. Označme

X=I a = p/p = p, i=0, 1, ...,l

Kde rj=0, pre j ≤ 0, Preto

p ==

Algoritmus 2-stochastic search algoritmus

yi...yl(*δ(s,*x ...x

r

t0, , τ0, l+r0+...+ rn,

τ0 -*r...* τ0 +1

Subset- podmnožina; Conditions- podmienky, okolnosti ; initial-počiatočný; uniquely determined by- jednoznačne určený; prove- dokázať, preukázať; denote-znamenať, značiť, označiť; infinite -nekonečný; sequence –postupnosť; bounded –ohraničený; assume- predpokladať; label -označiť; Mutual- obojstranný; inconvenient.- nevyhovujúci; desired –požadovaný; viewpoint –hľadisko, stanovisko; rearranging –preskupiť; concatenation –zreťazenie; arbitrary –ľubovoľný; distinct –jednoznačný; rearranging –preskupiť; within –vnútri, počas, v rozsahu; occurs –nastane; repeatedly –opätovne; occurrence –náhoda, udalosť, jav; multiplicity –mnoho početnosť, množstvo; ranging- zaradenie; undirected, -neorientovaný; omit –vynechať; enumeration –vyčíslenie, vymenovanie; column label –názov stĺpca;

δ, φ, ,, , γ

cW0U0I1W1U1TrWrUr