Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE

Marcel Čurilla

Analýza proudové šifry QUAD

Katedra Algebry

Vedoucí diplomové práce:	doc. Mgr. Štěpán Holub, Ph.D.
Studijní program:	Matematika
Studijní obor:	Matematické metody informační bezpečnosti

Praha 2012

Ďakujem môjmu vedúcemu, doc. Mgr. Štěpánovi Holubovi, Ph.D. za vedenie tejto práce, prínosné konzultácie a trpezlivosť. Ďakujem, Ing. Andrei Szabóo, Ph.D., za pomoc a povzbudenie.

Prohlašuji, že jsem tuto diplomovou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ..... Podpis autora

Název práce: Analýza proudové šifry QUAD
Autor: Marcel Čurilla
Katedra: Katedra algebry
Vedoucí diplomové práce: doc. Mgr. Štěpán Holub, Ph.D.

Abstrakt: Prúdová šifra QUAD bola predstavená na Eurocrypte autormi Côme Berbain, Henri Gilbert a Jacques Patarin [1]. Ukázali redukciu tejto šifry na problém riešenia m kvadratických rovníc n premenných nad konečným telesom známy, ako MQ problém. Pre zjednodušenie, autori uvažovali len prípad telesa GF(2). V tejto práci predstavím túto prúdovú šifru. Uvidiem dôkaz (redukciu) bezpečnosti šifry QUAD na MQ problém nad ľubovoľným konečným telesom GF(q). Popíšem základné metódy pre riešenie systému kvadratických rovníc nad konečným telesom, linearizáciu a relinearizáciu. Podrobnejšie sa budem venovať algoritmu XL, momentálne najrýchlejšiemu algoritmu na riešenie kvadratických systémov. V analýze šifry QUAD ukážem pre ktoré instancie je šifra QUAD prelomiteľná a naopak pre ktoré instancie je bezpečnosť zaručená.

Klíčová slova: prúdová šifra , QUAD, MQ problém, algoritmus XL

Title: Analysis of the stream cipher QUAD
Author: Marcel Čurilla
Department: Katedra algebry
Supervisor: doc. Mgr. Štěpán Holub, Ph.D.

Abstract: Stream cipher QUAD was introduced in 2006 on Eurocrypt by Côme Berbain, Henri Gilbert a Jacques Patarin cite quad. The authors showed a reduction of this cipher for the problem of solving m quadratic equations of n variables over finite fields known as the MQ problem. For simplicity, they considered only the case of the field GF(2).

In this thesis I introduce this stream cipher. I show the proof (reduction) of safety ciphers QUAD for MQ problem over any finite field GF(q). I describe the basic methods for the solution of system of quadratic equations over finite fields, linearization and relinearization. I focus on XL algorithm - which is currently the fastest algorithm for solving quadratic systems. This algorithm was designed precisely to deal with overdefined quadratic systems. While analyzing the cipher QUAD I show for what instance is a cipher QUAD breakable and vice versa for what instance is the security guaranteed.

Keywords: stream cipher, QUAD, MQ problem, algorithm XL,

Obsah

1 Úvod **4**

2 Základne poznatky z kryptografie **6**
2.1 Prúdová šifra 6
2.2 Kryptoanalýza 8
2.3 Prúdová šifra ako aproximácia Vernamovej šifry 10
2.4 Generátor pseudonáhodných čísel 10
2.5 Prúdová šifra s inicializačným vektorom `IV` 13
2.6 Generátor pseudonáhodných funkcií 13
2.7 Bezpečnosť prúdovej šifry 14

3 MQ problém **16**
3.1 Definícia MQ problému 16
3.2 MQ je $\mathcal{NP}$ úplný problém 17

4 Algoritmy pre riešenie polynomiálnych systémov **19**
4.1 Linearizácia 20
4.2 Relinearizácia 22
4.3 Algoritmus XL 24
4.3.1 Analýza algoritmu XL 25

5 Popis šifry QUAD **31**
5.1 Generovanie keystreamu 31
5.2 Generovanie inicializačnej hodnoty $\boldsymbol{x_0}$ z K a `IV` 32

6 QUAD: Preukázateľne bezpečná šifra **35**
6.1 Bezpečnosť generátora keystreamu šifry QUAD 35
6.1.1 Z rozlišovača šifry QUAD, prediktor lineárnej formy 36
6.1.2 Bezpečnosť generátora keystreamu šifry QUAD v telese GF(2) 39
6.1.3 Dôkaz bezpečnosti generátora keystreamu šifry QUAD 44
6.1.4 Zhrnutie dôkazu 44
6.2 Bezpečnost prúdovej šifry QUAD s `IV` 47
6.2.1 Bezpečnostné požiadavky pre prúdovú šifru s `IV` 47
6.2.2 Jednoduchá veta o skladaní 48
6.2.3 Konštrukcia PRF z PRNG pomocou binárneho stromu 52
6.2.4 Konštrukcia bezpečnej prúdovej šifry 55
6.2.5 Aplikácia na šifru QUAD 56

7 Klasifikácia šifry QUAD podľa hodnôt parametrov **59**
7.1 Rozdelenie paramaterov 59
7.2 QUAD s prelomiteľnou bezpečnosťou 59
7.3 QUAD s nedokázateľnou bezpečnosťou 61

7.4 QUAD s (ne)preukázateľnou bezpečnosťou 62
7.4.1 Parametre pre generátor keystreamu šifry QUAD 62
7.4.2 Parametre pre šifru QUAD s `IV` 63

8 Záver **65**

Literatúra **67**

1 Úvod

Konštrukcia preukazateľne bezpečnej prúdovej šifry, nie je novou témou. Napriek tomu konštrukcia prakticky použiteľnej a zároveň preukazateľne bezpečnej prúdovej šifry, je otvorený problém. Existuje veľké množstvo preukazateľne bezpečných generátorov pseudonáhodných čísel (PRNG), ktorý z krátkej postupnosti znakov vygeneruje dlhú postupnosť znakov, nerozlišiteľnú od náhodne vybranej postupnosti. Tento dlhý reťazec, môže byť použitý ako keystream u prúdovej šifry. Nevýhoda konštrukcie prúdovej šifry z bezpečného PRNG, je, že tieto PRNG niesú príliš efektívne, aby nám poskytli prúdovú šifru pre praktické použitie. Prebieha úsilie o zrýchlenie týchto PRNG. Napríklad v článku [20] je predstavená preukazateľne bezpečná prúdova šifra BMGL, ktorej bezpečnosť sa redukuje na bezpečnosť šifry AES. Táto šifra je v porovnaní s inými preukazateľne bezpečnými šiframi veľmi rýchla. Problém je, že bezpečnosť tejto šifry spočíva na bezpečnosti šifry AES a nie, na nejakom jendoduchom dobre známom $\mathcal{NP}$ ťažkom probléme.

Bezpečnosť šifry QUAD, spočíva práve na takomto probléme. Bezpečnosť šifry QUAD sa redukuje na problém riešenia $2n$ kvadratických rovníc n premenných nad konečným telesom, nazývaný MQ problém. Je to jednoduchý, dobre preskúmaný problém, ktorý už pre malé hodnoty $n > 100$ sa stáva nedosažiteľným.

V úvode tejto práce uvedieme formálne požiadavky, pre bezpečnu prúdovú šifru. Také, aby generátor keystreamu prúdovej šifry, bez inicializačného vektora, bol bezpečný PRNG. A generátor keystreamu prúdovej šifry, s inicializačným vektorom, bol bezpečný generátor pseudonáhodných funkcií (PRF).

V kapitole 3 zadefinujeme MQ problém a dokážeme, že patrí do triedy $\mathcal{NP}$ úplných problémov. Predstavíme algoritmy na riešenie MQ problému nad konečným telesom, kde počet rovníc je väčší, ako počet neznámych (pre-definovaný). Zameráme sa na algoritmus XL, ktorý sa momentálne zdá byť najrýchlejším algoritmom na riešenie pre-definovaných kvadratických systémov. V oddiely 4.3.1 stanovíme nútnu podmienku pre funkčnosť algoritmu XL.

V kapitole 5 definujeme šifru QUAD. V kapitole 6 ukážeme redukciu celej šifry QUAD na MQ problém. Túto redukciu bezpečnosti šifry QUAD môžeme rozdelíiť do 3 častí.

1. Najprv v podkapitole 6.1 ukážeme, že bezpečnosť generátora (PRNG) $\mathbf{G}$ (PRNG), ktorý z inicializačnej hodnoty vygeneruje keystream, sa redukuje na problém riešenia systému kvadratických rovníc.

2. V podkapitole 6.2 ukážeme, že generátor (PRF) f, ktorý z tajného kľúča a inicializačnho vektora vygeneruje inicializačnú hodnotu, sa redukuje na problém riešenia systému kvadratických rovníc.

3. Nakoniec podľa jednoduchej vety o skládaní 6.9 dostávame, že $\mathbf{G} \circ f$ je generátor pseudonáhodných funkcií, ktorého bezpečnosť sa dá zredukovať na MQ problém (riešenie systému kvadratických rovníc nad konečným telesom).

V poslednej kapitole, rozdelíme šifru QUAD, podľa parametrov pre ktoré je šifra QUAD prelomiteľná (existuje uskutočniteľný útok), nedokázatená (existuje uskutočniteľný útok na príslušný MQ problém) alebo (ne)preukazateľne bezpečná.

2 Základne poznatky z kryptografie

V tejto kapitole stručne predstavíme základne pojmy z kryptografie týkajúce sa najmä prúdových šifier. V úvode kapitoly definujeme prúdovú šifru. Zadefinujeme pojem preukázateľne bezpečná šifra a ukážeme základnú myšlienku dôkazu tejto vlastnosti. Definujeme generátor pseudonáhodných čísel (PRNG) a generátor pseudonáhodných funkcií (PRF). Na záver podáme formálne požiadavky pre bezpečnú prúdovú šifru.

Kryptológia je vedná oblasť zaoberajúca sa skúmaním bezpečnostných aspektov komunikácie. Základnou úlohou kryptografie je umožniť dvom ľuďom, ktorých nazveme Alica a Bob, komunikovať po nezabezpečenom kanále (telefónna linka, počítačová sieť), takým spôsobom aby protivník, ktorého nazveme Oscar, nedokázal tejto komunikácii porozumieť. Rozvoj technológie podnietil rozšírenie poľa pôsobnosti kryptografie aj na ďalšie bezpečnostné požiadavky. Napríklad integrita alebo nepopretie autorstva, aplikačné oblasti ako sú protokoly pre autentifikáciu, digitálne podpisy, elektronické voľby a podobne.

Kryptológia sa delí na dve hlavné časti a to na kryptografiu a kryptoanalýzu. Kryptografia sa zaoberá návrhom konštrukcií (algoritmov, protokolov) spĺňajúcich konkrétne bezpečnostné požiadavky. Úlohou kryptoanalýzy je skúmať možnosti útokov na tieto konštrukcie.

Cieľom tejto časti je predstaviť základné pojmy z kryptológie a to predovšetkým o prúdových šifrách. Začneme s formálnou definíciou prúdovej šifry a ďalej, trochu neformálnym spôsobom ukážeme, aké vlastnosti musí splňovať *bezpečná* prúdová šifra. Zadefinujeme pojem preukázateľne bezpečná šifra a ukážeme typický postup pri dôkaze tejto vlastnosti.

2.1 Prúdová šifra

Definícia 2.1. *Prúdová šifra je usporiadaná sedmica* $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{L}, \mathcal{F}, \mathcal{E}, \mathcal{D})$*, pričom platí:*

1. $\mathcal{P}$ *je konečná množina prípustných otvorených správ*
2. $\mathcal{C}$ *je konečná množina prípustných šifrových správ*
3. $\mathcal{K}$ *je konečná množina kľúčov*
4. $\mathcal{L}$ *je konečná množina, nazývaná abeceda keystreamu*
5. $\mathcal{F} = (f_1, f_2, \ldots)$ *je generátor keystreamu. Pre* $i \geq 1$*, platí*

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

6. *Pre každé $z \in \mathcal{L}$, existuje šifrovacia transformácia*[1] $e_z \in \mathcal{E}$ *a dešifrovacia transformácia* $d_z \in \mathcal{D}$ *a platí,* $e_z : \mathcal{P} \to \mathcal{C}$ *a* $d_z : \mathcal{C} \to \mathcal{P}$ *sú funkcie pre ktoré platí,* $d_z(e_z(x)) = x$ *pre všetky* $x \in \mathcal{P}$.

Základnou myšlienkou prúdovej šifry je generovať keystream $\boldsymbol{z} = (z_1, z_2, \ldots, z_n)$, ktorý použijeme na zašifrovanie otvoreného reťazca $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ podľa pravidla

$$(e_{z_1}(x_1), e_{z_2}(x_2), \ldots, e_{z_n}(x_n)) = (y_1, \ldots y_n) = \boldsymbol{y}.$$

Funkcia f_i nám generuje i-ty znak keystreamu z_i, pričom f_i je funkcia kľúča K a prvých $i-1$ znakov otvoreného reťazca

$$f_i(K, x_1, \ldots x_{i-1}) = z_i.$$

Odšifrovanie prebieha podobne, kde zo zašifrovaného reťazca $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)$ a znalosti kľúča K postupne počítame hodnoty $z_i = f_i(K, x_1, \ldots x_{i-1})$ a znaky otvoreného reťazca $\boldsymbol{x} = (x_1, x_2, \ldots, x_n) = (d_{z_1}(y_1), d_{z_2}(y_2), \ldots, d_{z_n}(y_n))$.

V prípade, že keystream nezávisí na otvorenom ani zašifrovanom texte hovoríme o *synchrónnych prúdových šifrách.* Keystream šifier tohoto typu je generovaný, ako funkcia závislá len od kľúča K. Pri dešifrovaní je nevyhnutné, aby príjemca a odosielateľ boli presne zosynchronyzovaní, pretože výpadok jedného znaku šifrovaného textu naruší celý nasledujúci otvorený text. Šifry, ktoré vedia eliminovať takéto chyby, se nazývajú *asynchrónne.*

Príklad 2.2. Nech $\mathcal{P} = \mathcal{C} = \mathcal{L} = \mathrm{GF}(2)$ a šifrovacie a dešifrovacie funkcie sú sčítaním a odčítaním modulo 2. Definujme generátor keystreamu funkcie f_i rekurzívnym vzťahom nasledovne

$$z_{i+4} = f_{i+4}(z_i, \ldots z_{i+4}) = z_i + z_{i+1} \mod 2\,, \quad \text{pre} \quad i \geq 1.$$

Položme hodnotu kľúča $\boldsymbol{K} = (1, 0, 0, 0) = (z_1, \ldots z_4)$. Potom funkcie f_i nám vygenerujú nasledovný keystream

$$1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, \ldots.$$

Samotné šifrovanie otvoreného textu sa riadi vzťahom $y_i = e_{z_i}(x_i) = x_i + z_i \mod 2$. Napríklad pre hodnoty $\boldsymbol{x} = (1, 0, 0, 1, 1)$, dostávame zašifrovaný reťazec

$$\boldsymbol{y} = (e_1(1), e_0(0), e_0(0), e_0(1), e_1(1)) = (0, 0, 0, 1, 0).$$

Pri dešifrovaní postupne odčítavame (sčítavame) znaky šifrového textu s keystreamom a obdržíme pôvodnú správu.

Najčastejším typom prúdovej šifry je prípad, kedy $\mathcal{P} = \mathcal{C} = \mathcal{L} = G$, kde G značí konečné teleso (grupu) a šifrovacie a dešifrovacie funkcie sú obyčajné sčítanie a odčítanie v G

$$e_z(x) = x \oplus z \quad \text{a} \quad d_z(y) = y \ominus z.$$

V tejto práci, pod pojmom prúdová šifra budeme rozumieť synchrónne prúdové šifry práve tohoto typu.

[1] Transformácia je funkcia definujúca spracovanie dát pomocou daného kľúča.

2.2 Kryptoanalýza

Kryptoanalýza je veda zaoberajúca sa metódami získavania obsahu šifrovaných informácií bez prístupu k tajným informáciám, ktoré sú potrebné pri dešifrovaní. Kryptoanalýza je vlastne opakom kryptografie, ktorá šifry vytvára.

Vždy budeme predpokladať, že útočník pozná *šifrovací systém*[2] a jediné čo nepozná je použitý tajný kľúč. Táto požiadavka je jedným z Kerckhoffsovych princípov: *„Bezpečnosť šifry nesmie byť založená na predpokladanej obmedzenosti útočníka.“* Historicky sa počítalo iba s útokmi vedenými iba zo znalosti šifrového textu. Rozvojom moderných komunikačných a počítačových systémov pribudla rada nových možností. Útočník už nemusí byť len pasívny, t. j. sledovať komunikačný kanál, ale môže aktívne zasahovať do šifrovaného textu, komunikáciu môže dokonca sám vyvolať a posielať správy k zašifrovaniu, odstrániť alebo vsunúť časť šifrového textu do správ a podobne.

Prirodzeným požiadavkom na bezpečnosť šifrovacieho systému je, aby útočník nemohol zo znalosti šifrového textu rozlúštiť otvorený text. Pri aktívnych útokoch je tento požiadavok na bezpečnosť šifry nedostačujúci. Pri niektorých šifrovacích systémoch sú požiadavky na bezpečnosť vyššie a vyžaduje sa aby šifrovací systém odolal niekoľkým typom útokov. Medzi základné typy útokov patria:

- COA – *Útok len so znalosťou šifrového textu:* útočník má k dispozícii množinu šifrových textov, pričom nepozná zodpovedajúce otvorené texty:

$$\{E_K(p_1), E_K(p_2), \dots E_K(p_n)\}$$

- KPA – *Útok so znalosťou otvoreného textu:* útočník má k dispozícii množinu dvojíc otvorených a šifrových textov:

$$\{(p_1, E_K(p_1)), (p_2, E_K(p_2)) \dots (p_n, E_K(p_n))\}$$

- CPA – *Útok s možnosťou voľby otvoreného textu:* útočník má možnosť zvoliť si niekoľko otvorených textov, ku ktorým získa zodpovedajúce šifrové texty, pri použití rovnakého kľúča.

- CCA – *Útok s možnosťou voľby šifrového textu:* útočník má možnosť zvoliť si niekoľko šifrových textov, ku ktorým získa zodpovedajúce otvorené texty, pri použití rovnakého kľúča.

Cieľom týchto útokov je získať určitú informáciu, ktorá znižuje bezpečnosť celej šifry. Napríklad pri útoku COA touto informáciou môžu byť príslušné otvorené texty. Pri útoku KPA to môže byť schopnosť z ľubovoľného otvoreného textu p zostrojiť $E_K(p)$. Hlavným cieľom je samozrejme získať samotný kľúč, ktorý bol pri šifrovaní použitý.

[2] T. j. pozná šifrovaciu a dešifrovaciu transformáciu, generátor keystreamu, pozná všetky pravidlá potrebné pre šifrovanie a dešifrovanie.

Útoky sú zoradené podľa rastúcej sily. Útočník má najľahšiu situáciu pri type útoku s možnosťou voľby šifrového textu.

Základným cieľom šifry je zaistenie utajenia správ tak, aby útočník nemohol zo šifrového textu získať otvorený text a to na základe akéhokoľvek reálneho útoku, napríklad podľa základných štyroch typov útokov. Najspoľahlivejšia miera bezpečnosti je založená na informačno-teoretickom prístupe.

Šifrovací systém nazveme *výpočtovo bezpečný*, ak na jeho prelomenie potrebujeme určitú výpočtovú silu. Presnejšie, ak najrýchlejší (všetky) algoritmus na prelomenie šifry vyžaduje vykonať najmenej N operácií. Bohužiaľ o žiadnom doposiaľ známom šifrovacom systéme sa nepodarilo dokázať, že je výpočtovo bezpečný, podľa tejto definície. Pre praktické použitie túto definíciu zoslabíme a šifrovací systém nazveme výpočtovo bezpečný, ak *doteraz* najrýchlejší známy algoritmus na prelomenie šifrovacieho systému vyžaduje najmenej N operácií.

Jedným zo spôsobov dôkazu výpočtovej bezpečnosti je redukcia bezpečnosti šifry na nejaký obtiažny matematický problém, ktorý je všeobecne považovaný za nedosažiteľný. Šifrovacie systémy tohoto typu nazývame *preukázateľne bezpečné* (provable secure). Myšlienka tohoto prístupu spočíva v demonštrácii, že výpočtová nedosažiteľnosť problému P_1 implikuje výpočtovú nedosažiteľnosť problému P_2 (ekvivalentne výpočtová dosažiteľnosť problému P_2 implikuje výpočtovú dosažiteľnosť problému P_1) a to nasledujúcim spôsobom.

Problém P_1 je nejaký obtiažny matematický problém, ktorý je všeobecne považovaný za nedosažiteľný, napríklad problém faktorizácie celých čísel alebo problém diskrétneho logaritmu. Problém P_2 predstavuje určitý typ útoku voči uvažovanému kryptosystému. Cieľom je ukázať, že ak existuje efektívny algoritmus na riešenie problému P_2, tak tento algoritmus dokážeme efektívnym spôsobom využiť na vyriešenie problému P_1. V tomto prípade hovoríme, že problém P_1 sa redukuje na problém P_2. Typicky sa postupuje sporom. T. j. nech existuje efektívny algoritmus (útočník), schopný realizovať určitý typ útoku na daný kryptosystém, potom sa tento algoritmus dá efektívne využiť na vyriešenie matematického problému, čo je v spore s jeho predpokladanou výpočtovou nedosažiteľnosťou. Preto nemôže existovať algoritmus, ktorý dokáže úspešne, s pravdepodobnosťou, ktorá nie je zanedbateľná, realizovať uvažovaný typ útoku voči danému kryptosystému.

Šifrovací systém je *absolútne bezpečný* (perfect secrecy), ak je bezpečný i v prípade, že útočník má k dispozícii neobmedzené výpočtové prostriedky. Útočník nedokáže získať zo zašifrovaného textu **žiadnu informáciu** o pôvodnom texte. Pojem zaviedol C.E Shannon a ukázal, že Vernamova šifra (One-time pad) je absolútne bezpečná.

Vernamova šifra pracuje nasledovne. Postupne sčítava znaky otvoreného textu a hesla. Preto dĺžka hesla sa musí rovnať dĺžke otvoreného textu. *Heslo je vyberané náhodne, nezávisle a nikdy sa nepoužíva opakovane.* Na jednoduchom príklade 2.3 si ukážeme použitie Vernamovej šifry.

Príklad 2.3. Nech otvorený text je správa `THISCRYPTOSYSTEM` a heslo, ktoré bolo vybrané náhodne je `GALFOVMMETOXDKST`. Písmenám priradíme poradové číslo v abe-

cede a otvorený text postupne sčítame s heslom modulo počet písmen v abecede, v anglickej abecede je to 26. Z čoho dostaneme

19	7	8	18	2	17	24	15	19	14	18	24	18	19	4	12
6	0	11	5	14	21	12	12	4	19	14	23	3	10	18	19
25	7	19	23	16	12	10	1	23	7	6	21	21	3	22	5

kde v prvom riadku je pôvodná správa, v druhom riadku je heslo a posledný riadok je sčítanie hodnôt nad sebou modulo 26. Nakoniec prevedieme reťazec čísel späť do znakov abecedy a obdržíme túto zašifrovanú správu `ZHTXQNKBXHGVVDWF`. Vidíme, že celý šifrovací postup spočíva v posune každého znaku správy o náhodne zvolený počet miest v abecede.

Napriek bezpečnostným kvalitám Vernamovej šifry, má táto šifra aj dôležitý praktický nedostatok. Ak by sme chceli preniesť n znakovú správu, je potrebné vygenerovať a bezpečne doručiť adresátovi kľúč o dĺžke n. Týmto sa z problému dôverného prenosu správy stáva problém dôverného prenosu kľúča rovnakej dĺžky.

2.3 Prúdová šifra ako aproximácia Vernamovej šifry

Na myšlienke absolútnej bezpečnosti Vernamovej šifry funguje práve prúdová šifra. Prúdovú šifru, ktorej šifrovacie a dešifrovacie funkcie sú sčítanie a odčítanie, môžeme chápať, ako snahu o aproximáciu Vernamovej šifry a jej bezpečnostných vlastností, ktorá zároveň zmierňuje jej hlavný nedostatok - dĺžku kľúča.

Predpokladajme, že Alica a Bob sa dohodli na použití konkrétnej prúdovej šifry, keďže uvažujeme prípad, kedy šifrovacie a dešifrovacie funkcie sú sčítanie a odčítanie, stačí aby si dohodli akým spôsobom budú generovať samotný keystream. Tajne si vymenia krátky kľúč a obaja si pomocou kľúča a dohodnutého generátora vygenerujú rovnaký keystream, ktorý sa ďalej použije ako heslo u Vermanovej šifry.

Čím viac sa keystream „podobá“ náhodne vybranej postupnosti (heslu), tým viac sa bezpečnostné vlastnosti prúdovej šifry približujú vlastnostiam absolútne bezpečnej Vernamovej šifry. Vidíme, že bezpečnosť celej prúdovej šifry závisí len od bezpečnostných vlastností použitého generátora keystreamu, ktorý by mal generovať postupnosť, ktorá sa podobá náhodne vybranej postupnosti.

2.4 Generátor pseudonáhodných čísel

V tejto časti zadefinujeme pojmy a vlastnosti, ktoré sme trochu neformálne vysvetlili v predošlom oddiele. Definujeme vlastnosť *„podobať sa“* náhodnej postupnosti a definujeme generátor pseudonáhodných čísel (PRNG).

Generátor pseudonáhodných čísel je algoritmus na generovanie postupnosti čísel, ktorá sa približuje vlastnosti postupnosti náhodných čísel. Vstupnými dátami pre pseudonáhodné generátory je náhodná, ale krátka, postupnosť znakov nazývaná semiačko

(seed), ktorá jednoznačne určuje ďalší beh generátora. Výstupom je dlhšia postupnosť znakov, ktorá sa „podobá“ náhodne vybranej postupnosti. Bezpečný generátor pseudonáhodných čísel generuje postupnosť, ktorá je nerozlíšiteľná od náhodnej postupnosti.

Definícia 2.4. *Dve pravdepodobnostné rozdelenia X a Y nad konečnou množinou Ω nazveme výpočtovo nerozlíšiteľné v čase T s výhodou ε, ak pre každý algoritmus $\mathbf{A}$, ktorý pracuje v čase lepšiom ako T platí*

$$|\Pr_{x\in X}(\mathbf{A}(x)=1)-\Pr_{x\in Y}(\mathbf{A}(x)=1)|<\varepsilon,$$

kde $\varepsilon>0$.
A naopak, ak existuje algoritmus, pre ktorý platí

$$|\Pr_{x\in X}(\mathbf{A}(x)=1)-\Pr_{x\in Y}(\mathbf{A}(x)=1)|\geq\varepsilon,$$

potom rozdelenia X a Y sú výpočtovo rozlíšiteľné v čase T s výhodou ε.

Definícia 2.5 (PRNG)**.** *Polynomiálny algoritmus $\mathbf{G}:K^n\to K^L$, kde K je konečná číselná množina a $L\geq n$, nazývame generátor pseudonáhodných čísel, ak platí, že pravdepodobnostné rozdelenie $\mathbf{G}(U_n)$ a U_L, kde U_n a U_L značia rovnomerné pravdepodobnostné rozdelenie priestorov K^n a K^L, sú výpočtovo nerozlíšiteľné v čase T s výhodou ε. Hodnoty T a $\varepsilon>0$ sú bezpečnostné parametre pseudonáhodného generátora $\mathbf{G}$ a sú volené tak, aby vyhovovali konkrétnym bezpečnostným požiadavkám.*

Namiesto toho, že pravdepodobnostné rozdelenie $\mathbf{G}(U_n)$ a U_L sú výpočtovo (ne)rozlíšiteľné v čase T s výhodou ε, budeme hovoriť, že (ne)existuje algoritmus $\mathbf{A}$, ktorý rozlíši postupnosť vygenerovanú generátorom $\mathbf{G}$ od náhodne vybranej postupnosti v čase T s výhodou

$$\mathbf{Adv}_G^{\mathrm{prng}}(\mathbf{A})=|\Pr_{\boldsymbol{x}\in U_n}(\mathbf{A}(G(\boldsymbol{x}))=1)-\Pr_{\boldsymbol{x}\in U_L}(\mathbf{A}(\boldsymbol{x})=1)|=\varepsilon.$$

Ďalej pre generátor pseudonáhodných čísel G a čas T definujeme hodnotu

$$\mathbf{Adv}_g^{\mathrm{prng}}(T)=\max_{\mathbf{A}}\left\{\mathbf{Adv}_g^{\mathrm{prng}}(\mathbf{A})\right\},$$

maximum je brané cez všetky algoritmy $\mathbf{A}$, ktoré pracujú v čase maximálne T.

Algoritmus $\mathbf{A}$, ktorý pracuje v čase T rozlišuje r hodnôt $(G(\boldsymbol{x_1}),\ldots,G(\boldsymbol{x_r}))$ od r náhodne vybraných postupností z K^L s výhodou

$$\mathbf{Adv}_g^{\mathrm{prng}}(\mathbf{A})=|\Pr(\mathbf{A}(G(\boldsymbol{x_1}),\ldots,G(\boldsymbol{x_r}))=1)-\Pr(\mathbf{A}(\boldsymbol{y_1},\ldots,\boldsymbol{y_r})=1)|,$$

kde $\boldsymbol{x_i}\in_R K^n$ a $\boldsymbol{y_i}\in_R K^L$. Analogicky $\mathbf{Adv}_g^{\mathrm{prng}}(T,r)=\max_{\mathbf{A}}\left\{\mathbf{Adv}_g^{\mathrm{prng}}(\mathbf{A})\right\}$, kde maximum berieme cez všetky algoritmy $\mathbf{A}$, ktoré používajú r vstupov a pracujú v čase maximálne T.

Lema 2.6. $\mathbf{Adv}_g^{prng}(T_1) \leq \mathbf{Adv}_g^{prng}(T_2)$, *pre* $T_1 \leq T_2$.

Dôkaz. Tvrdenie priamo vyplýva z definície 2.5. □

Definícia 2.7 (Bezpečný PRNG)**.** *Generátor pseudonáhodných čísel G budeme nazývať bezpečným PRNG, ak hodnota $\mathbf{Adv}_g^{prng}(t)$ je zanedbateľná (napríklad $< 1/100$) pre každé $t < T$, kde T je pevne daná hodnota (napríklad 2^{80} alebo 2^{128}). Definícia bezpečného PRNG je teda závislá na voľbe hodnoty T, ktorá odrážať konkrétny požiadavok na bezpečnosť PRNG.*

Na ujasnenie predošlých definícii uvedieme na záver tohoto oddielu jednoduchý príklad na ktorom názorne ukážeme význam týchto definícii. Zostrojíme algoritmus $\mathbf{A}$, ktorý v polynomiálnom čase dokáže s pravdepodobnosťou ε rozlíšiť výstup z jednoduchého generátora pseudonáhodných čísel $\mathbf{G}$ od náhodne vybranej postupnosti.

Príklad 2.8. Nech $\mathbf{G} : \{0,1\}^n \to \{0,1\}^{2n}$ je PRNG, ktorý generuje postupnosť núl a jedničiek o dĺžke $2n$, ktorá obsahuje presne n núl a n jedničiek. Definujme algoritmus $\mathbf{A}$ takto,

$$\mathbf{A}(z_1, \ldots, z_{2n}) = \begin{cases} 1 & \text{ak } (z_1, \ldots, z_{2n}) \text{ obsahuje presne } n \text{ bitov rovných } 0 \\ 0 & \text{inak} \end{cases}$$

Pre takto definovaný algoritmus $\mathbf{A}$ platí, že ak jeho vstupom bude výstup z generátora $\mathbf{G}(\boldsymbol{x})$, teda postupnosť obsahujúca presne n bitov rovných nule, tak algoritmus $\mathbf{A}$ vždy správne odpovie „áno, postupnosť bola vygenerovaná generátorom $\mathbf{G}$“, kde túto odpoveď reprezentujeme číslom 1

$$\mathbf{A}(\mathbf{G}(\boldsymbol{x})) = 1, \text{ kde } \boldsymbol{x} \in_R \{0,1\}^n .$$

Pričom $\boldsymbol{x} \in_R X$ značí, že $\boldsymbol{x}$ je vybrané z množin X náhodne s rovnomerným rozdelením.
A naopak, ak vstupom algoritmu $\mathbf{A}$ bude náhodne vybraná postupnosť dĺžky $2n$, tak s pravdepodobnosťou $\binom{2n}{n}/2^n$ bude táto postupnosť obsahovať presne n bitov rovných nule a algoritmus $\mathbf{A}$ túto postupnosť nesprávne označí ako postupnosť vygenerovanú generátorom $\mathbf{G}$.

$$\text{Pre } \boldsymbol{y} \in_R \{0,1\}^{2n} \text{ platí, že } \mathbf{A}(y) = 1 \text{ s pravdepodobnosťou } \binom{2n}{n}/2^n.$$

Z čoho dostávame, že algoritmus $\mathbf{A}$ dokáže rozlíšiť výstup z generátora $\mathbf{G}$ od náhodne vybranej postupnosti s výhodou

$$\mathbf{Adv}_G^{\text{prng}}(\mathbf{A}) = \left|\Pr_{\boldsymbol{x} \in U_n}(\mathbf{A}(\mathbf{G}(\boldsymbol{x})) = 1) - \Pr_{\boldsymbol{y} \in U_{2n}}(\mathbf{A}(\boldsymbol{y}) = 1)\right| = 1 - \frac{\binom{2n}{n}}{2^n},$$

ktorá sa pre rastúce n blíži k 1.

Algoritmus $\mathbf{A}$ pracuje v lineárnom čase $O(n)$. Pre ľubovoľnú zafixovanú hodnotu $\varepsilon < 1$ a dostatočne veľké n , rozlíši postupnosť $(z_1, \ldots, z_{2n})$ vygenerovanú generátorom $\mathbf{G}$ od náhodne vybranej postupnosti dĺžky $2n$, v čase $T = O(n)$ s výhodou ε. A teda takýto generátor pseudonáhodných čísel by sme určite nenazvali bezpečným PRNG.

2.5 Prúdová šifra s inicializačným vektorom IV

Takmer všetky moderné prúdové šifry používajú na generovanie keystreamu dva vstupy: tajný kľúč K a prídavný parameter nazývaný inicializačný vektor IV, ktorý vo väčšine prípadov nie je tajný. Inicializačný vektor IV nám umožnuje vygenerovať niekoľko na sebe nezávyslých keystreamov pomocou resynchonizácie šifry pre každý nový IV.

$$\underbrace{\mathbf{G}(K, \mathtt{IV}_1),\ \mathbf{G}(K, \mathtt{IV}_2),\ \ldots,\ \mathbf{G}(K, \mathtt{IV}_m)}_{\text{postupnosť dĺžky } mL}.$$

Použitie IV poskytuje vhodnú metódu pre zašifrovanie niekoľkých správ pomocou jedného tajného kľúča K. Čo je značná praktická výhoda oproti prúdovej šifre bez IV, kde jediným vstupom je tajný kľúč. Použitie inicializačného vektora má samozrejme vplyv na bezpečnosť celej prúdovej šifry a celkovo nám sťažuje formalizovanie bezpečnostných požiadavok pre prúdové šifry používajúce inicializačný vektor. Ak útočník má k dispozícií rôzne keystreamy vygenerované z rovnakého kľúča, ale rôznych inicializačných vektorov má samozrejme väčšiu šancu z nich odvodiť užitočné informácie, ktoré by v prípade znalosti jedného keystreamu nemohli byť odvodené. Za posledné roky bolo publikované veľké množstvo útokov na prúdové šifry s IV a preto pri formalizovaní bezpečnostných požiadaviek musíme byť veľmi opatrní.

Podrobnejšie sa budeme prúdovej šifre s inicializačným vektorom IV a požiadavkám na jej bezpečnosť venovať v časti 6.2. Kde na konkrétnom na príklade prúdovej šifry Quad objasníme, prečo požadujeme od bezpečnej prúdovej šifry s IV, aby množina funkcií $\mathcal{G} = \{G(K, \cdot);\ K \in \mathcal{K}\}$ bola množinou pseudonáhodných funkcií (PRF).

2.6 Generátor pseudonáhodných funkcií

Uvažujme funkciu $F : \mathcal{K} \times D \to R$, kde $\mathcal{K}$ je konečná množina kľúčov a D, R sú konečné neprázdne množiny. Funkcia F nám z hodnoty $K \in \mathcal{K}$ a $X \in D$ vráti hodnotu $Y \in R$, ktorú budeme ďalej značiť $F(K, X)$.

Pre ľubovoľné $K \in \mathcal{K}$ definujeme zobrazenie $F_K : D \to R$ tak, že $F_K(X) = F(K, X)$. Funkcia F nám takto definuje množinu funkcií

$$\mathcal{F} = \{F_K;\ K \in \mathcal{K}\}.$$

Označme $\mathcal{H} = \{f;\ f : D \to R\}$ množinu všetkých funkcií z D do R. Keďže D a R sú konečné množiny, sú množiny $\mathcal{H}$ a $\mathcal{F}$ taktiež konečné. Ak pre náhodne vybrané $K \in_R \mathcal{K}$ je funkcia F_K výpočtovo nerozlíšiteľná od náhodnej funkcie, t. j. náhodne vybranej funkcie z množiny $\mathcal{H}$, tak zobrazenie F parametrizované podľa K nazveme generátorom pseudonáhodných funkcií a množinu $\mathcal{F}$ množinou pseudonáhodných funkcií. Pre lepšiu predstavu o tom, čo znamená rozlíšiť funkciu vybranú z $\mathcal{F}$ od náhodnej funkcie, predstavme si nasledujúcu hru.

Vyberieme jednu z množín $\mathcal{F}$ alebo $\mathcal{H}$ a následne z tejto množiny náhodne vyberie funkciu, ktorú označíme f. Úlohou hráča je len pomocou r orakulovských prístupov

k funkcii[3] f v čase maximálne T uhádnuť a to aspoň s určitou pravdepodobnosťou ε úspechu, či funkcia f bola vybraná z množiny $\mathcal{F}$ alebo $\mathcal{H}$.

Ak neexistuje hráč, ktorý túto úlohu dokáže splniť v čase T pomocou r dotazov, tak množina $\mathcal{F}$ je PRF s bezpečnostným stupňom T, r a ε.

Definícia 2.9 (PRF). *Nech funkcia $F : \mathcal{K} \times D \to R$ je polynomiálne vypočítateľná, kde $\mathcal{K}$ je konečná množina kľúčov a D, R sú konečné neprázdne množiny. Označme množinu funkcií $\mathcal{F} = \{F_K;\ K \in \mathcal{K}\}$ a množinu všetkých funkcií z D do R označíme $\mathcal{H}$. Ak pre každý algoritmus $\mathbf{A}$, ktorý má možnosť položiť r orakulovských dotazov a pracuje v čase maximálne T, platí že, algoritmus $\mathbf{A}$ rozlišuje funkciu $F_K \in_R \mathcal{F}$ od náhodnej funkcie (náhodne vybranej funkcie z $\mathcal{H}$) s výhodou*

$$\begin{aligned}\mathbf{Adv}_F^{prf}(\mathbf{A}) &= |\Pr_{F_K \in_R \mathcal{F}}(\mathbf{A}(F_K) = 1) - \Pr_{f \in_R \mathcal{H}}(\mathbf{A}(f) = 1)| = \\ &= |\Pr_{K \in_R \mathcal{K}}(\mathbf{A}(F_K) = 1) - \Pr_{f \in_R \mathcal{H}}(\mathbf{A}(f) = 1)| < \varepsilon,\end{aligned}$$

potom funkciu F nazveme generátorom pseudonáhodných funkcií a $\mathcal{F}\{F_K\}$ množinou pseudonáhodných funkcií s bezpečnostným stupňom T, r a ε.

Hodnoty T a $\varepsilon > 0$ a počet dotazov r vyjadrujú „kvalitu“, t. j. bezpečnosť daného PRF a sú volené tak, aby vyhovovali konkrétnym bezpečnostným požiadavkám. Funkcia F nie je tajná. Každý dokáže k z hodnôt K a X vypočítať hodnotu $F(K, X)$.

Pre funckiu F, r dotazov a čas T definujeme hodnotu

$$\mathbf{Adv}_F^{\mathrm{prf}}(T, r) = \max_{\mathbf{A}} \left\{ \mathbf{Adv}_F^{\mathrm{prf}}(\mathbf{A}) \right\},$$

maximum je cez všetky algoritmy pracujúce v čase T, ktoré môžu položiť r orakulovských dotazov danej funkcii.

Definícia 2.10 (Bezpečná PRF). *Množinu funkcií $\mathcal{F} = \{F_K;\ K \in \mathcal{K}\}$ nazveme bezpečnou PRF, ak hodnota $\mathbf{Adv}_F^{prf}(t, r)$ je zanedbateľná pre každé $t < T$ a $r < R$, kde T a R sú pevne dané hodnoty (napríklad $T = 2^{80}$ až 2^{128} a $R = 2^{40}$).*

2.7 Bezpečnosť prúdovej šifry

V prípade prúdových šifier, ktoré v tejto práci uvažujeme, kedy je šifrovacia a dešifrovacia funkcia sčítanie a odčítanie v danom telese, závisí bezpečnosť celej prúdovej šifry len na bezpečnosti použitého generátora keystreamu. Cieľom bezpečnej prúdovej šifry je nájsť bezpečný generátor, ktorý produkuje keystream nerozlíšiteľný od náhodnej postupnosti. V závere tejto kapitoly prehľadne zhrnieme požiadavky, ktoré sa požadujú od bezpečnej prúdovej šifry.

[3] Zariadenie (čierna skrinka), ktorému môžeme dávať dotazy v podobe hodnôt $X \in D$ a výstupom z orakula bude hodnota $f(X)$.

Formálne požiadavky pre prúdovú šifru bez `IV`: V tomto prípade sa na generátor kystreamu pozeráme ako na zobrazenie G z konečnej množiny kľúčov $\mathcal{K}$ do množiny keystreamov.

$$G(K) = \text{keystream}$$

Prúdová šifra je považovaná za bezpečnú, ak generátor G je bezpečný PRNG. Inač povedané keystream vyprodukovaný z náhodne vybraného kľúča K je výpočtovo nerozlíšiteľný v čase T s výhodou ε od náhodne vybranej postupnosti.

Formálne požiadavky pre prúdovú šifru s `IV`: Generátor keystreamu G je v tomto prípade funkcia dvoch premenných, ktorá z dvojice (K, IV) vygeneruje keystream.

$$G(K, IV) = \text{keystream}.$$

Pre ľubovoľné $K \in \mathcal{K}$ definujme zobrazenie G_K, ako $G_K(\texttt{IV}) = G(K, \texttt{IV})$. Funkcia G nám teda definuje množinu funkcií $\mathcal{G} = \{G_K;\ K \in \mathcal{K}\}$. Bezpečnostným požiadavkom na prúdovú šifru s `IV` je, aby $\mathcal{G}$ bola bezpečná PRF.

Všetky definície použité v tejto práci sa týkajú konkrétneho (nie asymptotického) bezpečnostného modelu. Dôvod je ten, že v tejto práci nebudeme chcieť ukázať len redukcia bezpečnosti šifry QUAD na MQ problém. T. j. ukázať, že ak by existoval efektívny (polynomiálny) algoritmus, ktorý rozlíši keystream vygenerovaný šifrou QUAD od náhodnej postupnosti, tak tento algoritmus dokážeme efektívnym spôsobom využiť na vyriešenie MQ problému, čo je v spore s jeho predpokladanou výpočtovou nedosažiteľnosťou.

V tejto práci určíme konkrétne parametre šifry QUAD, kedy môžeme zaručiť bezpečnosť šifry a to pre konkrétny bezpečnostný požiadavok. T. j. ukážeme, že ak existuje algoritmus (útočník) **A**, ktorý v čase $T = 2^{80}$ dokáže roslíšiť keystream vygenerovaný šifrou QUAD (pre konkrétne parametre) od náhodnej postupnosti s výhodou $\varepsilon \geq 0,01$, tak existuje algoritmus **B**, ktorý v čase T_B vyrieši MQ problém (konkrétny získaný po redukcii). Pričom $T_B < T_{\min}$, kde $T_{\min}$ je čas najrýchlejšieho algoritmu (doteraz známeho) na vyriešenie MQ problému, čo považujeme za spor. A teda predpoklad o existencii algoritmu **A** bol nesprávny.

3 MQ problém

V tejto kapitole zadefinujeme MQ problém, ktorý spočíva v nájdení jedneho riešenia kvadratického systému n premenných nad konečným telesom. Ukážeme, že tento problém je $\mathcal{NP}$ úplný a predstavíme základné algoritmy na riešenie MQ problému, konkrétne linearizáciu, relinearizáciu a algoritmus XL.

3.1 Definícia MQ problému

Kvadratickým polynómom n premenných nad telesom $\mathrm{GF}(q)$ budeme rozumieť polynóm stupňa nie viac ako 2 v telese $\mathrm{GF}(q)[x_1, \ldots, x_n]$

$$Q(x) = \sum_{1 \le i \le j \le n} \alpha_{i,j} x_i x_j + \sum_{1 \le n} \beta_i x_i + \gamma,$$

kde $\alpha_{i,j}, \beta_i, \gamma \in \mathrm{GF}(q)$. Množina $\mathcal{Q}$ všetkých kvadratických polynómov n premenných je N-dimenzionálny vektorový priestor nad $\mathrm{GF}(q)$. Báza tohoto vektorového priestoru je daná $N-1$ rôznymi monomickými polynómami stupňa 1 alebo 2 a konštantným polynómom. V telese $\mathrm{GF}(2)$ platí $x_i x_i = x_i$ preto je

$$\begin{aligned} N &= \binom{n}{2} + 1 = \frac{n(n-1)}{2} + 1, \qquad \text{pre } q = 2 \\ N &= \binom{n}{1} + \binom{n}{2} + 1 = \frac{n(n+1)}{2} + 1, \qquad \text{pre } q \neq 2. \end{aligned}$$

Kvadratickým systémom $\boldsymbol{S} = (Q_1, \ldots, Q_m)$ n premenných nad telesom $\mathrm{GF}(q)$, rozumieme systém m kvadratických polynómov n premenných nad telesom $\mathrm{GF}(q)$. Takýto kvadratický systém môžeme reprezentovať mN-ticou z $\mathrm{GF}(q)$.

Definícia 3.1 (MQ problém)**.** *Nech $\boldsymbol{S} = (Q_1, \ldots, Q_m)$ značí kvadratický systém n premenných nad telesom $\mathrm{GF}(q)$.*

Rozhodovací MQ problém: *Existuje vektor $\boldsymbol{x} \in \mathrm{GF}(q)^n$, pre ktorý by platilo $\boldsymbol{S}(\boldsymbol{x}) = \boldsymbol{0}$. T. j. aby $Q_i(\boldsymbol{x}) = 0$ pre všetky $1 \le i \le n$?*

Vyhľadávací MQ problém: *Nájdi vektor $\boldsymbol{x} \in \mathrm{GF}(q)^n$, pre ktorý by platilo $\boldsymbol{S}(\boldsymbol{x}) = \boldsymbol{0}$. T. j. aby $Q_i(\boldsymbol{x}) = 0$ pre všetky $1 \le i \le n$, ak také $\boldsymbol{x}$ existuje, inak odpovedz „$\boldsymbol{S}$ nemá riešenie“.*

Obe verzie, vyhľadávací a rozhodovací MQ problém sú zložitostne ekvivalentné:

Predpokladajme, že máme algoritmus **A**, ktorý dokáže riešiť rozhodovací MQ problém nad telesom $\mathrm{GF}(2)$ v čase T. Potom existuje algoritmus **B**, ktorý dokáže nájsť riešenie daného kvadratického systému v čase menej ako $(n+1)T$. Algoritmus **B** funguje nasledovne

```
if A(S(x_1, ..., x_n)) = 'neriešiteľný' then
  B = 'S nemá riešenie'
else
  for i := 1 to n do
    if A(S(u_1, ... u_{i-1}, 1, x_{i+1}, ..., x_n)) = 'riešiteľný' then
      u_i := 1
    else
      u_i := 0
    end if
  end for
  print (u_1, ... u_n) je riešením systému S
end if
```

Pre ľubovoľné konečné teleso GF(q) bude časová zložitosť algoritmu **B** nie viac ako $T + (q-1)nT$.
A naopak ak dokážeme riešiť vyhľadávací MQ problém, tak samozrejme dokážeme riešiť aj rozhodovací MQ problém, dokonca v rovnakom čase.

3.2 MQ je $\mathcal{NP}$ úplný problém

Definícia 3.2. *Zložitostná trieda $\mathcal{P}$ je množina všetkých rozhodovacích problémov, ktoré sa dajú riešiť na deterministickom Turingovom stroji v polynomiálnom čase. Táto trieda zodpovedá intuitívnej predstave problémov, ktoré vieme efektívne riešiť.*

Definícia 3.3. *Zložitostná trieda $\mathcal{NP}$ je množina rozhodovacích problémov, ktoré sa dajú riešiť na nedeterministických Turingových strojoch v polynomiálnom čase. Ekvivaletne, $\mathcal{NP}$ je trieda rozhodovacích problémov takých, že problém $X \in \mathcal{NP}$, práve keď existuje relácia $R_P \in \mathcal{P}$, taká že*

$$x \in X \Leftrightarrow \exists\, y : |y| \leq |x|^k,\ k \geq 1\ \wedge\ (x, y) \in R_P.$$

Reláciu R_P nazývame svedeckou reláciou problému X. Ak $(x, y) \in R_P$, tak y nazývame svedkom pre x.

Definícia 3.4 (Polynomiálna redukcia)**.** *Rozhodovací problém P_1 sa polynomiálne redukuje na rozhodovací problém P_2, ak existuje polynomiálne vyčisliteľná funkcia f taká, že $\forall x$ platí*

$$x \in P_1 \Leftrightarrow f(x) \in P_2.$$

Definícia 3.5. *Rozhodovací problém P nazveme $\mathcal{NP}$ ťažkým, ak každý rozhodovací problém $P' \in \mathcal{NP}$ sa polynomiálne redukuje na problém P.*

Definícia 3.6. *Rozhodovací problém P nazveme $\mathcal{NP}$ úplný, ak*

1. *$P \in \mathcal{NP}$*
2. *P je $\mathcal{NP}$ ťažký problém.*

Vidíme, že predchádzajúce definície sa týkajú len rozhodovacích problémov. Pre prípad MQ problému sme ukázali, že rozhodovací a príslušný vyhľadávací problém sú zložitostne ekvivalené. To implikuje, že ak rozhodovací MQ problém patrí do triedy $\mathcal{P}$, tak príslušný vyhľadávací MQ problém je riešiteľný v polynómialnom čase. (Nehovoríme, že vyhľadavajúci problém patrí do triedy $\mathcal{P}$.)

Veta 3.7. *Rozhodovací MQ problém nad ľubovoľným konečným telesom* GF(q) *patrí do triedy $\mathcal{NP}$ úplných problémov.*

Dôkaz. Podľa definície (3.6) potrebujeme ukázať, že MQ problém patrí do triedy $\mathcal{NP}$ a zároveň je $\mathcal{NP}$ ťažký. Pretože overiť, že konkrétná hodnota je riešením daného systému, je možné dosadením previesť v polynomiálnom čase, platí, že MQ $\in \mathcal{NP}$. Ďalej ukážeme, polynomiálnú transformáciu známeho $\mathcal{NP}$ úplného problému 3SAT na MQ problém, čím dokážeme, druhú podmienku z definície (3.6).

Ľubovoľnú logickú formulu 3SAT (CNF forma) s m klauzulami a n premenných $x_1, \dots x_n$ v tvare

$$C = \bigwedge_{i=1}^{m} (l_{i1} \vee l_{i2} \vee l_{i3}), \quad \text{kde } l_{ij} \text{ je literal rovný } x_k \text{ alebo } \neg x_k, \text{ pre nejaké } 1 \leq k \leq n,$$

prevedieme do systému kvadratických rovníc nad telesom GF(2) nasledovne.
Pre každu klauzulu $(l_{i1} \vee l_{i2} \vee l_{i3})$ zostrojíme nasledujúce 3 rovnice:

$$\begin{aligned} c_i &= l_{i1} + l_{i2} + l_{i3} \\ d_i &= l_{i1}l_{i2} + l_{i1}l_{i3} + l_{i2}l_{i3} \\ 1 &= c_i + d_i + c_i d_i, \end{aligned}$$

kde

$$\begin{aligned} l_{ij} &= x_k && \text{ak } l_{ij} \text{ je pozitívny literál;} \\ l_{ij} &= (1 + x_k) && \text{ak } l_{ij} \text{ je negatívny literál.} \end{aligned}$$

Posledná z troch rovníc, platí v prípade, že $c_i = 1$ alebo $d_i = 1$. Z prvej rovnice dostávame, že $c_i = 1$, práve vtedy, ak jeden alebo tri literály sú splnené. Z druhej rovnice dostávame, že $d_i = 1$, práve vtedy, ak dva alebo tri literály sú splnené. Čo znamená, že sústava má riešenie, práve vtedy a len vtedy ak aspoň jeden literál je splňený. Preto riešenie tejto sústavy kvadratických rovníc, bude tiež riešením logickej formule C.
Celú formulu sme previedli na systém kvadratických rovníc nad telesom GF(2), kde počet rovníc sa rovná $3m$ a počet premenných $n+2m$. Jedná sa jednoznačne o prevod v lineárnom čase vzhľadom k počtu klauzúl a premenných. Nárast dĺžky polynómu je lineárny (je napísaný v lineárnom čase) a teda naozaj máme polynomiálnu redukciu problému 3SAT na problém riešenia sústavy kvadratických rovníc nad GF(2).

Čím sme ukázali, že MQ problém nad telesom GF(2) patrí do triedy $\mathcal{NP}$ úplných problémov. Pretože teleso GF(2) je podtelesom telesa GF(2^n), tak existencia polynomiálneho algoritmu nad GF(2^n), by nám dáva existenciu polynomiálneho algoritmu nad GF(2). A preto platí, že MQ problém nad telesom GF(2^n) patrí do triedy $\mathcal{NP}$ úplných problémov.

Pre prípad konečného telesa GF(p), kde p je prvočíslo rôzne od 2, nastáva problém, že premenné x_k môžu nadobúdať aj iné hodnoty ako 0 alebo 1. Pridaním rovnice

$$x_k(1 - x_k) = 0, \text{ kde } 1 \leq k \leq n \text{ (pre každú premennú)},$$

zaručíme, že premenné môžu nadobúdat len hodnoty 0 alebo 1. Celú formulu prevedieme na systém kvadratických rovníc nad telesom GF(p), kde počet rovníc sa rovná $3m + n$ a počet premenných $n + 2m$. Je to stále, prevod v lineárnom čase vzhľadom k počtu klauzúl a premenných. Pretože GF(p) je podtelesom telesaGF(p^n) platí, že MQ problém nad telesom GF(p^n) patrí do triedy $\mathcal{NP}$ úplných problémov. □

4 Algoritmy pre riešenie polynomiálnych systémov

Cieľ týchto algoritmov je nájsť jedno riešenie m kvadratických (polynomiálnych) rovníc n premenných nad konečným telesom GF(q). Z dôvodu, že MQ problém patrí do triedy $\mathcal{NP}$ - úplných problémov, nájdenie efektívneho (polynomialného vzhľadom k veľkosti systému) algoritmu, ktorý by riešil všeobecný MQ problém je nemožné, ak $\mathcal{NP} \neq \mathcal{P}$.

Zložitosť konkrétneho MQ problému závisí od príslušných parametroch (počet rovníc, počet neznámych a veľkosť telesa). Dôvod prečo sa MQ problém obzvlášť dobre hodí pre kryptografické aplikácie je predpoklad, že MQ problém je ťažký nie len asymptoticky a to v najhoršom prípade, ale už pre malé vhodne vybrané hodnoty m a n je tento problém ťažký a to z hľadiska priemernej zložitosti riešenia náhodnej instancie $\boldsymbol{S}$. Problém sa zdá byť najťažší pre $m \approx n$. Pre $m = n$ a teleso GF(2) je MQ problém, doteraz známymi algoritmami, riešiteľný so zložitosťou nie lepšou ako $2^{n-O(\sqrt{n})}$, čo je pre $n \geq 100$ mimo dosah momentálnej schopnosti počítačov.

Systém kvadratických rovníc, kde počet neznámych n je väčší ako počet rovníc m nazveme *pod-definovaný*. Na kryptografickej konferencii Eurocrypt 1999 autori článku [17] ukázali, že pre $n \geq m(m+1)$ a konečne teleso GF(2^s), t. j. teleso charakteristiky 2, je tento MQ problém polynomiálne riešiteľný.

Systém kvadratických rovníc, kde počet rovníc m je väčší ako počet neznámych n nazveme *pre-definovaný*. Klasický algoritmus na riešenie pre-definovaných systémov je Buchbergerov algoritmus na nájdenie Groebnerovej bázy s lexikografickým usporiadaním. Avšak zložitosť tohoto algoritmu je príliš veľká. Už pre $n \geq 15$ je tento algoritmus pre praktické použitie nepoužiteľný. V najhoršom prípade má Buchberge-

rov algoritmus dvojito exponenciálnu zložitosť. Priemerná zložitosť je exponenciálna.

U náhodného systému kvadratických rovníc, kde $m > n$, sa neočakáva žiadne riešenie. Ak tento systém zvolíme tak, že riešenie existuje, tak s veľkou pravdepodobnosťou to bude jediné riešenie tohoto systému. Inač povedané, ak sa na systém $\boldsymbol{S} = (Q_1, \ldots, Q_m)$, kde Q_i je polynóm n premenných nad telesom $\mathrm{GF}(q)$, pozeráme ako na zobrazenie z $\mathrm{GF}(q)^n$ do $\mathrm{GF}(q)^m$, tak zobrazenie $\boldsymbol{S}$ je pre $m \geq n$ s pravdepodobnosťou blízkej 1 prostým zobrazením.

Bezpečnosť šifry QUAD je založená práve na probléme riešenia pre-definovaných kvadratických systémov nad konečným telesom $\mathrm{GF}(q)$. Z tohoto dôvodu sa algoritmom na riešenie pre-definovaných systémov budeme venovať podrobnejšie. Predstavíme základné algoritmy na riešenie pre-definovaných systémov rovníc.

4.1 Linearizácia

Základným a dobre známym algoritmuom pre riešenie kvadratických polynomiálnych systémov sa nazýva *linearizácia*. Uvažujme systém m kvadratických rovníc n premenných nad konečným telesom $\mathrm{GF}(q)$. Potom tento systém môžeme riešiť pomocou nasledujúceho algoritmu:

Algoritmus 1 Linearizácia

1. Každý kvadratický monóm $x_i x_j$ nahradíme novou nezávislou premennou z_{ij}, čím získame systém lineárnych rovníc.
2. Riešime systém lineárnych rovníc (napríklad pomocou Gaussovej eliminačnej metódy).
3. Riešenie lineárneho systému, dosadíme do pôvodného kvadratického systému a skontrolujeme správnosť riešenia.

Príklad 4.1. Riešme nasledujúci systém kvadratických rovníc nad $\mathrm{GF}(2)$, pomocou linearizácie:

$$\begin{aligned} xy + y &= 0 \\ xy + y + x &= 1 \\ xy &= 1 \end{aligned}$$

V prvok kroku nahradíme monóm xy novou premennou z a získame nasledujúci systém lineárnych rovníc:

$$\begin{aligned} z + y &= 0 \\ z + y + x &= 1 \\ z &= 1 \end{aligned}$$

Riešením lineárneho systému pomocou Gaussovej metódy, krok 2, získame nasledujúce riešenie:

$$x = 1, \qquad y = 1, \qquad z = 1.$$

A nakoniec, dosadením do pôvodného kvadratického systému, overíme správnosť riešenia.

$$1 \cdot 1 + 1 = 0, \qquad 1 \cdot 1 + 1 + 1 = 1, \qquad 1 \cdot 1 = 1.$$

Z definície algoritmu vyplýva, že ak existuje riešenie pôvodného kvadratického systému, potom toto riešenie je aj riešením systému získaného jeho *linearizáciou.* Žiaľ mnoho riešení z lineárneho systému nie sú riešením pôvodného kvadratického systému. Takéto riešenia nazveme *falošné riešenia lineárneho systému.*

Lineárny systém nad konečným telesom GF(q) nemá riešenie alebo má $q^{n'-r}$ riešení, kde n' je počet neznámych a r je hodnosť množiny rovníc. Za predpokladu, že kvadratický systém má riešenie nad konečným telesom GF(q), vieme že lineárny systém získaný linearizíciou má presne $q^{n'-r}$ riešení. Pre prípad, kedy je $r' \approx n$ je počet riešení lineárného systému (kandidátov na riešenie kvadratického systému) malý.

Pretože počet možných nekonštantných monómov kvadratického systému nad konečným telesom GF(q) je podla kapitoly (3)

$$\begin{aligned} N &= \binom{n}{2} = \frac{n(n-1)}{2}, \qquad \text{pre } q = 2 \\ N &= \binom{n}{1} + \binom{n}{2} = \frac{n(n+1)}{2}, \qquad \text{pre } q \neq 2, \end{aligned}$$

získame linearizáciou systém m kvadratických rovníc n premenných, systém m lineárnych rovníc o N premenných.

Príklad. Uvažujme náhodne vybraný systém 1000 kvadratických rovníc 100 premenných nad konečným telesom GF(2), zvolený tak, že má aspoň jedno riešenie (s veľkou pravdepodobnosťou to bude jediné riešenie tohoto systému). Po linearizácii tohoto systému získame systém 1000 lineárnych rovníc o 5050 neznámych. Takýto lineárny systém má minimálne 2^{4500} riešení. Nájsť jedno riešenie pôvodného kvadratického systému medzi 2^{4050} by trvalo dlhšie ako vyskúšať všetkých 2^{100} možných riešení pôvodného kvadratického systému.

Z predchádzajúceho príkladu vidíme, že linearizáciu má zmysel použít na riešenie pre-definovaných kvadratických systémov m kvadratických rovníc n premenných, kde $m \geq n(n+1)/2$. Čím zaručíme, že získame po linearizácii získame s veľkou pravdepodobnosťou, lineárny systém len s niekoľkými riešeniami.

V opačnom prípade, je množina riešení lineárneho systému príliš početná a obsahuje veľký počet falošných riešení lineárneho systému a len jedno respektíve niekoľko riešení pôvodného systému. Napríklad, linearizáciou systému $m = n^2/2$ kvadratických

rovníc n premenných nad GF(5), získamé lineárny systém s viac ako $5^{n/2}$ riešení.

4.2 Relinearizácia

Kipnis a Shamir v článku [19] predstavili metódu na riešenie systému εn^2 kvadratických rovníc n premenných nazývaný *relinearizácia*. Základnou myšlienkou relinearizácie je pridať k danému systému lineárnych rovníc v y_{ij} premenných ďalšie nelineárne rovnice, ktoré vyjadrujú fakt, že premenné y_{ij} sú na sebe závislé. Čím sa zníži počet tzv. falošných riešení lineárneho systému.

Uvažujme systém εn^2 kvadratických rovníc n premenných nad telesom GF(q), kde $0 < \varepsilon < 1/2$. Tento systém prepíšeme na systém εn^2 lineárnych rovníc $n^2/2$ premenných $y_{ij} = x_i x_j$ pre $i \leq j$. Množina riešení tohoto systému je vektorový priestor s dimenziou $\approx n^2/2 - \varepsilon n^2 = (1/2 - \varepsilon)n^2$. A teda každé y_{ij} môžeme vyjadriť ako lineárnu kombináciu $(1/2 - \varepsilon)n^2$ parametrov, ktoré označíme z_k, pre $1 \leq k \leq (1/2 - \varepsilon)n^2$. K týmto lineárnym rovniciam pridáme nové nelineárne rovnice, ktoré vyjadrujú fakt, že premenné y_{ij} sú na sebe závislé. A to nasledujúcim spôsobom.

Uvažujme štvoricu indexov (a, b, c, d) spĺňajúcu $1 \leq a \leq b \leq c \leq d \leq n$, potom $x_a x_b x_c x_d$ môže byť ozátvorkované 3 nasledujúcimi spôsobmi

$$(x_a x_b)(x_c x_d) = (x_a x_c)(x_b x_d) = (x_a x_d)(x_b x_c) \implies y_{ab} y_{cd} = y_{ac} y_{bd} = y_{ad} y_{bc}.$$

Lema 4.2. *Počet rovníc (obmedzení pre z_k) v tvare $(x_a x_c)(x_b x_d) = (x_a x_d)(x_b x_c)$, pre $1 \leq a \leq b \leq c \leq d \leq n$ vytvorené pomocou relinearizácií je*

$$2\binom{n}{4} + 3\binom{n}{3} + \binom{n}{2} = \frac{1}{12}\left(n^4 - n^2\right).$$

Dôkaz. Počet spôsobov ako vybrať 4 rôzne indexy a, b, c, d je rovný $\binom{n}{4}$. Každá takáto štvorica nám vygeneruje 2 lineárne nezávislé rovnice

$$(x_a x_b)(x_c x_d) = (x_a x_c)(x_b x_d) \text{ a } (x_a x_c)(x_b x_d) = (x_a x_d)(x_b x_c).$$

Podobne, počet spôsobov, ako vybrať štvoricu indexov v ktorej sú práve 3 indexy rôzne, je $3\binom{n}{3}$. Každá takáto štvorica nám vygeneruje jednu rovnicu

$$(x_a x_b)(x_b x_d) = (x_a x_d)(x_b x_b).$$

A nakoniec počet spôsobov, ako vybrať štvoricu indexov v ktorej sú práve 2 indexy rôzne je $\binom{n}{2}$. Každá takáto štvorica nám vygeneruje jednu rovnicu

$$(x_a x_b)(x_a x_b) = (x_a x_a)(x_b x_b).$$

□

Z predošlej lemy dostávame, že po relinearizácií získame približne $\frac{n^4}{12}$ kvadratických rovníc o $(1/2-\varepsilon)n^2$ premenných z_k. Následne tento nový kvadratický systém linearizujeme. Čím získame $n^4/12$ lineárnych rovníc o $((1/2-\varepsilon)n^2)^2/2$ premenných. Pre hodnoty $m^4/12 \geq ((1/2-\varepsilon)n^2)^2/2$ sa očakáva, že tento lineárny systém bude jednoznačne riešiteľný. A to je splnené pre $\varepsilon \geq 1/2 - 1/\sqrt{6} \approx 0,1$, čo je približne 1/5 rovníc požadovaná pri linearizácii.

Existuje mnoho variant a optimalizácii tejto základnej techniky relinearizácie. Hodnotu ε môžeme znížiť napríklad, použitím relinearizácie rekurzívne alebo pridaním ďalších nelineárnych rovníc, ktoré vystihujú závislosť premenných pred linearizáciou. Čo je možne urobiť, tak že namiesto štvoríc budeme uvažovať $n^6/6!$ šestíc $x_a x_b x_c x_d x_e x_f$, ktoré nám dávaju 14 rôznych rovníc stupňa 3. Následne tento systém linearizujeme a každú premennú $z_i z_j z_k$, pre $i \leq j \leq k$ označíme ako novú nezávislú premennú. Takto dostaneme približne $14n^6/720$ lineárnych rovníc a $((1/2-\varepsilon)n^2)^3/6$ premenných. Z čoho dostávame, že pre $\varepsilon \geq 0,008$ je tento systém jednoznačne riešiteľný. Presná zložitosť relinearizácie zatiaľ nie je známa. Očakáva sa, že pre $m \geq \varepsilon n^2$ je táto zložitosť polynomiálna, približne $n^{O(1/\sqrt{\varepsilon})}$.

V článku [12] sa autori venujú praktickej analýze relinearizácie. Na veľkom počte pokusov pre rôzne hodnoty m a n analyzovali, ktorý systém je pomocou relinearizácie riešiteľný. Ukázali, že mnohé z rovníc vygenerovaných pri relinearizácií sú lineárne závislé. A preto sa zdá, že efektívnosť tejto metódy pri praktickom použití je o trochu menšia, ako sa očakávalo. Techniku relinearizácie si predvedieme na nasledujúcom príklade.

Príklad. Uvažujme systém 5 náhodne vybraných kvadratických rovníc 3 premenných x_1, x_2, x_3 nad konečným telesom GF(7).

$$\begin{aligned}
3x_1x_1 + 5x_1x_2 + 5x_1x_3 + 2x_2x_2 + 6x_2x_3 + 4x_3x_3 &= 5\\
6x_1x_1 + 1x_1x_2 + 4x_1x_3 + 4x_2x_2 + 5x_2x_3 + 1x_3x_3 &= 6\\
5x_1x_1 + 2x_1x_2 + 6x_1x_3 + 2x_2x_2 + 3x_2x_3 + 2x_3x_3 &= 5\\
2x_1x_1 + 0x_1x_2 + 1x_1x_3 + 6x_2x_2 + 5x_2x_3 + 5x_3x_3 &= 0\\
4x_1x_1 + 6x_1x_2 + 2x_1x_3 + 5x_2x_2 + 1x_2x_3 + 4x_3x_3 &= 0
\end{aligned}$$

Po linearizácií, nahradením y_{ij} za $x_i x_j$ dostávame systém 5 lineárnych rovníc o 6 premenných. Gaussovou eliminačnou metódou získame parametrizované riešenie

$$y_{11} = 2 + 5z,\ y_{12} = z,\ y_{13} = 3 + 2z,\ y_{22} = 6 + 4z,\ y_{23} = 6 + z,\ y_{33} = 5 + 3z.$$

Tento systém má 7 rôznych riešení. Na odfiltrovanie tzv. falošných riešení lineárneho systému použijeme nasledujúce rovnice:

$$y_{11}y_{23} = y_{12}y_{13}, y_{12}y_{23} = y_{13}y_{22}, y_{12}y_{33} = y_{13}y_{23}.$$

Dosadením parametrického vyjadrenia pre y_{ij}, dostaneme

$$(2+5z)(6+z) = z(3+2z),\ z(6+z) = (3+2z)(6+4z),\ z(5+3z) = (3+2z)(6+z).$$

Po úprave dostávame nasledujúci kvadratický systém

$$\begin{aligned} 3z^2 + 1z + 5 &= 0 \\ 0z^2 + 4z + 4 &= 0 \\ 1z^2 + 4z + 3 &= 0 \end{aligned}$$

Nahradením (linearizáciou) $z_1 = z^2$ a $z_2 = z$ dostávame lineárny systém 3 rovníc o 2 premenných z_1, z_2. Gaussovou elimináciou dostávame $z_1 = 6$ a $z_2 = 1$. Spätným dosadením a výpočtom dostávame dve riešenia pôvodného kvadratického systému $(2, 3, 4)$ a $(5, 4, 3)$.

4.3 Algoritmus XL

Ďalšou metódou pre riešenie kvadratických rovníc je algoritmus XL (eXtended Linearization). Napriek jednoduchosti algoritmu XL sa momentálne zdá, že algoritmus XL je najrýchlejším algoritmom pre riešenie náhodne vybraných pre-definovaných kvadratických systémov. Hlavná myšlienka algoritmu je zvýšiť počet rovníc, pridaním nových rovníc.

Na algoritmus XL sa môžeme pozerať ako na vylepšenú verziu relinearizácie. Autori článku [12] dokázali, že ak *relinearizácia* uspeje pri riešení m kvadratických rovníc n neznámych, tak uspeje aj algoritmus XL. Budeme používať nasledujúce značenia. Stupeň monómu ${x_1}^{b_1}{x_2}^{b_2}\cdots{x_n}^{b_n} = \boldsymbol{x}^{\boldsymbol{b}}$, označíme $|\boldsymbol{b}| = \sum b_i$. Množinu všetkých monómov stupňa $\leq D$ označíme $\mathcal{T}^D = \mathcal{T}$. Počet prvkov množiny $\mathcal{T}^D$ označíme $T = T^D = \left|\mathcal{T}^D\right|$.

Algoritmus XL funguje pre pre-definovaný systém, ktorý má riešenie. V prípade kedy je $m < n$ je potrebné uhádnuť aspoň $n - m$ premenných a tak získať systém, ktorý má aspoň toľko premenných ako rovníc.

Definícia 4.3 (Popis algoritmu XL stupňa D)**.** *Nech $l_1(\boldsymbol{x}), l_2(\boldsymbol{x}), \ldots l_m(\boldsymbol{x})$ sú kvadratické polynómy n premenných nad telesom* $\mathrm{GF}(q)$*. Predpokladajme, že existuje riešenie tohoto systému, t. j. $\boldsymbol{x_0}$, také že $l_i(\boldsymbol{x_0})$ pre $1 \leq i \leq m$. Algoritmus XL pracuje nasledujúcim spôsobom:*

1. *(e**X**tend): Pre každý monóm stupňa $\leq D-2$, $\boldsymbol{x}^{\boldsymbol{b}} \in \mathcal{T}^{D-2}$, vygenerujeme množinu rovníc $\boldsymbol{x}^{\boldsymbol{b}} l_i(\boldsymbol{x}) \in \mathcal{T}^D$, kde $1 \leq i \leq m$. Systém všetkých týchto rovníc, označíme $\mathcal{R} = \mathcal{R}^D$.*

2. *(**L**inearize): Systém rovníc $\mathcal{R}$ pozostáva z $R = mT^{D-2}$ rovníc stupňa D. Pomocou linearizácie, kde každý monóm $\boldsymbol{x}^{\boldsymbol{b}} \in \mathcal{T}^D$ označíme ako novu premennú, získame lineárny systém pozostávajúci z R lineárnych rovníc a T^D neznámych. Následne na systém $\mathcal{R}$ prevedieme (Gaussovú) eliminačú metódu. Poradie monómov musí byť také, aby monómy obsahujúce jednu premennú boli eliminované na konci.*

3. *Predpokladajme, že z predchádzajúceho kroku sme získali jednu rovnicu o jednej neznámej, napríklad* $a_0 + a_1x_1 + \cdots + a_k{x_1}^D = 0$. *Túto rovnicu nad konečným telesom* $\mathrm{GF}(q)$ *vyriešime. Následne spätným chodom v (Gaussovej) eliminácii postupne získavame hodnoty ďalších premenných.*

Poznámka. Pred začiatkom algoritmu XL musíme rozhodnúť o hodnote D. Ak pre konkrétne D algoritmus neuspeje, skúsime XL stupňa na $D + 1$.

Poznámka 4.4. Algoritmus XL sa dá analogicky použiť pre systém polynomiálnych rovníc ľubovoľného stupňa d. V prvom kroku by sme akurát generovali systém pomocou monómov z množiny T^{D-d}.

Je zrejmé, že pre väčšie D algoritmus XL vygeneruje väčší počet rovníc, čím sa zvýši šanca, že po linearizácií a následnej eliminácií týchto rovníc, získame rovnicu jednej premennej a teda algoritmus uspeje. Na stranu druhú, väčšie D má negatívny dopad na zložitosť algoritmu. Jednoducho by sme mohli začat s najmenším možným $D = 3$ a v prípade, že algoritmus XL neuspeje, hodnotu D zvýšime o jedničku a algoritmus spustíme znovu. Čo samozrejme nie je najefektivnejší spôsob. Preto sa pokusíme hodnotu najmenšieho D, pre ktorý algoritmus XL uspeje odhadnúť.

4.3.1 Analýza algoritmu XL

Systém $\mathcal{R}$ je homogénny systém lineárnych rovníc, kde neznáme sú monómy z množiny $\mathcal{T}^D$. Očakáva sa, že táto sústava lineárnych rovníc má riešenie, ak $R \geq T$ (počet rovníc je väčši ako počet neznámych).

Lema 4.5 (Počet monómov stupňa $\leq D$)**.** *Označme* $[u]p$ *koeficient pri výraze* u *v rozvoji* p. *Napríklad* $[x^2](1-x)^4 = 6$. *Potom*

$$T = T^{(D)} = [t^D]\frac{(1-t^q)^n}{(1-t)^{n+1}}. \tag{4.1}$$

Dôkaz. Uvažujme súčin $\prod_{i=1}^{n}(1 + x_i + {x_i}^2 + \cdots + {x_i}^{q-1}) = \prod_{i=1}^{n}[(1-x_i^q)/(1-x_i)]$. Takýto súčin nám vygeneruje všetky možné monómy (každý práve raz). Ak v tomto súčine položíme každé x_i rovno t, tak koeficient pri člene t^D nám dáva počet monómov stupňa D. Pretože výraz

$$\left(a_0 + a_1t + a_2t^2 + \cdots + a_Dt^D + \cdots\right)\left(1 + t + t^2 + \cdots t^D + \cdots\right)$$

má koeficient pri člene t^D rovný $(a_0 + a_1 + \cdots a_D)\,t^D$, dostávame počet všetkých monómov do stupňa D je

$$T = T^{(D)} = [t^D]\prod_{i=1}^{n}[(1-t^q)/(1-t)]\,\frac{1}{1-t} = [t^D]\frac{(1-t^q)^n}{(1-t)^{n+1}}.$$

□

Poznámka 4.6. Pre $D > q$ je $T = \sum_{j=0}^{D} \binom{n+j-1}{j}$ (počet kombinácií s opakovaním). Pre $q = 2$ dostávame $T = \sum_{j=0}^{D} \binom{n}{j}$.

Pre zjedonušenie výpočtu dolného odhadu D, nutnej podmienky pre fungovanie algoritmu XL, t. j. kedy $R \geq T$ použijeme nasledujúci odhad

$$\binom{n}{k} \approx \frac{n^k}{k!}. \tag{4.2}$$

Uvažujme, sústavu kvadratických rovníc nad GF(2) a predpokladajme, že $D \leq n/2$. Počet rovníc vygenerovaných v 1. kroku algoritmu XL je $R = R^D = mT^{D-2}$. Preto z nerovnosti $R \geq T$ dostávamé

$$m \sum_{j=0}^{D-2} \binom{n}{j} \geq \sum_{j=0}^{D} \binom{n}{j}$$

$$m \geq \frac{\sum_{j=0}^{D} \binom{n}{j}}{\sum_{j=0}^{D-2} \binom{n}{j}} \geq \frac{\binom{n}{D}}{\binom{n}{D-2}}.$$

Použitím (4.2) dostávame

$$m \geq \frac{\frac{n^D}{D!}}{\frac{n^{D-2}}{(D-2)!}} = \frac{n^2}{D(D-1)} \geq \frac{n^2}{D^2}.$$

Odkiaľ dostávame požadovanú nerovnosť

$$D \geq \frac{n}{\sqrt{m}}.$$

Bohužial nie všetky rovnice vygenerované z kroku 1 algoritmu XL sú lineárne nezávislé. Preto k určeniu hodnoty D pred spustením algoritmu XL, potrebujeme odhadnúť hodnotu $I = \dim(\mathrm{span}\mathcal{R})$. T. j. počet lineárne nezávislých rovníc v množine $\mathcal{R}$.

Označme $[p]$ polynóm $p(\boldsymbol{x})$ a nech $l_i(\boldsymbol{x}) = \sum_{j \leq k} a_{ijk} x_j x_k + \sum_j b_{ij} x_j + c_i$, potom

$$\sum_{j \leq k} a_{ijk}[x_j x_k l_{i'}] + \sum_j b_{ij}[x_j l_{i'}] + c_i[l_{i'}] = \sum_{j \leq k} a_{i'jk}[x_j x_k l_i] + \sum_j b_{i'j}[x_j l_i] + c_{i'}[l_i]$$

$$l_i[l_{i'}] = l_{i'}[l_i]$$

Teda $[l_i l_j]$, kde $i \neq j$ sa javia ako dve rozdielne lineárne kombinácie polynómov. Analogicky dostávame $l_i^{q-1}[l_i] = [l_i]$.

Príklad 4.7. Uvažujme systým kvadratických rovníc n premenných nad telesom GF(2) $l_1 = x_1x_3 + x_4, l_2 = x_1x_2 + x_4x_7$. V množine $\mathcal{R}^4$, vygenerovanej pomocou prvého kroku algoritmu XL, sa vyskytujú tieto prirodzene vygenerované závislosti

$$l_1\,[l_2] = [l_1]\,l_2$$

$$l_1x_1x_2 + l_1x_4x_7 = l_2x_1x_3 + l_2x_4,$$

čo je presne $\binom{m}{2}$ lineárnych závislosti. A podobne

$$l_1\,[l_1] = l_1$$
$$l_1 x_1 x_3 + l_1 x_4 = l_1.$$

Čo nám dáva ďalších m lineárnych závislosti v množine $\mathcal{R}^4$.

Takže určitá závislosť medzi závislosťami existuje a dá sa očakávať, že počet lineárne nezávyslých rovníc, vygenerovaných v kroku 1 definície algoritmu XL, môžeme odhadnúť. A to pomocou nasledujúcej vety (4.8). Z dôvodu, že dôkaz nasledujúcej vety je technický náročný a zároveň, pre túto prácu nie je doležitý, uvedieme nasledujúcu vetu bez dôkazu. Dôkaz nájdete v článku [15].

Veta 4.8. *Počet lineárne nezávyslých rovníc v množine $\mathcal{R}^D$ nad konečným telesom* $\mathrm{GF}(q)$ *môžeme odhadnúť nasledujúcim spôsobom*

$$T - I \geq [t^D]G(t) = [t^D]\left(\frac{(1-t^q)^n}{(1-t)^{n+1}}\left(\frac{1-t^2}{1-t^{2q}}\right)^m\right), \quad \textit{pre všetky } D < D_{XL}, \tag{4.3}$$

kde D_{XL} sa nazýva stupeň regularity, definovaný ako najmenšie D, pre ktoré rovnica 4.3 nemôže platiť, pokiaľ má systém riešenie, t. j.

$$D_{XL} = \min\left\{D:\ [t^D]G(t) \leq 0\right\}.$$

Funkcia $G(t)$ sa nazýva Hilbertova rada.

Tvrdenie 4.9. *Nech $l_i(\boldsymbol{x})$ sú kvadratické polynomy a v množine $\mathcal{R}^D$ nie sú žiadne iné závislosti, než tie „prirodzene“, vygenerované z $l_i[l_j] = l_i[l_j]$ a $l_i^{q-1}[l_i] = [l_i]$, potom v* (4.3) *platí rovnosť.*

Definícia 4.10. *Ak pre všetky $D < D_{XL}$ plati, že $\mathcal{R}^D$ neobsahuje žiadne iné závislosti okrem $l_i[l_j] = l_i[l_j]$ a $l_i^{q-1}[l_i] = [l_i]$, potom takýto systém rovníc nazveme* ***semiregulárny****.*

Parameter D algoritmu XL, ktorý vyberáme na začiatku, sa snažíme vybrať, tak, aby algoritmus XL fungoval a aby jeho zložitosť bola čo najmenšia. Takže hodnota D by mala byť čo najmenšia, ale dostatočne veľká, aby zaručovala, že $T-I \leq \min(D, q-1)$. Čo nám zaručuje, že po linearizácií (2. krok) získame jednu rovnicu o jednej neznámej,

$$a_0 + a_1 x + \cdots + a_{\min(D,q-1)} x^{\min(D,q-1)} = 0.$$

Nech D_0 značí najmenšie D, pre ktoré XL funguje.

Predpokladá sa [21], že náhodne vybraný kvadratický systém je s veľkou pravdepodobnostou semiregulárny. Výsledky testov napríklad z článku [2] „platnosť“ tohoto predpokladu potvrdzujú. Z tvrdenia (4.9) dostávame, že

$$D_0 = \min\left\{D : [t^D]G(t) \leq \min(D, q-1)\right\},$$

a s veľkou pravdepodobnosťou platí, že $D_{XL} = D_0$.

Poznámka 4.11. Parameter $D_0 = D_{XL}$, pre náhodne vybraný kvadratický systém nad končným telesom GF(q), je stupeň prvého nekladného koeficientu v Hilberovej rade

$$G(t) = \left(\frac{(1-t^q)^n}{(1-t)^{n+1}} \left(\frac{1-t^2}{1-t^{2q}} \right)^m \right),$$

čo vyplýva priamo z definície D_{XL}.

Po linearizácii systému $\mathcal{R}^D$ získame riedku maticu (počet nenulových prvkov je oveľa nižší, ako počet ich nulových prvkov). Klasická Gaussová eliminačná metóda nepracuje rýchlejšie na riedkých maticiach. *Metóda Wiedemann*, využívajúca existenciu Krylovho podpriestoru, je prispôsobená práve pre nájdenie riešenia riedkeho systému rovníc. Vyžaduje menej pamäte a pracuje rýchlejšie ako Gaussová metóda. Pretože tématika riešenia lineárnych rovníc je veľmi rozsiahla a v tejto práci nás zaujíma skôr riešenie nelineárnych systémov, nebudeme sa touto tématikou vôbec zaoberať a tvrdenia (zložitosť algoritmu Wiedemann) prevezme bez dôkazu. Popis algoritmu Wiedemann je možné nájsť napríklad v [2].

Tvrdenie 4.12. *Očakávaná časová zložitosť algoritmu XL(Wiedemann) je*

$$C_{XL} = E_W(R,T) \approx 3\tau T\,\mathfrak{m}, \text{ jedno násobenie } \mathfrak{m} \approx (c_0 + c_1 \lg T) \text{ CPU cyklov.}$$

kde $\tau = kT$ je celkový (k priemerný $\left(= \binom{n+2}{2}\right)$) počet členov v rovniciach sústavy $\mathcal{R}$ a c_i sú konštanty závisle na kokrétnej architektúre CPU.

Na záver tohoto oddielu na jednoduchom príklade ukážeme fungovanie algoritmu XL.

Príklad 4.13. Uvažujme náhodne vybraný kvadratický systém $(p_1(\boldsymbol{x}), p_2(\boldsymbol{x}))$ 2 premenných nad konečným telesom GF(5). Tento systém upravíme tak, aby jeho riešením bol vektor $(1,0)$. T. j. $l_i(\boldsymbol{x}) = p_i(\boldsymbol{x}) - p_i((1,0))$

$$\begin{aligned} l_1: &\quad y^2 + 2xy + 4x + 1 &= 0 \\ l_2: &\quad 2x^2 + y + 3 &= 0. \end{aligned}$$

Na začiatku si zvolíme parameter $D = 4$. V 1. kroku obidve rovnice $l_i(\boldsymbol{x}) = 0$ prenásobime s každým monómom z $\mathcal{T}^2 = \{1, x, y, x^2, y^2, xy\}$. Takto získame systém $\mathcal{R} = \left\{\boldsymbol{x}^{\boldsymbol{b}} l_1, \boldsymbol{x}^{\boldsymbol{b}} l_2\right\}$, kde $\boldsymbol{x}^{\boldsymbol{b}} \in \mathcal{T}^2$. T. j. systém $2 \times 6 = 12$ rovníc stupňa 4.

$\boldsymbol{x}^{\boldsymbol{b}} l_1$:

$$\begin{aligned} y^2 + 2xy + 4x + 1 &= 0 \\ xy^2 + 2x^2y + 4x^2 + x &= 0 \\ y^3 + 2xy^2 + 4xy + y &= 0 \\ x^2y^2 + 2x^3y + 4x^3 + x^2 &= 0 \\ y^4 + 2xy^3 + 4xy^2 + y^2 &= 0 \\ xy^3 + 2x^2y^2 + 4x^2y + xy &= 0 \end{aligned}$$

$\boldsymbol{x}^{\boldsymbol{b}} l_2$:

$$\begin{aligned} 2x^2 + y + 3 &= 0 \\ 2x^3 + xy + 3x &= 0 \\ 2x^2y + y^2 + 3y &= 0 \\ 2x^4 + x^2y + 3x^2 &= 0 \\ 2x^2y^2 + y^3 + 3y^2 &= 0 \\ 2x^3y + xy^2 + 3xy &= 0 \end{aligned}$$

Prirodzene vygenerovaná závislosť v množine $\mathcal{R}$ je

$$\begin{aligned} l_1[l_2] &= 2(2x^3y + xy^2 + 3xy) + 1(2x^2y^2 + y^3 + 3y^2) + 4(2x^3 + xy + 3x) + \\ &+ 1(2x^2 + y + 3) = 2(x^2y^2 + 2x^3y + 4x^3 + x^2) + 1(y^3 + 2xy^2 + 4xy + y) + \\ &+ 3(y^2 + 2xy + 4x + 1) = l_2[l_1] \end{aligned}$$

A teda napríklad rovnica $(2x^3y + xy^2 + 3xy = 0)$ je lineárnou kombináciou ostatných rovníc a preto ju môžeme v systéme rovníc vynechať. Lineárne závislosti typu l_i^{q-1} sa pre prípad nevyskytuje (D¡q).Preto $I = \dim(\mathrm{span}\mathcal{R}) \leq 11$. Aby bol algoritmus XL úspešný musí byť počet lineárne nezávislých rovníc $I \geq 15 - 4$. Preto systéme rovníc $\mathcal{R}$ musí byť už lineárne nezávislí (mimo jednej z rovníc zo závislosti $l_1[l_2] = l_2[l_1]$). Po linearizácii získavame maticu o rozmere 11×15.

$$\mathcal{M} = \begin{pmatrix} x^3y & x^2y^2 & xy^3 & xy^2 & x^2y & xy & y^4 & y^3 & y^2 & y & x^4 & x^3 & x^2 & x & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 2 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 1 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 3 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 & 1 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 4 & 1 \\ 0 & 0 & 0 & 1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 1 & 0 \\ 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 1 & 0 & 0 \\ 0 & 0 & 2 & 4 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 1 & 0 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Po Gaussovej eliminačnej metóde získame maticu.

$$\begin{pmatrix} x^3y & x^2y^2 & xy^3 & xy^2 & x^2y & xy & y^4 & y^3 & y^2 & y & x^4 & x^3 & x^2 & x & 1 \\ 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 1 & 0 & 0 \\ 0 & 2 & 1 & 0 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 1 & 4 & 0 & 1 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 1 & 4 & 3 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 4 & 3 & 1 & 1 & 0 & 0 & 0 & 2 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 & 3 & 1 & 0 & 2 & 0 & 0 & 2 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 & 1 & 0 & 2 & 0 & 2 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 3 & 0 & 4 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 3 & 3 & 4 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 3 & 1 & 4 & 0 \end{pmatrix}$$

Z posledného riadku získavame rovnicu $2x^4 + 3x^3 + x^2 + 4x = 0$ nad GF(5), ktorej jediným riešením je $x = 1$. Dosadením do druhého riadku matice získanej po eliminácií dostávame $y = 0$. Samozrejme v tomto prípade by bolo určite jednoduchšie a

rýchlejšie vyskúšať všetkých 25 možností pre riešenie danej sústavy. Príklad len ilustruje jednotlivé kroky algoritmus XL. A zároveň poukazuje aj na pamäťovú náročnosť algoritmu XL. Preto pri práci s maticami je takmer nevyhnutné využívať teóriu riedkych matíc. K uloženiu riedkej matice je potreba menej pamäte a algoritmy na riešenie riedkych sústav ako metóda združených gradientov (CGM) alebo algoritmy Lanczos a Wiedemann fungujú na riedkych maticiach podstatne rýchlejšie ako klasiké algoritmy na riešenie „hustých“ sústav.

5 Popis šifry QUAD

V tejto kapitole predstavíme prúdovú šifru QUAD. Definujeme jej generátor keystreamu a generátor inicializačnej hodnoty.

Nech $\boldsymbol{S} = (Q_1, \ldots, Q_{k \cdot n})$ značí náhodne vybraný kvadratický systém kn kvadratických polynómov n premenných nad konečným telesom GF(q). Nech $\boldsymbol{S_0}$ a $\boldsymbol{S_1}$ sú náhodne vybrané kvadratické systémy n polynómov n premenných nad telesom GF(q). Pričom $\boldsymbol{S}$, $\boldsymbol{S_1}$ a $\boldsymbol{S_0}$ sú pevne dané a verejne známe.

5.1 Generovanie keystreamu

Označme $\boldsymbol{S_{\text{int}}} = (Q_1, \ldots, Q_n)$ a $\boldsymbol{S_{\text{out}}} = (Q_{n+1}, \ldots, Q_{kn})$. Generovanie keystreamu prebieha pomocou nasledujúceho algoritmu

Algoritmus 2 Generovanie keystreamu z $(\boldsymbol{S}, \boldsymbol{x_0}, \lambda)$

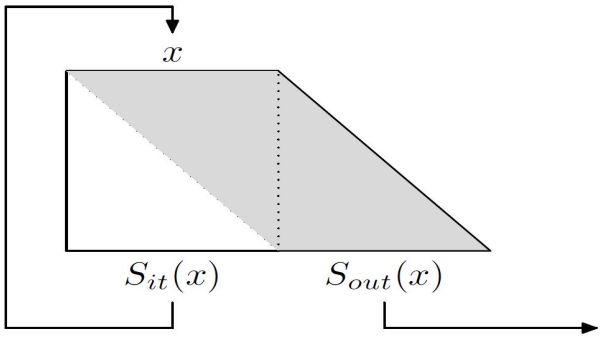
```
x := x_0
for i := 0 to λ do
    S_out(x) = (Q_{n+1}(x), ..., Q_{kn}(x))
    x := (Q_1(x), ..., Q_n(x))
    keystream := keystream || S_out(x)
end for
```

$$\text{KEYSTREAM} = \boldsymbol{S_{\text{out}}}(\boldsymbol{x}) || \underbrace{\boldsymbol{S_{\text{out}}}(\boldsymbol{S_{\text{int}}}(\boldsymbol{x}))}_{n(k-1) \text{ znakov z GF}(q)} || \cdots || \boldsymbol{S_{\text{out}}}(\boldsymbol{S_{\text{int}}}^{\lambda}(\boldsymbol{x}))$$

Princíp generovania keystreamu je jednoduchý. Spočíta sa hodnota

$$\boldsymbol{S}(\boldsymbol{x}) = (Q_1(\boldsymbol{x}), \ldots, Q_{kn}(\boldsymbol{x})),$$

z ktorej posledných $(k-1)n$ hodnôt je výstupom generátora (časťou keystreamu) a prvých n hodnôt sa použije ako nový vstup pre $\boldsymbol{S}$. Celý tento cyklus opakujeme λ krát. Pri každom jednom prechode cyklu sa vyprodukuje $(k-1)n$ hodnôt keystreamu z telesa GF(q). Dĺžku keystreamu vygenerovaná po λ prechodov z jednej dvojice $(\boldsymbol{S}, \boldsymbol{x_0})$ je preto rovná $L = \lambda(k-1)n$. Samotné šifrovanie otvoreného textu $(x_1, \ldots, x_l)$, kde x_i sú hodnoty z GF(q) a $l \leq L$, prebieha postupným sčítavaním keystreamu $(k_1, \ldots k_l)$ a otvoreného textu v telese GF(q).



Obr. 1: Generáto keystreamu šifry QUAD.

5.2 Generovanie inicializačnej hodnoty $\boldsymbol{x_0}$ z K a IV

Predtým než začneme generovať keystream, potrebujeme vygenerovať inicializačnú hodnotu vektor $\boldsymbol{x_0}$ z kľúča $\boldsymbol{K}$ a inicializačného vektora IV. Kde $\boldsymbol{K}$ značí náhodne vybraný vektor z GF(q) dĺžky n a $\mathtt{IV} \in_R \{0,1\}^{|\mathtt{IV}|}$. Generovanie hodnoty $\boldsymbol{x_0}$ prebieha nasledovne.

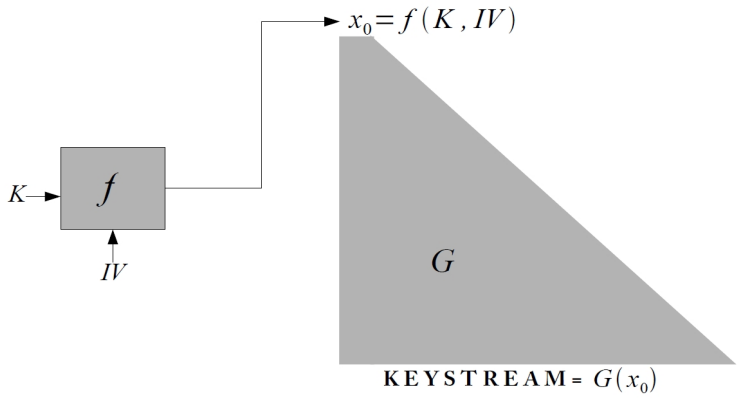
Do $\boldsymbol{x}$ vložíme $\boldsymbol{K}$ a pre každý bit inicializačného vektoru IV hodnotu $\boldsymbol{x}$ v prípade, že príslušný bit je rovný 1 prepíšeme na hodnotu $\boldsymbol{S_1}(\boldsymbol{x})$ v opačnom prípade $\boldsymbol{x}$ prepíšeme na hodnotu $\boldsymbol{S_0}(\boldsymbol{x})$. Nakoniec ešte $|\mathtt{IV}|$ krát prepíšeme hodnotu $\boldsymbol{x}$ na hodnotu $\boldsymbol{S_{\text{int}}}(\boldsymbol{x})$ s cieľom ďalej pretransformovať hodnotu $\boldsymbol{x}$.

Algoritmus 3 Generovanie inicializačného vektora $\boldsymbol{x_0}$ z $(\mathtt{IV}, \boldsymbol{K})$

```
x := K
for i := 0 to |IV| do
  if IV_i = 0 then
    x := S_0(x)
  else
    x := S_1(x)
  end if
end for
for i := 0 to |IV| do
  x := S_int(x)
end for
```

Algoritmus 1 spolu a 2 nám definujú generátor keystreamu prúdovej šifry QUAD generátor (Obr. 2). Tento generátor nám z kľúča $\boldsymbol{K}$ a inicializačného vektora IV vygeneruje keystream $\textsc{quad} : \mathrm{GF}(q)^n \times \{0,1\}^{|\mathtt{IV}|} \longrightarrow \mathrm{GF}(q)^L$

$$\textsc{quad}(\boldsymbol{K}, \mathtt{IV}) = \mathbf{G} \circ f(\boldsymbol{K}, \mathtt{IV}) = \text{keystream}.$$



Obr. 2: Šifra QUAD $= G \circ f$.

Kde f je funkcia definovaná algoritmom 3, ktorá z kľúča $\boldsymbol{K}$ a `IV` vygeneruje inicializačnú hodnotu $\boldsymbol{x_0}$. **G** je samotný generátor keystreamu, definovaný algoritmom 2, ktorý z inicializačnej hodnoty $\boldsymbol{x_0}$ vygeneruje samotný keystream.

Pod pojmom šifra QUAD budeme rozumieť nasledujúce dva prípady

- Zobrazenie $G \circ f$ – generátor keystreamu vrátane generovania inicializačnej hodnoty. (Keystream je vygenerovaný šifrou QUAD z kľúča $\boldsymbol{K}$ a inicializačného vektora `IV`.)
- Zobrazenie G – len samotný generátor keystreamu bez generovania inicializačnej hodnoty. (Keystream je vygenerovaný šifrou QUAD z náhodne vybranej inicializačnej hodnoty $\boldsymbol{x}$.)

Keďže z kontextu bude vždy jasné o ktorý prípad sa jedná, t.j. či pri generovaní uvažujeme aj samotne generovanie inicializačnej hodnoty $\boldsymbol{x_0}$ z kľúča $\boldsymbol{K}$ a `IV`, bolo by zbytočne a zrejme aj mätúce zavádzať pre tieto dva prípady rozdielne značenia.
Šifru QUAD asociovanú s kn kvadratickými polynómami n premenných nad konečným telesom GF(q) budeme značiť QUAD(k, n, q).

Príklad 5.1. Uvedieme jednoduchý príklad šifry QUAD$(2, 3, 2)$. T. j. QUAD asociovaný so 6 kvadratickými polynómami 3 premenných nad telesom GF(2), kde keystream bude vygenerovaný z náhodne vybranej inicializačnej hodnoty $\boldsymbol{x_0}$. Náhodne vygenerujeme kvadratický systém $\boldsymbol{S} = (Q_1, \ldots, Q_6)$ 3 premenných nad telesom GF(2).

$$\begin{aligned} Q_1 &= x_1 + x_3; & Q_2 &= x_1x_3 + x_2x_3 + x_1 + 1; \\ Q_3 &= x_1x_2 + x_3; & Q_4 &= x_1x_2 + x_2 + x_3; \\ Q_5 &= x_1x_3 + x_2x_3 + x_1 + x_2 + 1; & Q_6 &= x_2x_3 + x_3 + 1; \end{aligned}$$

Ďalej náhodne vyberieme inicializačnú hodnotu $\boldsymbol{x_0} = (1, 0, 1)$. Pomocou algoritmu 2 zo vstupných hodnôt $\boldsymbol{S}$, $\boldsymbol{x_0}$ a $\lambda = 6$ dostávame nasledujúce hodnoty

i	$\boldsymbol{x_i}$	$\boldsymbol{S}(\boldsymbol{x_i})$	keystream
0	**(1,0,1)**	(**0,1,1**, 1, 1, 0)	(1, 1, 0)
1	**(0,1,1)**	(**0,0,1**, 0, 1, 1)	(1, 1, 0, 0, 1, 1)
2	**(0,0,1)**	(**1,1,1**, 1, 1, 0)	(1, 1, 0, 0, 1, 1, 1, 1, 0)
3	**(1,1,1)**	(**0,0,0**, 1, 1, 1)	(1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1)
4	**(0,0,0)**	(**0,1,0**, 0, 1, 1)	(1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1)
5	**(0,1,0)**	(**0,1,0**, 1, 0, 1)	(1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1)
6	**(0,1,0)**		

Samotné šifrovanie je už len jednoduché sčítavanie v telese GF(2). Napríklad zašifrovanie správy $(1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1)$, ktorej dĺžka musí byť menšia ako dĺžka keystreamu je

$$\begin{array}{rl} \text{Otvorený text:} & (1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1) \\ \text{Keystream:} & (1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1) \\ \hline \text{Šifrový text:} & (0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0) \end{array}$$

Dešifrovanie je odčítanie (v prípade GF(2) sčítanie) keystreamu od zašifrovanej správy.

$$\begin{array}{rl} \text{Šifrový text:} & (0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0) \\ \text{Keystream:} & (1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1) \\ \hline \text{Otvorený text:} & (1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1) \end{array}$$

6 QUAD: Preukázateľne bezpečná šifra

V tejto kapitole predvedieme redukciu bezpečnosti šifry QUAD na MQ problém. Najprv v podkapitole 6.1 predvedieme redukciu generátora keystreamu, ktorú následne v podkapitole 6.2 rozšírime na celú šifru QUAD vrátane generovania inicializačnej hodnoty. Ukážeme konštrukciu bezpečnej PRF z bezpečného PRNG pomocou binárneho stromu. Na tejto konštrukcii je postavené práve generovanie inicializačnej hodnoty šifry QUAD.

Podľa formálnych požiadavok na bezpečnosť prúdovej šifry sformulovaných v oddiely 2.7. Je našim cieľom je ukázať, že zobrazenie QUAD definované ako zobrazenie

$$\text{QUAD}(\boldsymbol{K}, \texttt{IV}) = \mathbf{G} \circ f(\boldsymbol{K}, \texttt{IV}) = \text{keystream}$$

je bezpečný generátor pseudonáhodných funkcií (parametrizované podľa $\boldsymbol{K}$). Úlohou je teda ukázať, že ak existuje efektívny algoritmus, ktorý rozlíši QUAD (ako PRF) od náhodnej funkcie, tak tento algoritmus dokážeme efektívnym spôsobom využiť na vyriešenie MQ problému. Čo ako ukážeme v podkapitole 7 je pre určité parametre šifry QUAD v spore s predpokladanou výpočtovou nedosažiteľnosťou MQ problému. Redukciu bezpečnosti šifry QUAD na MQ rozdelíme do 3 častí.

1. Najprv v podkapitole 6.1 ukážeme, redukciu bezpečnosti generátora $\mathbf{G}$ keystreamu (PRNG) na riešenie systému kvadratických rovníc. T. j. ak by existoval algoritmus, ktorý rozlíši keystream vygenerovaný generátorom $\mathbf{G}$ šifry QUAD, potom tento algoritmus dokážeme efektívne využiť na riešenie systému kvadratických rovníc.

2. V podkapitole 6.2 ukážeme, ze konštrukcia funkcie f definovaná algoritmom 3 je zhotovená podľa stromovej konštrukcie na zostrojenie bezpečnej PRF z bezpečného PRNG. A opäť ukážeme redukciu bezpečnosti tejto funkcie (PRF) na MQ problém.

3. Na koniec podľa jednoduchej vety o skladaní 6.9 dostávame, že $\mathbf{G} \circ f$ je generátor pseudonáhodných funkcií. A bezpečnosť celej šifry QUAD sa dá zredukovať na MQ problém (riešenie systému kvadratických rovníc na konečným telesom).

6.1 Bezpečnosť generátora keystreamu šifry QUAD

V celej tejto podkapitole budeme pod pojmom šifra QUAD rozumieť práve tento PRNG, ktorého vstupom je náhodne vybraná inicializačná hodnota $\boldsymbol{x_0}$. Generovanie inicializačnej hodnoty z IV a kľúča K v celej tejto časti *neuvažujeme*.

Autori šifrovacieho systému QUAD vo svojom článku, QUAD*: a Practical Stream Cipher with Provable Security* [1], predstavili prúdovú šifru, kde pre zjednodušenie

uvádzajú tvrdenie o bezpečnosti tejto šifry a jeho dôkaz, len pre prípad, kedy je tento keystream generovaný kvadratickým systémom nad telesom GF(2). Toto tvrdenie bolo pre špeciálne polynomiálne systémy rozšírené na ľubovoľné konečné teleso $\mathrm{GF}(q)$ v článku [3], *Secure PRNGs from Specialized Polynomial Maps over Any GF(q).*

V tejto časti dokážeme, že ak by existoval útočník, ktorý dokáže efektívne rozlíšiť keystream vyprodukovaný generátorom šifry QUAD šifrou od náhodne vybranej postupnosti, tak tento útočník dokáže efektívne riešiť MQ problém, čo je v spore s jeho predpokladanou výpočtovou nedosažitelnosťou. Dôkaz môžeme rozdeliť do troch samostatných krokov.

V prvom kroku (veta 6.1) dokážeme, že ak vieme rozlíšiť keystream vygenerovaný šifrou QUAD, tak tak potom dokážeme rozlíšiť výstup z ľubovoľného kvadratického systému $\boldsymbol{S}(\boldsymbol{x})$ od náhodne vybranej postupnosti.

V druhom kroku (veta 6.2) ukážeme, že ak vieme rozlíšiť $\boldsymbol{S}(\boldsymbol{x})$ od náhodne vybranej postupnosti, tak dokážeme z hodnoty $\boldsymbol{S}(\boldsymbol{x})$, kde $\boldsymbol{x}$ je neznáma náhodne vybraná hodnota, vypočítať hodnotu $R(\boldsymbol{x})$ pre ľubovoľnú lineárnej formu R.

V treťom kroku (veta 6.3) ukážeme, že ak vieme z hodnoty $\boldsymbol{S}(\boldsymbol{x})$, kde $\boldsymbol{x}$ je neznáma náhodne vybraná hodnota, vypočítať hodnotu ľubovoľnú lineárnej formu $R(\boldsymbol{x})$, potom dokážeme vypočítať hodnotu $\boldsymbol{x}$ z hodnoty $\boldsymbol{S}(\boldsymbol{x})$. To by znamenalo, že $\boldsymbol{S}$ nie je jednosmerná funkcia a to je pre vhodne zvolené n, m, q v spore s predpokladanou zložitosťou MQ problému.

6.1.1 Z rozlišovača šifry QUAD, prediktor lineárnej formy

Veta 6.1. *Nech $L = \lambda(k-1)n$ značí dĺžku keystreamu vygenerovaného v čase $\lambda T_{\boldsymbol{S}}$, ktorý dostaneme vykonaním λ prechodov konštrukcie šifry* QUAD(k, n, q)*, asociovanej so známym, náhodne vybraným kvadratickým systémom $\boldsymbol{S}$ a neznámou náhodne vybranou inicializačnou hodnotou $\boldsymbol{x} \in_R \mathrm{GF}(q)^n$. Predpokladajme, že existuje algoritmus* **A***, ktorý dokáže rozlíšiť tento keystream od náhodne vybranej postupnosti z* $\mathrm{GF}(q)$ *dĺžky L, v čase T s výhodou ε. Potom existuje algoritmus* **B***, ktorý pre náhodne vybraný kvadratický systém $\boldsymbol{S}$, rozlišuje $\boldsymbol{S}(\boldsymbol{x})$, kde $\boldsymbol{x} \in_R \mathrm{GF}(q)^n$ je neznáma, náhodne vybraná hodnota, od náhodne vybranej postupnosti z* $\mathrm{GF}(q)$ *dĺžky kn a to v čase lepšom ako $T + \lambda T_{\boldsymbol{S}}$ s výhodou ε/λ.*

Dôkaz. Definujme pravdepodobnostné rozdelenia $D^i(\boldsymbol{S})$ na $\mathrm{GF}(q)^L$, kde $D^i(\boldsymbol{S})$ je asociované s hodnotami

$$\boldsymbol{t^i}(\boldsymbol{S}, \boldsymbol{x}) = \left(\boldsymbol{r_1}, \ldots, \boldsymbol{r_i}, \boldsymbol{S_{\mathrm{out}}(x)}, \boldsymbol{S_{\mathrm{out}}(S_{\mathrm{int}}(x))}, \ldots, \boldsymbol{S_{\mathrm{out}}(S_{\mathrm{int}}}^{\lambda-i-1}\boldsymbol{(x))}\right),$$

pre každé $0 \leq i \leq \lambda$, pričom $\boldsymbol{r_j} \in_R \mathrm{GF}(q)^{(k-1)n}$ a $\boldsymbol{x} \in_R \mathrm{GF}(q)^n$ sú nezávislé náhodne vybrané vektory. Pre $i = 0$ definujeme $(r_1, \ldots, r_i) = \emptyset$ a pre $i = \lambda$ definujeme

$$\left(\boldsymbol{S_{\mathrm{out}}(x)}, \ldots, \boldsymbol{S_{\mathrm{out}}(S_{\mathrm{int}}}^{\lambda-i-1}\boldsymbol{(x)}\right) = \emptyset.$$

Takže $D^0(\boldsymbol{S})$ je pravdepodobnostné rozdelenie keystreamu dĺžky L a naopak $D^\lambda(S)$ je rovnomerným rozdelením vektorov na $\mathrm{GF}(q)^L$. Pravdepodobnosť, že **A** prijíme (odpovie 1) náhodnú L znakovú postupnosť s rozdelením $D^i(\boldsymbol{S})$ označíme $p^i(\boldsymbol{S})$ a nech

p^i značí priemernú hodnotu z $p^i(\boldsymbol{S})$ cez všetky kvadratické systémy $\boldsymbol{S}$. Z predpokladu vyplýva, že $|p^0 - p^\lambda| \geq \varepsilon$. Nech algoritmus $\mathbf{B}$ pracuje nasledovne

1. zo vstupu $(\boldsymbol{x_1}, \boldsymbol{x_2}) \in \mathrm{GF}(q)^{kn}$, kde $\boldsymbol{x_1} \in \mathrm{GF}(q)^n$ a $\boldsymbol{x_2} \in \mathrm{GF}(q)^{n(k-1)}$ vytvorí vektor dĺžky L

$$\boldsymbol{t}(\boldsymbol{S}, \boldsymbol{x_1}, \boldsymbol{x_2}) = \left(\boldsymbol{r_1}, \ldots, \boldsymbol{r_i}, \boldsymbol{x_2}, \boldsymbol{S}_{\mathbf{out}}(\boldsymbol{x_1}), \ldots, \boldsymbol{S}_{\mathbf{out}}({\boldsymbol{S}_{\mathbf{int}}}^{\lambda-i-2}(\boldsymbol{x_1}))\right),$$

   kde $i \in_R \{0, \ldots, \lambda - 1\}$;

2. zavolá algoritmus $\mathbf{A}$ so vstupom $(\boldsymbol{S}, \boldsymbol{t}(\boldsymbol{S}, \boldsymbol{x_1}, \boldsymbol{x_2}))$ a vráti hodnotu získanú z výsledku algoritmu $\mathbf{A}$.

V prípade, že $(\boldsymbol{x_1}, \boldsymbol{x_2}) = (\boldsymbol{S}_{\mathbf{int}}(\boldsymbol{x}), \boldsymbol{S}_{\mathbf{out}}(\boldsymbol{x})) = \boldsymbol{S}(\boldsymbol{x})$, kde $\boldsymbol{x} \in_R \mathrm{GF}(q)^n$ je náhodne vybraný vektor platí

$$\boldsymbol{t}(\boldsymbol{S}, \boldsymbol{x_1}, \boldsymbol{x_2}) = \left(\boldsymbol{r_1}, \ldots, \boldsymbol{r_i}, \boldsymbol{S}_{\mathbf{out}}(\boldsymbol{x}), \ldots, \boldsymbol{S}_{\mathbf{out}}({\boldsymbol{S}_{\mathbf{int}}}^{\lambda-i-2}(\boldsymbol{x}))\right).$$

A teda vidíme, že platí $\boldsymbol{t}(\boldsymbol{S}, \boldsymbol{x_1}, \boldsymbol{x_2}) = \boldsymbol{t^i}(\boldsymbol{S}, \boldsymbol{x})$ a preto pravdepodobnostné rozdelenie vektoru $\boldsymbol{t}(\boldsymbol{S}, \boldsymbol{x_1}, \boldsymbol{x_2})$ je $D^i(\boldsymbol{S})$.

V prípade, že vektor $(\boldsymbol{x_1}, \boldsymbol{x_2}) \in_R \mathrm{GF}(q)^{kn}$ je vybraný náhodne, z čoho samozrejme vyplýva že $\boldsymbol{x_1} \in_R \mathrm{GF}(q)^n$ a $\boldsymbol{x_2} \in_R \mathrm{GF}(q)^{n(k-1)}$, platí

$$\boldsymbol{t}(\boldsymbol{S}, \boldsymbol{x_1}, \boldsymbol{x_2}) = \boldsymbol{t^{i+1}}(\boldsymbol{S}, \boldsymbol{x}).$$

Vektor $\boldsymbol{t}(\boldsymbol{S}, \boldsymbol{x_1}, \boldsymbol{x_2})$ má v tomto prípade rozdelenie $D^{i+1}(\boldsymbol{S})$.

Pre algoritmus $\mathbf{B}$ dostávame, že dokáže rozlíšiť $\boldsymbol{S}(\boldsymbol{x})$ od náhodne vybranej kn-bitovej postupnosti s výhodou

$$\left|\Pr_{\boldsymbol{S},\,\boldsymbol{x}}(\mathbf{B}(\boldsymbol{S}, \boldsymbol{S}(\boldsymbol{x})) = 1) - \Pr_{\boldsymbol{S},\,\boldsymbol{x_1},\,\boldsymbol{x_2}}(\mathbf{B}(\boldsymbol{S}, (\boldsymbol{x_1}, \boldsymbol{x_2})) = 1)\right| =$$

$$= \left|\Pr_{\boldsymbol{S},\,\boldsymbol{x}}(\mathbf{A}(\boldsymbol{S}, \boldsymbol{t^i}(\boldsymbol{S}, \boldsymbol{x})) = 1) - \Pr_{\boldsymbol{S},\,\boldsymbol{x}}(\mathbf{A}(\boldsymbol{S}, \boldsymbol{t^{i+1}}(\boldsymbol{S}, \boldsymbol{x})) = 1)\right| =$$

$$= \left|\frac{1}{\lambda}\sum_{i=0}^{\lambda-1} p^i - \frac{1}{\lambda}\sum_{i=1}^{\lambda} p^i\right| = \frac{1}{\lambda}\left|p^0 - p^\lambda\right| \geq \frac{\varepsilon}{\lambda}$$

a to v čase $T + \lambda T_{\boldsymbol{S}}$.

□

Vo vete 6.2 ukážeme, že ak vieme rozlíšiť $\boldsymbol{S}(\boldsymbol{x})$ od náhodne vybranej postupnosti ako je to popísane v predošlej vete, tak z $\boldsymbol{S}(\boldsymbol{x})$ dokážeme určiť hodnotu $R(\boldsymbol{x})$ ľubovoľej lineárnej formy.

Veta 6.2. *Nech* $\mathbf{B}$ *je algoritmus, ktorý v čase* T *a s výhodou* ε *rozlišuje* $\boldsymbol{S}(\boldsymbol{x})$*, kde* $\boldsymbol{S}$ *označuje náhodne vybraný kvadratický systém* kn *polynomiálnych rovníc* n *premenných a* $\boldsymbol{x} \in_R \mathrm{GF}(q)^n$ *je neznámy náhodne vybraný vektor, od náhodne vybranej postupnosti z* $\mathrm{GF}(q)^{kn}$*. Potom existuje algoritmus* $\mathbf{C}$*, ktorý zo vstupu* $(\boldsymbol{S}, R, \boldsymbol{y})$*, kde*

- $\boldsymbol{S}$ *značí náhodne vybraný kvadratický systém* kn *kvadratických rovníc* n *premenných nad konečným telesom* $\mathrm{GF}(q)$*;*
- R *je ľubovoľná lineárna forma z* $\mathrm{GF}(q)^n$ *do* $\mathrm{GF}(q)$*;*
- $\boldsymbol{y} = \boldsymbol{S}(\boldsymbol{x})$*, kde* $\boldsymbol{x} \in_R \mathrm{GF}(q)^n$ *je neznámy náhodne vybraný vektor,*

dokáže vypočítať hodnotu $R(\boldsymbol{x})$ *v čase lepšom ako* $T + 2T_{\boldsymbol{S}}$ *s pravdepodobnosťou úspechu lepšou ako* $\frac{1}{q} + \frac{\varepsilon}{2q}$*.*

Dôkaz. Bez ujmy na všeobecnosti predpokladajme, že $\Pr(\mathbf{B}(\boldsymbol{S}, \boldsymbol{S}(\boldsymbol{x})) = 1)$ je väčšia ako $\Pr(\mathbf{B}(\boldsymbol{S}, \boldsymbol{u}) = 1)$, kde $\boldsymbol{u} \in_R \mathrm{GF}(q)^{kn}$. Definujme algoritmus $\mathbf{B}'$ nasledovne

$$\mathbf{B}'(\boldsymbol{S}, \boldsymbol{w}) = \begin{cases} \mathbf{B}(\boldsymbol{S}, \boldsymbol{w}) & \text{s pravdepodobnosťou } 1/2 \\ 1 - \mathbf{B}(\boldsymbol{S}, \boldsymbol{u}), \boldsymbol{u} \in_R \mathrm{GF}(q)^{kn} & \text{s pravdepodobnosťou } 1/2, \end{cases}$$

Algoritmus $\mathbf{B}'$ pri vstupe $(\boldsymbol{S}, \boldsymbol{w})$, kde $\boldsymbol{w} \in \mathrm{GF}(q)^n$ a $\boldsymbol{S}$ značí kvadratický systém, „hodí mincou“ a podľa výsledku si vyberie jeden z vyššie definovaných výstupov.

Pri vstupe $(\boldsymbol{S}, \boldsymbol{S}(\boldsymbol{x}))$ algoritmus $\mathbf{B}'$ vracia hodnotu 1 s pravdepodobnosťou najmenej $\frac{1}{2} + \frac{\varepsilon}{2}$ a pri vstupe $(\boldsymbol{S}, \boldsymbol{u})$, kde vektor $\boldsymbol{u} \in_R \mathrm{GF}(q)^n$, vráti hodnotu 1 s pravdepodobnosťou $\frac{1}{2}$.

Algoritmus $\mathbf{C}$ so vstupom $\boldsymbol{S} = (Q_1, \ldots, Q_{kn})$, lineárnou formou R a vektorom $\boldsymbol{y} = \boldsymbol{S}(\boldsymbol{x}) = (y_1, \ldots, y_{kn})$ pracuje nasledovne

1. vyberie náhodný vektor $\boldsymbol{a} \in_R \mathrm{GF}(q)^{kn}$, náhodne zvolí hodnotu $b \in_R \mathrm{GF}(q)$ a predpokladá, že $R(\boldsymbol{x}) = b$;
2. spočíta $P_i = Q_i + a_i R$ pre $\forall i \in \{1 \ldots kn\}$ a získa nový kvadratický systém
$$\boldsymbol{S}' = (P_1, \ldots, P_{kn}) = \boldsymbol{S} + \boldsymbol{a}R;$$
3. hodnotu $\mathbf{C}(\boldsymbol{S}, R, \boldsymbol{y})$ určí pomocou algoritmu $\mathbf{B}'$ nasledovne
$$\mathbf{C}(\boldsymbol{S}, R, \boldsymbol{y}) = \begin{cases} b & \text{ak } \mathbf{B}'(\boldsymbol{S}', \boldsymbol{y} + b\boldsymbol{a}) = 1 \\ v \in_R \mathrm{GF}(q) \setminus b & \text{ak } \mathbf{B}'(\boldsymbol{S}', \boldsymbol{y} + b\boldsymbol{a}) = 0 \end{cases}$$

Ak $\boldsymbol{b} = \boldsymbol{R}(\boldsymbol{x})$ platí, $\boldsymbol{S}'(\boldsymbol{x}) = \boldsymbol{S}(\boldsymbol{x}) + R(\boldsymbol{x})\boldsymbol{a} = \boldsymbol{y} + b\boldsymbol{a}$. Algoritmus $\mathbf{C}$ bude v tomto prípade úspešný (odpovie b) v prípade, že algoritmus $\mathbf{B}'(\boldsymbol{S}', \boldsymbol{y}+b\boldsymbol{a}) = \mathbf{B}'(\boldsymbol{S}', \boldsymbol{S}'(\boldsymbol{x})) = = 1$. A to nastane s pravdepodobnosťou

$$\Pr_{\boldsymbol{S}', \boldsymbol{x} \in_R \mathrm{GF}(q)^n} (\mathbf{B}'(\boldsymbol{S}', \boldsymbol{S}'(\boldsymbol{x})) = 1) \geq \frac{1}{2} + \frac{\varepsilon}{2}.$$

Ak $\boldsymbol{b} \neq \boldsymbol{R}(\boldsymbol{x})$, platí, že $\boldsymbol{y} + b\boldsymbol{a} = \boldsymbol{u} \in_R \mathrm{GF}(q)^{kn}$. Algoritmus $\mathbf{C}$ bude úspešný, ak algoritmus $\mathbf{B}'(\boldsymbol{S}', \boldsymbol{y} + b\boldsymbol{a}) = \mathbf{B}'(\boldsymbol{S}', \boldsymbol{u}) = 0$ a z $(q-1)$ hodnôt náhodne vyberie $v \in_R \mathrm{GF}(q) \setminus b$, tak že $v = R(\boldsymbol{x})$. A to nastane s pravdepodobnosťou

$$\Pr_{\boldsymbol{S}', \boldsymbol{u} \in_R \mathrm{GF}(q)^{kn}} ((\mathbf{B}'(\boldsymbol{S}', \boldsymbol{u})) = 1) \wedge (v = R(\boldsymbol{x}))) = \frac{1}{2} \cdot \frac{1}{q-1}.$$

Pravdepodobnosť, že algoritmus $\mathbf{C} = \mathbf{C}(\boldsymbol{S}, R, \boldsymbol{y}) = R(\boldsymbol{x})$ nám dá na výstupe správnu hodnotu je

$$\Pr(\mathbf{C}(\boldsymbol{S}, R, \boldsymbol{y}) = R(\boldsymbol{x})) = \Pr(\mathbf{B}' = 1 | R(\boldsymbol{x}) = b)\Pr(b = R(\boldsymbol{x})) +$$

$$+\Pr((\boldsymbol{B}' = 0) \wedge (v = R(\boldsymbol{x})) | R(\boldsymbol{x}) \neq b)\Pr(b \neq R(\boldsymbol{x})) \geq$$

$$\geq \frac{1}{q}\left(\frac{1}{2} + \frac{\varepsilon}{2}\right) + \left(\frac{q-1}{q}\right)\frac{1}{2}\left(\frac{1}{q-1}\right) = \frac{1}{q}\left(1 + \frac{\varepsilon}{2}\right).$$

Pretože spočítať P_i vyžaduje spočítať všetky monómy Q_i a R, čo netrvá dlhšie ako dva prechody systému QUAD pre ľubovoľný vstup, bude celkový čas algoritmu $\mathbf{C}$ menej ako $T + 2T_{\boldsymbol{S}}$. □

6.1.2 Bezpečnosť generátora keystreamu šifry QUAD v telese GF(2)

Lema 6.3 (Goldreich-Levin). *Nech $\boldsymbol{x}$ označuje neznámy pevne daný n-bitovú vektor a f je pevne zvolená funkcia z $\{0,1\}^n$ do $\{0,1\}^m$. Predpokladajme, že existuje algoritmus $\mathbf{C}$, ktorý dokáže zo vstupu $(f(\boldsymbol{x}), R)$, kde R je ľubovoľná lineárna forma n neznámych, určiť v čase T hodnotu $R(x)$ s pravdepodobnosťou úspechu lepšou ako $\frac{1}{2} + \frac{\varepsilon}{2}$. Potom existuje algoritmus $\mathbf{D}$, ktorý v čase T' zo vstupu $f(\boldsymbol{x})$ vyprodukuje zoznam vektorov $\mathcal{M}$ a platí*

1. *mohutnosť množiny $\mathcal{M}$ je maximálne $2n/\varepsilon^2$;*
2. *pre $\forall \boldsymbol{x_o} \in \mathcal{M}$, platí $f(\boldsymbol{x_0}) = f(\boldsymbol{x})$;*
3. *pravdepodobnosť, že $x \in \mathcal{M}$ je viac ako $1/2$;*
4. $T' \leq \dfrac{2n^2}{\varepsilon^2}\left(T + \log\left(\dfrac{2n}{\varepsilon^2}\right) + 2\right) + \dfrac{2n}{\varepsilon^2}T_f.$

Dôkaz. Pre $1 \leq i \leq n$ definujme lineárne formy $L_i : \{0,1\}^n \to \{0,1\}$ nasledujúcim spôsobom

$$L_i(\boldsymbol{x}) = x_i \text{ , kde } \boldsymbol{x} = (x_1, x_2, \ldots, x_n).$$

Myšlienka dôkazu je zavolať algoritmus $\mathbf{C}(L_i, f(\boldsymbol{x}))$ dostatočne veľa krát aby sme získali všetky hodnoty $x_i = L_i(\boldsymbol{x})$, čim obdržíme hľadaný vektor $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$. K tomu použijeme t náhodne vybraných lineárnych foriem

$$R_1(\boldsymbol{x}), R_2(\boldsymbol{x}), \ldots R_t(\boldsymbol{x}) \,, \qquad \text{kde } R_i : \{0,1\}^n \to \{0,1\} \,,$$

za účelom vytvoriť lineárnu formu $L^\alpha = \bigoplus_{j=1}^t \alpha_j R_j \oplus L_i$ pomocou lineárnej kombinácii foriem R_j a L_i a tým požiadavok na algoritmus $\mathbf{C}$ urobiť náhodným. Pre vektor $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_t) \in \{0,1\}^t$ a lineárnu formu L_i definujeme

$$C(i, \boldsymbol{\alpha}) = \mathbf{C}\left(\bigoplus_{j=1}^t \alpha_j R_j \oplus L_i, f(\boldsymbol{x})\right) \oplus \bigoplus_{j=1}^t \alpha_j k_j,$$

kde $\boldsymbol{k} = (k_1, \ldots, k_t)$ je zvolený vektor z $\{0,1\}^r$.
Predpokladajme, že $R_i(\boldsymbol{x}) = k_i$, pre $1 \leq i \leq r$ za predpokladu, že algoritmus $\mathbf{C}$ nám vráti správnu hodnotu, dostávame

$$\begin{aligned} C(i, \boldsymbol{\alpha}) &= \left(\bigoplus_{j=1}^t \alpha_j R_j \oplus L_i\right)(\boldsymbol{x}) \oplus \bigoplus_{j=1}^t \alpha_j k_j = \\ &= L_i(\boldsymbol{x}) \oplus \bigoplus_{j=1}^t \alpha_j R_j(\boldsymbol{x}) \oplus \bigoplus_{j=1}^t \alpha_j R_j(\boldsymbol{x}) = L_i(\boldsymbol{x}). \end{aligned}$$

Označme $R^k = \bigoplus_{j=1}^t k_j R_j$ a $L^\alpha = \bigoplus_{j=1}^t \alpha_j R_j \oplus L_i$, potom pravdepodobnosť, že $C(i, \boldsymbol{\alpha})$ nám vráti hodnotu $L_i(\boldsymbol{x})$ je

$$\begin{aligned} \Pr\left(C(i, \boldsymbol{\alpha}) = L_i(\boldsymbol{x})\right) =& \Pr\left(\mathbf{C}(L^\alpha, f(\boldsymbol{x})) = L^\alpha(\boldsymbol{x}) | R^k(\boldsymbol{x}) = \bigoplus_{j=1}^t \alpha_j R_j(\boldsymbol{x})\right) + \\ &+ \Pr\left(\mathbf{C}(L^\alpha, f(\boldsymbol{x})) \neq L^\alpha(\boldsymbol{x}) | R^k(\boldsymbol{x}) \neq \bigoplus_{j=1}^t \alpha_j R_j(\boldsymbol{x})\right) = \\ &\geq \frac{1}{2}\left(\frac{1}{2} + \varepsilon\right) + \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{2} + \frac{\varepsilon}{2} \end{aligned}$$

A teda pravdepodobnosť, že $C(i, \boldsymbol{\alpha})$ nám pre ľubovoľné $\boldsymbol{\alpha}$ vráti $L_i(\boldsymbol{x})$ (správnu hodnotu) je väčšia ako 1/2. Preto stačí pre zafixované i spočítať $C(i, \boldsymbol{\alpha})$, pre všetky možné $\boldsymbol{\alpha}$, ktorých je 2^t. V prípade, že viac ako $\frac{2^t}{2}$ hodnôt $C(i, \boldsymbol{\alpha}) = 1$, položíme

hodnotu $L_i(\boldsymbol{x}) = 1$, v opačnom prípade je $L_i(\boldsymbol{x}) = 0$.

Teraz na základe tejto myšlienky popísanej v predchádzajúcom odstavci, podáme formálne správny dôkaz. V dôkaze použijeme Walshovú transformáciu, ktorá je definovaná nasledujúcim spôsobom. Nech $g : \{0,1\}^t \to \mathbb{R}$ je reálna funkcia, potom Walshová transformácia funkcie g je reálna funkcia definovaná nasledovne

$$W(g) = G(u_1, \ldots, u_t) = \sum_{(x_1,x_2,\ldots,x_t)\in\{0,1\}^t} g(x_1, x_2, \ldots, x_t)\,(-1)^{x_1u_1+x_2u_2+\cdots+x_tu_t}\,.$$

Čas potrebný pre výpočet Walshovej transformácie funkcie g je $t \cdot 2^t$.

Algoritmus **D** pracuje nasledovne

1. Náhodne vyberie t lineárnych foriem z $\{0,1\}^n$ do $\{0,1\}$

$$R_1(\boldsymbol{x}), R_2(\boldsymbol{x}), \ldots, R_t(\boldsymbol{x}).$$

2. Pre každé $1 \le i \le n$ a každé $\boldsymbol{\alpha} \in \{0,1\}^t$ spočítame

$$f(i, \boldsymbol{\alpha}) = (-1)^{\mathbf{C}\left(\bigoplus_{j=1}^{t} \alpha_j R_j \oplus L_i, f(\boldsymbol{x})\right)}\,,$$

   následne túto hodnotu uložíme na i-ty riadok j-tého stĺpca tabuľky, kde $j = \sum_{j=1}^{t} \alpha_j 2^j$. Týmto získame tabuľku o rozmere $n \times 2^t$.
   Hodnoty v i-tom riadku tejto tabuľky sú hodnoty funkcie f^i

$$f^i(\boldsymbol{\alpha}) = f^i((\alpha_1, \alpha_2, \ldots, \alpha_t)) = (-1)^{\mathbf{C}\left(\bigoplus_{j=1}^{t} \alpha_j R_j \oplus L_i, f(\boldsymbol{x})\right)}\,.$$

3. Aplikáciou Walshovej transformáciou na funkciu f^i, kde $1 \le i \le n$ získavame funkciu

$$W(f^i) = F^i(k_1, \ldots, k_t) = \sum_{\alpha} (-1)^{\mathbf{C}\left(\bigoplus_{j=1}^{t} \alpha_j R_j \oplus L_i, f(\boldsymbol{x})\right)} \cdot (-1)^{k_1\alpha_1+\cdots k_t\alpha_t} =$$

$$= |\{\boldsymbol{\alpha}; C(i, \boldsymbol{\alpha}) = 0\}| - |\{\boldsymbol{\alpha}; C(i, \boldsymbol{\alpha}) = 1\}|$$

   A teda $F^i(\boldsymbol{k})$ je rozdiel počtu vektorov $\boldsymbol{\alpha}$, kedy $C(i, \boldsymbol{\alpha}) = 0$ a počtu vektorov $\boldsymbol{\alpha}$, kedy $C(i, \boldsymbol{\alpha}) = 1$, pre pevne daný vektor $\boldsymbol{k}$.
   Pre každé $\boldsymbol{k} \in \{0,1\}^t$ zostrojíme vektor $\boldsymbol{z^k} = (z_1^k, z_2^k, \ldots, z_n^k)$, kde

$$z_i^k = \begin{cases} 0 & \text{ak } F^i(\boldsymbol{k}) \ge 0 \\ 1 & \text{ak } F^i(\boldsymbol{k}) < 0. \end{cases}$$

4. V 3. kroku sme získali zoznam $\mathcal{M} = \left\{\boldsymbol{z}^{\boldsymbol{k}}; \boldsymbol{k} \in \{0,1\}^t\right\}$, t. j. množinu 2^t kandidátov na hodnotu $\boldsymbol{x}$. Nakoniec pre každé $\boldsymbol{z}^{\boldsymbol{k}}$ spočítame $f(\boldsymbol{z}^{\boldsymbol{k}})$ a v prípade, že $f(\boldsymbol{z}^{\boldsymbol{k}}) = f(\boldsymbol{x})$ vektor $\boldsymbol{z}^{\boldsymbol{k}}$ ponecháme v zozname $\mathcal{M}$, inač $\boldsymbol{z}_{\boldsymbol{k}}$ zo zoznamu vyčiarkneme.

Celkový čas algoritmu **D** je

$$n \cdot 2^t T + n \cdot 2^t + n \cdot t2^t + n \cdot 2^t + 2^t T_f = n2^t\left(T + t + 2\right) + 2^t T_f.$$

Pretože v 2. kroku pre spočítanie $n \cdot 2^t$ hodnôt $f(i, \boldsymbol{\alpha})$ je potrebné $n2^t$ krát zavolať algoritmus **C** čo trvá $n \cdot 2^t T$. Ďalej do tabuľky vložíme týchto $n \cdot 2^t$ hodnôt. V 3. kroku spočítať n krát Walshovú transformáciu funkcie f^i zaberie $n \cdot t2^t$ operácii. Pri konštrukcii vektorov $\boldsymbol{z}^{\boldsymbol{k}}$ vykonáme $n \cdot 2^t$ porovnaní. V poslednom 4. kroku spočítame 2^t krát hodnotu $f(\boldsymbol{z}_{\boldsymbol{k}})$.

Teraz zhora ohraničíme pravdepodobnosť, že algoritmus **D** neuspeje. To je v prípade, že $\boldsymbol{x} \notin \mathcal{M}$. K čomu stačí aby existovalo aspoň jedno i, kedy $C(i, \boldsymbol{\alpha}) \neq L_i(\boldsymbol{x})$ pre viac ako $2^t/2$ vektorov $\boldsymbol{\alpha}$. (Ekvivalentne pre menej ako $2^t/2$ vektorov $\boldsymbol{\alpha}$ platí, že $C(i, \boldsymbol{\alpha}) = L_i(\boldsymbol{x})$) Označme pravdepodobnosť, že táto udalosť nastane ako p_i,

$$p_i = \Pr\left\{\left|\{\boldsymbol{\alpha}; C(i, \boldsymbol{\alpha}) = L_i \boldsymbol{x}\}\right| < \frac{2^t}{2}\right\} = \Pr\left\{\sum_{\boldsymbol{\alpha}} C(i, \boldsymbol{\alpha}) \oplus L_i \boldsymbol{x} \oplus 1 < \frac{2^t}{2}\right\}.$$

V úvode dôkazu sme ukázali, že pravdepodobnosť, že $C(i, \boldsymbol{\alpha}) = L_i(\boldsymbol{x})$ je $\frac{1}{2} + \frac{\varepsilon}{2}$. Z čoho dostávame, že stredná hodnota pre $C(i, \boldsymbol{\alpha}) \oplus L_i \boldsymbol{x} \oplus 1$ je $\mu_\alpha \geq \frac{1}{2} + \frac{\varepsilon}{2}$ a variácia $\upsilon_\alpha \geq \frac{1}{4} - \frac{\varepsilon^2}{4}$. Preto stredná hodnota μ a variácia σ pravdepodobnosti p_i je

$$\mu \geq 2^t\left(\frac{1}{2} + \frac{\varepsilon}{2}\right) \qquad \sigma^2 \geq 2^t\left(\frac{1}{4} - \frac{\varepsilon^2}{4}\right).$$

Použitím Čebyševovej nerovnosti dostávame

$$\begin{aligned} p_i &= \Pr\left\{\sum_{\boldsymbol{\alpha}} C(i, \boldsymbol{\alpha}) \oplus L_i(\boldsymbol{x}) \oplus 1 < \frac{2^t}{2}\right\} = \\ &= \Pr\left\{\sum_{\boldsymbol{\alpha}} C(i, \boldsymbol{\alpha}) \oplus L_i(\boldsymbol{x}) \oplus 1 - \mu < -\frac{2^t \varepsilon}{2}\right\} \leq \\ &= \Pr\left\{\left|\sum_{\boldsymbol{\alpha}} C(i, \boldsymbol{\alpha}) \oplus L_i(\boldsymbol{x}) \oplus 1 - \mu\right| > \frac{2^t \varepsilon}{2}\right\} \leq \frac{\sigma^2}{\left(2^t \frac{\varepsilon}{2}\right)^2} \leq \frac{1}{2^t \varepsilon^2} \end{aligned}$$

A teda pravdepodobnosť, že algoritmus **D** neuspeje je $n \cdot \frac{1}{2^t \varepsilon^2}$. A keďže chceme aby pravdepodobnosť úspechu algoritmu **D** bola väčšia ako $1/2$, zvolíme t tak aby

$$2^t \geq \frac{2n}{\varepsilon^2}.$$

A teda mohutnosť množiny $\mathcal{M}$ je maximálne $2^t = \frac{2n}{\varepsilon^2}$ a celková zložitosť algoritmu $\mathbf{D}$ je

$$\frac{2n^2}{\varepsilon^2}\left(T + \log\left(\frac{2n}{\varepsilon^2}\right) + 2\right) + \frac{2n}{\varepsilon^2}T_f.$$

□

Pravdepodobnosť úspechu algoritmu $\mathbf{C}$ z vety 6.2, je cez všetky dvojice $(\boldsymbol{S}, \boldsymbol{x})$, ale tvrdenie z lemy 6.3, platí len pre zafixovanú dvojicu. Vo vete 6.4 preto ukážeme, že existuje nezanedbateľná časť dvojíc $(\boldsymbol{S}, \boldsymbol{x})$, pre ktoré sú predpoklady lemy 6.3 splnené.

Veta 6.4. *Predpokladajme, že existuje algoritmus* $\mathbf{C}$*, ktorý z náhodne vybraného kvadratického systému* $\boldsymbol{S}$*, z náhodne vybranej lineárnej formy* R *a vektora* $\boldsymbol{y} = \boldsymbol{S}(\boldsymbol{x})$*, kde* $\boldsymbol{x} \in_R \mathrm{GF}(2)^n$ *je neznámy náhodne vybraný vektor, nájde v čase* T *hodnotu* $R(\boldsymbol{x})$ *s pravdepodobnosťou lepšou ako* $\frac{1}{2} + \varepsilon$ *pre každú trojicu* $(\boldsymbol{x}, \boldsymbol{S}, \boldsymbol{R})$*. Potom existuje algoritmus* $\mathbf{D}$*, ktorý z hodnoty* $\boldsymbol{S}(\boldsymbol{x})$ *vypočítanej z neznámeho vektora* $\boldsymbol{x} \in_R \mathrm{GF}(2)^n$ *nájde vektor* $\boldsymbol{x}$ *s pravdepodobnosťou úspechu lepšou ako* $\frac{\varepsilon}{2}$ *pre všetky dvojice* $(\boldsymbol{S}, \boldsymbol{x})$ *v čase*

$$T' \leq \frac{8n^2}{\varepsilon^2}\left(T + \log\left(\frac{8n}{\varepsilon^2}\right) + 2\right) + \frac{8n}{\varepsilon^2}T_{\boldsymbol{S}}.$$

Dôkaz. Predpoklad o algoritme $\mathbf{C}$, môžeme zapísať nasledovne

$$\Pr_{(\boldsymbol{x},R,\boldsymbol{S})\in\{0,1\}^{n+n+mN}}\left(\mathbf{C}(\boldsymbol{S}, R, \boldsymbol{S}(\boldsymbol{x})) = R(\boldsymbol{x})\right) \geq \frac{1}{2} + \varepsilon.$$

Z toho vyplýva, že pre najmenej časť ε zo všetkých možných dvojíc $(\boldsymbol{x}, \boldsymbol{S})$ platí,

$$\Pr_{R\in\{0,1\}^n}\left(\mathbf{C}(\boldsymbol{S}, R, \boldsymbol{S}(\boldsymbol{x})) = R(\boldsymbol{x})\right) \geq \frac{1}{2} + \frac{\varepsilon}{2}.$$

Ak by tomu tak nebolo, tak by pre najmenej časť $(1-\varepsilon)$ zo všetkých dvojíc $(\boldsymbol{x}, \boldsymbol{S})$ platí, $\Pr_{R\in\{0,1\}^n}\left(\mathbf{C}(\boldsymbol{S}, R, \boldsymbol{S}(\boldsymbol{x})) = R(\boldsymbol{x})\right) < \frac{1}{2} + \frac{\varepsilon}{2}$ z čoho vyplýva, že

$$\Pr_{(\boldsymbol{x},R,\boldsymbol{S})\in\{0,1\}^{n+n+mN}}\left(\mathbf{C}(\boldsymbol{S}, \boldsymbol{S}(\boldsymbol{x}), R) = R(\boldsymbol{x})\right) < (1-\varepsilon)(\frac{1}{2} + \frac{\varepsilon}{2}) + \varepsilon = \frac{1}{2} + \varepsilon - \varepsilon^2,$$

čo je v spore s predpokladom. Takže najmenej časť ε zo všetkých možných dvojíc $(\boldsymbol{x}, \boldsymbol{S})$ spĺňa predpoklady lemy 6.3, čo nám dáva existenciu algoritmu $\mathbf{D}$ s pravdepodobnosťou úspechu lepšou ako $\frac{1}{2}\varepsilon$. □

Veta 6.5 (QUAD v GF(2))**.** *Nech* $L = \lambda(k-1)n$ *označuje dĺžku keystreamu vygenerovaného v čase* $\lambda T_{\boldsymbol{S}}$*, ktorý dostaneme vykonaním* λ *prechodov konštrukcie šifry* $\mathrm{QUAD}(k, n, 2)$*, asociovanej so známym, náhodne vybraným kvadratickým systémom* $\boldsymbol{S}$ *a neznámou náhodne vybranou inicializačnou hodnotou* $\boldsymbol{x} \in_R \mathrm{GF}(2)^n$*. Predpokladajme, že existuje algoritmus* $\mathbf{A}$*, ktorý dokáže rozlíšiť tento keystream od náhodne*

vybranej L-bitovej postupnosti, v čase T s výhodou ε. Potom existuje algoritmus $\mathbf{D}$, *ktorý z hodnoty $\boldsymbol{S}(\boldsymbol{x})$ vypočítanej z neznámeho vektora $\boldsymbol{x} \in_R \mathrm{GF}(2)^n$ nájde vektor $\boldsymbol{x}$ s pravdepodobnosťou úspechu lepšou ako $\frac{\varepsilon}{2^3\lambda}$ pre všetky dvojice $(\boldsymbol{S}, \boldsymbol{x})$ v čase*

$$T' \leq \frac{2^7 n^2 \lambda^2}{\varepsilon^2} \left(T + (\lambda + 2)\, T_S + \log\left(\frac{2^7 n \lambda^2}{\varepsilon^2}\right) + 2 \right) + \frac{2^7 n \lambda^2}{\varepsilon^2} T_S.$$

Dôkaz. Postupným aplikovaním vety 6.1, 6.2, 6.4 dostávame existenciu algoritmu D, ako aj hodnoty pre pravdepodobnosť úspechu a časovú zložitosř tohoto algoritmu. □

6.1.3 Dôkaz bezpečnosti generátora keystreamu šifry QUAD

Nasledujúcu veta je zovšeobecnením Goldreich-Levin vety (6.3). Dôkaz je technicky náročnejší a do našej problematiky, by nepriniesol nič nové. Preto nasledujúcu vetu uvádzame bez dôkazu, ktorý je možné nájsť v článku [3].

Veta 6.6 (Goldereich-Rubinfeld-Sudan)**.** *Predpokladajme, že existuje algoritmus* $\mathbf{C}$, *ktorý z náhodne vybraného kvadratického systému $\boldsymbol{S}$, z náhodne vybranej lineárnej formy R a vektora $\boldsymbol{y} = \boldsymbol{S}(\boldsymbol{x})$, kde $\boldsymbol{x} \in_R \mathrm{GF}(q)^n$ je neznámy náhodne vybraný vektor, nájde v čase T hodnotu $R(\boldsymbol{x})$ v čase T s pravdepodobnosťou úspechu (pre všetky trojice $(\boldsymbol{S}, R, \boldsymbol{S}(\boldsymbol{x}))$ lepšou ako $\frac{1}{q} + \varepsilon$. Potom existuje algoritmus* $\mathbf{D}$, *ktorý zo vstupu $(\boldsymbol{S}, \boldsymbol{S}(\boldsymbol{x}))$, kde $\boldsymbol{x} \in_R \mathrm{GF}(g)^n$ je neznámy náhodne vybraný vektor, nájde vektor $\boldsymbol{x}$ s pravdepodobnosťou (pre všetky dvojice $(\boldsymbol{S}, \boldsymbol{x})$) lepšou ako $\frac{\varepsilon}{2}$ v čase*

$$T' \leq 2^{10} \left(\frac{nq}{\varepsilon^5}\right) \log^2\left(\frac{n}{\varepsilon}\right) T + \left(1 - \frac{1}{q}\right)^2 \varepsilon^{-2} T_{\boldsymbol{S}}.$$

Veta 6.7 (QUAD)**.** *Nech $L = \lambda(k-1)n$ označuje dĺžku keystreamu vygenerovaného v čase $\lambda T_{\boldsymbol{S}}$, ktorý dostaneme vykonaním λ prechodov konštrukcie šifry* $\mathrm{QUAD}(k, n, q)$, *asociovanej so známym, náhodne vybraným kvadratickým systémom $\boldsymbol{S}$ a neznámou náhodne vybranou inicializačnou hodnotou $\boldsymbol{x} \in_R \mathrm{GF}(q)^n$. Predpokladajme, že existuje algoritmus* $\mathbf{A}$, *ktorý dokáže rozlíšiť tento keystream od náhodne vybranej postupnosti z $\mathrm{GF}(q)$ dĺžky L, v čase T s výhodou ε. Potom existuje algoritmus* $\mathbf{D}$, *ktorý zo vstupu $(\boldsymbol{S}, \boldsymbol{S}(\boldsymbol{x}))$, kde $\boldsymbol{x} \in_R \mathrm{GF}(g)^n$ je neznámy náhodne vybraný vektor, nájde vektor $\boldsymbol{x}$ s pravdepodobnosťou (pre všetky dvojice $(\boldsymbol{S}, \boldsymbol{x})$) lepšou ako $\frac{\varepsilon}{4q\lambda}$ v čase*

$$T' \leq 2^{15} \frac{nq^6\lambda^5}{\varepsilon^5} \log^2\left(\frac{2qn\lambda}{\varepsilon}\right) (T + (\lambda + 2)\, T_S) + \left(1 - \frac{1}{q}\right)^2 \frac{4q^2\lambda^2}{\varepsilon^2} T_{\boldsymbol{S}}.$$

Dôkaz. Postupným aplikovaním vety 6.1, 6.2 a 6.6. □

6.1.4 Zhrnutie dôkazu

Na šifru Quad, ktorá z inicializačnej hodnoty $\boldsymbol{x_0}$ nazývanej semiačko, vygeneruje postupnosť dĺžky L, sa pozeráme ako na generátor pseudonáhodných čísel a požadujeme

aby generovala keystream, ktorý by bol nerozlíšiteľný od náhodne vybranej postupnosti. QUAD nie je jediná prúdová šifra s preukázateľnou bezpečnosťou, QUAD je ale výnimočná z niekoľkých dôvodov. Za prvé, dôkaz bezpečnosti stojí na dobre známom a preskúmanom probléme, ktorý patrí do triedy $\mathcal{NP}$ ťažkých problémov. Za druhé, QUAD je pre určité parametre dostatočne rýchly.

Vo vete 6.5 sme ukázali, že ak máme algoritmus **A**, ktorý dokáže rozlíšiť keystream vygenerovaný šifrou QUAD$(k, n, 2)$ od náhodne vybranej postupnosti rovnakej dĺžky, tak existuje algoritmus **D** s nasledujúcimi vlastnosťami.

Dokáže z hodnoty $\boldsymbol{S}(\boldsymbol{x})$, kde $\boldsymbol{S}$ je kvadratický systém kn polynómov n premenných nad telesom GF(2) a $\boldsymbol{x}$ je neznámy náhodne vybraný vektor, nájsť $\boldsymbol{y} \in \mathrm{GF}(2)^n$, také že $\boldsymbol{S}(\boldsymbol{x}) = \boldsymbol{S}(\boldsymbol{y})$. Kedže kvadratický systém $\boldsymbol{S}$ ako zobrazenie z $\mathrm{GF}(2)^n$ do $\mathrm{GF}(2)^{kn}$ pre $k > 1$, je takmer iste injektívnym (prostým) zobrazením, platí $\boldsymbol{x} = \boldsymbol{y}$. Pravdepodobnosť úspechu algoritmu **D** je najmenej $\frac{\varepsilon}{2^3\lambda}$ a jeho časová zložitosť je

$$T' \leq \frac{2^7 n^2 \lambda^2}{\varepsilon^2} \left(T + (\lambda + 2)\, T_S + \log\left(\frac{2^7 n \lambda^2}{\varepsilon^2} \right) + 2 \right) + \frac{2^7 n \lambda^2}{\varepsilon^2} T_S.$$

Takže **D** je pravdepodobnostný algoritmus na riešenie kvadratických systémov kn polynómov n premenných, pre $k > 1$. Ak by takýto algoritmus existoval pravdepodobne[4] by tento algoritmus dokázal riešiť všetky $\mathcal{NP}$ úplné problémy.

Dôkaz bezpečnosti keystreamu nezahrňuje bezpečnosť generovania inicializačnej hodnoty $\boldsymbol{x_0}$ z kluča $\boldsymbol{K}$ a inicializačného vektora `IV`. Vo všetkých tvrdeniach sme predpokladali, že inicializačná hodnota $\boldsymbol{x_0}$ bola vybraná náhodne.

Dôkaz bezpečnosti keystreamu nám nedáva záruku, že bezpečnosť keystreamu vyprodukovaného konkrétnym systémom kvadratických polynómov bude bezpečný, ale hovorí, že vo väčšine prípadov bude tento keystream bezpečný.

K tomu aby sme ukázali, že samotny generátor keystreamu šifry QUAD(k,n,q) je preukazateľne bezpečný PRNG. Musíme nastavit parametre šifry QUAD k, n, q a počet interácií λ (dĺžka keystreamu), ako aj konkrétnu požadovanú úroveň zabezpečenia šifry T, ε, tak aby

1. Z predpokladu, že existuje algoritmus (útočník) **A**, ktorý v čase T rozlíši keystream dĺžky L od náhodnej postupnosti s výhodou ε, dostávame z vety 6.7 algoritmus **B** na riešenie systému kn kvadratických rovníc n premenných nad telesom GF(q) v čase T' a s pravdepodobnosťou ε'. **Kde T' a ε' sú funkcie závisle práve od zvolených parametroch $k, n, q, \lambda, T, \varepsilon$.**

2. Algoritmus **B** vyriešil kvadratický systém (pre zvolené parametre) podstatne rýchlejšie ako doteraz najrýchlejší známi algoritmus na riešenie kn kvadratických rovníc n premenných nad telesom GF(q), čo považujeme za spor. A teda predpoklad o existencii útočníka **A** bol nesprávny.

[4]Treba si uvedomiť, že **D** by neriešil MQ problém všeobecne, vyriešil by MQ problém v prípade, kde počet rovníc je väčší ako počet neznámych.

Presné stanovenie hodnôt parametrov, pre ktoré šifra QUAD je (ne)preukázateľne bezpečná nájdeme v samostatnej kapitole 7.

6.2 Bezpečnost prúdovej šifry QUAD s IV

V úvode ukážeme, že bezpečnostné požiadavky pre prúdovú šifru s inicializačným vektorom IV sú v istom zmysle prirodzeným zovšeobecnením bezpečnostných požiadavkov pre prúdovú šifru bez IV, kde sme požadovali aby generátor keystreamu bol PRNG. Ďalej ukážeme konštrukciu „bezpečnej“ prúdovej šifry s IV (PRF) z generátora pseudonáhodných čísel. Dokážeme, že bezpečnosť tejto prúdovej šifry závisi práve na bezpečnosti použitého generátora pseudonáhodných čísel. Táto konštrukcia pochádza od autorov Goldreich, Goldwasser a Micali z článku [7]. Šifra QUAD používa práve túto konštrukciu. Na záver oddielu ukážeme, že bezpečnostný argument pre šifru QUAD z vety 6.7 a 6.5 môže byť rozšírený na celú šifru vrátane generovania inicializačnej hodnoty. Vďaka čomu môžeme bezpečnosť celej šifry QUAD previesť na problém riešenia kvadratického systému.

6.2.1 Bezpečnostné požiadavky pre prúdovú šifru s IV

Generátor keystreamu $\mathbf{G}$ je v tomto prípade funkcia dvoch premenných, ktorá z dvojice $(K, \mathtt{IV})$ vygeneruje keystream

$$\mathbf{G}(K, \mathtt{IV}) = \text{keystream}.$$

Funkcia $\mathbf{G}$ nám definuje množinu funkcií $\mathcal{G} = \{\mathbf{G}_K;\ K \in \mathcal{K}\}$. Bezpečnostným požiadavkom na prúdovú šifru s IV je aby $\mathcal{G}$ bola množinou pseudonáhodných funkcií viď. definícia 2.9.

Pokúsime sa objasniť dôvod tejto definície vysvetliť na konkrétnom príklade prúdovej šifre QUAD. Generátor keystreamu je v tomto prípade funkcia

$$\textsc{quad}(k, n, q) = \mathbf{Q} : \mathrm{GF}(q)^n \times \{0,1\}^{|\mathtt{IV}|} \to \mathrm{GF}(q)^L,$$

kde L značí maximálnu dĺžku kesystreamu vygenerovanú jednou dvojicou $(\boldsymbol{K}, \mathtt{IV})$. Nech $|\mathtt{IV}| = m$.

Zadefinujme generátor pseudonáhodných čísel, ktorý pomocou funkcie $\mathbf{Q}$ vygeneruje z tajného kľúča $\boldsymbol{K} \in \mathrm{GF}(q)^n$ keystream exponencionálnej dĺžky $2^m \cdot L$:

$\{Z_{\mathtt{IV}} = \mathbf{Q}_K(\mathtt{IV})\}_{\mathtt{IV} \in \{0,1\}^n}$. Tento keystream má špeciálnu vlastnosťou, že pre ľubovoľný inicializačný vektor IV časová zložitosť pre priamy prístup do L znakovej podpostupnosti $Z_{\mathtt{IV}}$ je konštantná.

A preto sa zdá logicky požadovať od bezpečnej prúdovej šifry s IV, aby neexistoval algoritmus, ktorý aj napriek dostatočne veľkému počtu r priamych prístupov k L

znakových podpostupnostiam postupnosti $\{Z_{\mathtt{IV}}\}$ asociovanej s náhodne vybraným neznámym kľúčom K alebo k náhodne vybranej $2^n L$ znakovej postupnosti, rozlíši oba prípady v čase T s výhodou ε. Hodnoty T, ε a r sú volené tak, aby vyhovovali konkrétnym bezpečnostným požiadavkám. Zhora uvedené, je teda akýmsi zovšeobecnením bezpečnostného požiadavku pre prúdovú šifru bez IV.

Na postupnosť $\{Z_{\mathtt{IV}}\}$, ako aj na náhodne vybranú $2^n L$ znakovú postupnosť, sa môžeme pozerať ako na funkciu $\mathbf{G}_K$, respektíve náhodnú funkciu z $\{0,1\}^n$ do $\mathrm{GF}(q)^L$ a priamy prístup do týchto postupnosti je orakulovský prístup k týmto funkciám. Takže predošlý požiadavok na bezpečnosť prúdovej šifry s IV sformulovaný vyšie je ekvivaletný požiadavku aby množina funkcií $\mathcal{G} = \{\mathbf{G}_K;\ K \in \mathcal{K}\}$ bola nerozlíšiteľná od množiny $\mathcal{H} = \left\{f;\ f : \{0,1\}^n \to \mathrm{GF}(q)^L\right\}$.

6.2.2 Jednoduchá veta o skladaní

Generovanie keystreamu v prípade prúdových šifier s IV môžeme rozdeliť do dvoch častí

1. Generovanie inicializačnej hodnoty x_0 z tajného kľúča K a IV, čo môžeme vyjadriť funkciou F
$$F(K, \mathtt{IV}) = x_0;$$
2. Generovanie samotného keystreamu z hodnoty x_0
$$g(x_0) = \text{keystream}.$$

Našim cieľom je ukázať, že funkcia $\mathbf{G}$ definovaná ako

$$\mathbf{G}(K, \mathtt{IV}) = g \circ F(K, \mathtt{IV})$$

je PRF (parametrizovaná podľa K). A ukázať vzťah medzi jej bezpečnosťou s bezpečnosťou g a F . Ukážeme, že ak generátor g je „bezpečný“ PRNG a funckia $\mathbf{F}$ je „bezpečná“ PRF, tak funkcia $\mathbf{G} = g \circ \mathbf{F}$ je „bezpečná“ PRF.

Lema 6.8. *Nech $g : \mathrm{GF}(q)^n \to \mathrm{GF}(q)^L$ je PRNG, ktorý pracuje v čase T_g, potom platí nasledujúca nerovnosť*

$$\mathbf{Adv}_g^{prng}(T, r) \leq r\mathbf{Adv}_g^{prng}(T + qT_g).$$

Dôkaz. Dôkaz je veľmi podobný dôkazu vety 6.1. Nech algoritmus $\mathbf{A}$ rozlišuje r keystreamov vyprodukovaných generátorom g z r náhodne vybraných inicializačných vektorov $\boldsymbol{x_i} \in_R \mathrm{GF}(q)^n$

$$g(\boldsymbol{x_1}), g(\boldsymbol{x_2}), \ldots, g(\boldsymbol{x_r})$$

od r náhodne vybraných vektorov z $\mathrm{GF}(q)^L$ v čase T s výhodou ε. Definujme pravdepodobnostné rozdelenia $D^i(g)$ na $\mathrm{GF}(q)^{rL}$, kde $D^i(g)$ je asociované s hodnotami

$$\boldsymbol{Z}^i = \left(g(\boldsymbol{x_1}), g(\boldsymbol{x_2}), \ldots, g(\boldsymbol{x_i}), \boldsymbol{m_{i+1}}, \ldots, \boldsymbol{m_r}\right),$$

kde $0 \leq i \leq r$, $\boldsymbol{x_i} \in_R \mathrm{GF}(q)^n$ a $\boldsymbol{m_i} \in_R \mathrm{GF}(q)^L$. Rozdelenie $D^r(g)$ je pravdepodobnostné rozdelenie r keystreamov o dĺžke L a naopak $D^0(g)$ je rovnomerným rozdelením vektorov na $\mathrm{GF}(q)^{rL}$ (r náhodne vybraných vektorov z $\mathrm{GF}(q)^L$). Pravdepodobnosť, že $\mathbf{A}$ príme (odpovie 1) náhodnú L znakovú postupnosť s rozdelením $D^i(g)$ označíme p^i. Z predpokladu vyplýva, že $|p^0 - p^r| \geq \varepsilon$.

Definujme algoritmus $\mathbf{B}$, ktorý pracuje so vstupom $\boldsymbol{y} \in \mathrm{GF}(q)^L$ nasledujúcim spôsobom

1. náhodne vyberie hodnotu $1 \leq i \leq r$;
2. zostrojí vektor $\boldsymbol{Z}(\boldsymbol{y}) = (g(\boldsymbol{x_1}), \ldots g(\boldsymbol{x_{i-1}}), \boldsymbol{y}, \boldsymbol{m_{i+1}}, \ldots \boldsymbol{m_r})$;
3. odpovie $\mathbf{A}(\boldsymbol{Z}(\boldsymbol{y}))$.

V prípade, že platí $\boldsymbol{y} = g(\boldsymbol{x})$, kde $\boldsymbol{x} \in_R \mathrm{GF}(q)^n$ má $\boldsymbol{Z}(\boldsymbol{y})$ pravdepodobnostné rozdelenie Z^i. V prípade, že $\boldsymbol{y} \in_R \mathrm{GF}(q)^L$ má $\boldsymbol{Z}(\boldsymbol{y})$ pravdepodobnostné rozdelenie Z^{i-1}.

Z čoho dostávame, že algoritmus $\mathbf{B}$ rozlišuje keystream o dĺžke L vygenerovaný generátorom pseudonáhodných čísel g od náhodne vybranej L znakovej postupnosti z $\mathrm{GF}(q)$ v čase $rT_g + T$ (2. + 3. krok) s výhodou

$$\left|\Pr_{\boldsymbol{x}}(\mathbf{B}(g(\boldsymbol{x})) = 1) - \Pr_{\boldsymbol{y}}(\mathbf{B}(\boldsymbol{y}) = 1)\right| =$$

$$= \left|\Pr_{\boldsymbol{x}}(\mathbf{A}(\boldsymbol{Z}(g(\boldsymbol{x}))) = 1) - \Pr_{\boldsymbol{y}}(\mathbf{A}(\boldsymbol{Z}(\boldsymbol{y})) = 1)\right| =$$

$$= \left|\frac{1}{r}\sum_{i=1}^{r} p^i - \frac{1}{r}\sum_{i=0}^{r-1} p^i\right| = \frac{1}{r}\left|p^0 - p^r\right| \geq \frac{\varepsilon}{r}.$$

Ukázali sme $\mathbf{Adv}_g^{\mathrm{prng}}(\mathbf{A}) = r\mathbf{Adv}_g^{\mathrm{prng}}(\mathbf{B}) \leq r\mathbf{Adv}_g^{\mathrm{prng}}(T + rT_g)$. Pretože táto nerovnosť platí pre ľubovoľné $\mathbf{A}$, platí

$$\mathbf{Adv}_g^{\mathrm{prng}}(T, r) \leq r\mathbf{Adv}_g^{\mathrm{prng}}(T + qT_g).$$

□

Veta 6.9. *Uvažujme zobrazenie $F : \mathcal{K} \times \mathrm{GF}(q)^m \to \mathrm{GF}(q)^n$ a k nemu prislúchajúcu množinu pseudonáhodných funkcií $\mathcal{F} = \{F_K\}$ a zobrazenie $g : \mathrm{GF}(q)^n \to \mathrm{GF}(q)^L$ je PRNG, ktorý pracuje v čase T_g. Označme zložené zobrazenie $G = g \circ F$, potom platí*

$$\mathbf{Adv}_G^{prf}(T, r) \leq \mathbf{Adv}_F^{prf}(T + rT_g, r) + r\mathbf{Adv}_g^{prng}(T + rT_g).$$

Dôkaz. Potrebujeme zhora ohraničiť výhodu algoritmu $\mathbf{A}$, ktorý v čase T pomocou r dotazov rozlišuje $g_K \in_R \{g \circ F_K\}$ od náhodnej funkcie $g^* : \mathrm{GF}(q)^m \to \mathrm{GF}(q)^L$

$$\mathbf{Adv}_G^{\mathrm{prf}}(\mathbf{A}) = |\Pr(\mathbf{A}(g_K) = 1) - \Pr(\mathbf{A}(g^*) = 1)|\,.$$

Uvažujme zloženú funckiu $g \circ f^* : \mathrm{GF}(q)^m \to \mathrm{GF}(q)^L$, kde g je generátor pseudonáhodných čísel $g : \mathrm{GF}(q)^n \to \mathrm{GF}(q)^L$ a $f^* : \mathrm{GF}(q)^m \to \mathrm{GF}(q)^n$ je náhodná funckia. Z trojuholníkovej nerovnosti dostávame

$$\begin{aligned}\mathbf{Adv}_G^{\mathrm{prf}}(\mathbf{A}) \leq\ & |\Pr(\mathbf{A}(g_K) = 1) - \Pr(\mathbf{A}(g \circ f^*) = 1)| + \\ & + |\Pr(\mathbf{A}(g \circ f^*) = 1) - \Pr(\mathbf{A}(g^*) = 1)|.\end{aligned}$$

Označme prvú a druhú absolutnú hodnotu z pravej strany nerovnice ako δ_1 a δ_2. Najprv zhora ohraničíme hodnotu δ_2. Pre ľubovoľné $\boldsymbol{x_i} \in \mathrm{GF}(q)^m$ a $\boldsymbol{x_j} \in \mathrm{GF}(q)^n$ platí

$$\begin{aligned} g \circ f^*(\boldsymbol{x_i}) &\equiv g(\boldsymbol{y_i}), \text{ kde } \boldsymbol{y_i} \in_R \mathrm{GF}(q)^n \\ g^*(\boldsymbol{x_j}) &\equiv \boldsymbol{y_j}, \quad \text{ kde } \boldsymbol{y_j} \in_R \mathrm{GF}(q)^L. \end{aligned}$$

Čo znamená, že algoritmus $\mathbf{A}$, ktorý má orakulovský prístup k funkcii f a môže položiť r dotazov, pošle orakulu dotaz $\boldsymbol{x_i} \in \mathrm{GF}(q)^n$, v prípade, že $f = g \circ f^*$ obdrží z orakula odpoveď v podobe hodnoty $g \circ f^*(\boldsymbol{x_i})$ a pretože f^* je náhodná funckia platí, že $\boldsymbol{y_i} = f^*(\boldsymbol{x_i}) \in_R \mathrm{GF}(q)^n$ a teda hodnota $g \circ f^*(\boldsymbol{x_i})$ je ekvivaletná hodnote $g(\boldsymbol{y_i})$ pre náhodne vybrané $\boldsymbol{y_i} \in_R \mathrm{GF}(q)^n$. V prípade, že $f = g^*$ obdrží z orakula odpoveď $g^*(\boldsymbol{x_i})$, čo je ekvivaletné náhodne vybranej hodnote z $\mathrm{GF}(q)^L$.

Na základe tohoto pozorovania zostrojíme jednoduchou úpravou algoritmu $\mathbf{A}$ algoritmus $\mathbf{B}$, ktorý rozlišuje r hodnôt $(g(\boldsymbol{x_1}), \ldots, g(\boldsymbol{x_r}))$, kde $\boldsymbol{x_i} \in_R \mathrm{GF}(q)^n$ od r náhodne vybraných hodnôt (vektorov) z $\mathrm{GF}(q)^L$.

Algoritmus $\mathbf{B}$ so vstupom $(\boldsymbol{y_1}, \ldots \boldsymbol{y_r})$ pracuje nasledovone. Zavolá algoritmus $\mathbf{A}$, ktorý pošle r dotazov $\boldsymbol{x_i}$, na ktoré obdrží odpoveď $\boldsymbol{y_i}$. Keď skončí algoritmus $\mathbf{A}$ skončí aj algoritmus $\mathbf{B}$ a vratí výsledok z algoritmu $\mathbf{A}$. Pre algoritmus $\mathbf{B}$ platí

$$\begin{aligned} \Pr(\mathbf{B}(g(\boldsymbol{y_1}), \ldots g(\boldsymbol{y_r})) = 1) &= \Pr(\mathbf{A}(g \circ f^*) = 1) \\ \Pr(\mathbf{B}(\boldsymbol{y_1}, \ldots \boldsymbol{y_r}) = 1) &= \Pr(\mathbf{A}(g^*) = 1) \end{aligned}$$

a teda

$$\delta_2 = \mathbf{Adv}_g^{\mathrm{prng}}(\mathbf{B}) \leq \mathbf{Adv}_g^{\mathrm{prng}}(T, r) \overset{\text{lema 6.8}}{\leq} r\mathbf{Adv}_g^{\mathrm{prng}}(T + rT_g).$$

Ďalej definujme algoritmus $\mathbf{C}$, ktorý bude rozlišovať funckiu F_K od náhodnej funckie $f^* : \mathrm{GF}(q)^m \to \mathrm{GF}(q)^m$, nasledujúcim spôsobom

1. zavolá algoritmus $\mathbf{A}$, ktorý pošle r orakulovských dotazov $\boldsymbol{x_i} \in \mathrm{GF}(q)^m$ na ktoré obdrží odpoveď $f(\boldsymbol{x_i})$, kde f je buď F_K alebo náhodná funckia f^*.

2. Algoritmus **C**, ale túto odpoveď pozmení na $g(f(\boldsymbol{x_i}))$ a teda algoritmus **A** na dotaz $\boldsymbol{x_i}$ obdrži odpoved $g(f(\boldsymbol{x_i}))$.

3. Keď skončí algoritmus **A** skončí aj algoritmus **C** a vratí výsledok z algoritmu **A**.

Pre algoritmus **C** teda platí

$$\Pr(\mathbf{C}(f_K) = 1) = \Pr(\mathbf{A}(g \circ f_K) = 1) = \Pr(\mathbf{A}(g_K) = 1)$$
$$\Pr(\mathbf{C}(f^*) = 1) = \Pr(\mathbf{A}(g \circ f^*) = 1)$$

Časová zložitosť algoritmu **C** je rovná časovej zložitosti algoritmu **A** plus r krát čas potrebný na spočítanie funckie g. Z čoho dostávame

$$\delta_1 = \mathbf{Adv}_F^{\mathrm{prf}}(\mathbf{C}) \leq \mathbf{Adv}_F^{\mathrm{prf}}(T + rT_g, r).$$

Nakoniec pre algoritmus **A** dostávame

$$\mathbf{Adv}_G^{\mathrm{prf}}(\mathbf{A}) \leq \mathbf{Adv}_F^{\mathrm{prf}}(T + rT_g, r) + r\mathbf{Adv}_g^{\mathrm{prng}}(T + rT_g)$$

a kedže **A** je ľubovoľný algoritmus, ktorý pracuje v čase T s r dotazmi z predchádzajúcej nerovnosti dostávame

$$\mathbf{Adv}_G^{\mathrm{prf}}(T, r) \leq \mathbf{Adv}_F^{\mathrm{prf}}(T + rT_g, r) + r\mathbf{Adv}_g^{\mathrm{prng}}(T + rT_g).$$

□

Aplikáciou vety 6.9 na bezpečnosť prúdovej šifry používajúcej inicializačný vektor dostávame nasledujúce tvrdenie.

Tvrdenie 6.10. *Prúdová šifra G s* `IV`

$$G(K, \texttt{IV}) = g \circ F(K, \texttt{IV}) = keystream,$$

kde F značí generátor, ktorý z kľúča K a inicializačného vektora `IV` *vygeneruje inicializačnú hodnotu x_0 a zobrazenie g je generátor keystreamu, ktorý z x_0 vygeneruje keystream, sa považuje za bezpečnú ak*

1. *F je PRF*

2. *g je PRNG*

3. *hodnota $\mathbf{Adv}_G^{prf}(T, r)$, ktorú získame z vety 6.9 nám garantuje dostatočnú odolnosť voči prípadným útokom.*

V oddiely 6.1 vo vetách 6.5 a 6.7 sme dokázali, že existuje konštrukcia generátora pseudonáhodných čísel, kde dokážeme zhora ohraničiť hodnotu $\mathbf{Adv}_g^{\mathrm{prng}}(T)$. Ostáva už len problém zostrojiť generátor, ktorý z kľúča K a inicializačného vektora `IV` vygeneruje inicializačnú hodnotu, tak aby bol PRF a zároveň bola splnená *3.* podmienka z predchádzajúceho tvrdenia.

6.2.3 Konštrukcia PRF z PRNG pomocou binárneho stromu

V tejto časti ukážeme konštrukciu generátora, ktorý z kľúča K a inicializačného vektora `IV` vygeneruje inicializačnú hodnotu x_0, tak aby tento generátor bol PRF. Konštrukcia tohoto generátora je odvodená z návrhu od Goldereich, Goldwasse a Micali v [7]. Je to jdnoduchý návod na to ako zostrojiť PRF z PRNG, tak aby bezpečnosť tohoto PRF záviselá len na bezpečnosti použitého PRNG[5].

K zostrojeniu prúdovej šifry s `IV` teda budeme potrebovať 2 generátory pseudonáhodných čísel, z prvého za pomoci stromovej konštrukcie zostrojíme generátor na generovanie inicializačnej hodnoty x_0 z kľúča K a inicializačného vektora `IV`; druhý PRNG z inicializačnej hodnoty x_0 vygeneruje keystream. Tieto dva generátory pseudonáhodných čísel môžu byť samozrejme rovnaké a teda je možné zostrojiť prúdovú šifru s `IV` z jedného PRNG. (Prípad šifry QUAD.) Konštrukcia ponocou binárneho stromu umožňuje zostrojiť generátor pseudonáhodných funkcií F^g z generátora pseudonáhodných čísel g.

Nech g značí generátor pseudonáhodných čísel $g : \mathrm{GF}(q)^n \to \mathrm{GF}(q)^L$, kde $L \geq 2n$. Zo zobrazenia g zostrojíme dve nové zobrazenia g_0 a g_1 definované nasledovne, pre $\boldsymbol{x} \in \mathrm{GF}(q)^n$ platí

$$g_0(\boldsymbol{x}) = (z_1, \dots z_n) \quad \text{a} \quad g_1(\boldsymbol{x}) = (z_{n+1}, \dots z_{2n}),$$

pričom $(z_1, \dots, z_{2n}, \dots, z_L) = g(\boldsymbol{x})$.

Potom funkciu $F^g : \mathrm{GF}(q)^n \times \{0,1\}^m \to \mathrm{GF}(q)^n$ definujeme nasledovne

$$F^g(\boldsymbol{x}, \boldsymbol{y}) = g_{y_m} \circ g_{y_{m-1}} \circ \cdots \circ g_{y_1}(\boldsymbol{x})$$

a platí nasledujúca veta.

Veta 6.11. *Nech $g : \mathrm{GF}(q)^n \to \mathrm{GF}(q)^L$, $L \geq 2n$ je PRNG, ktorý vyprodukuje prvých $2n$ znakov z keystreamu v čase T_g^{2n}. Definujme funckiu $F^g : \mathrm{GF}(q)^n \times \times \{0,1\}^m \to \mathrm{GF}(q)^n$*

$$F^g(\boldsymbol{x}, \boldsymbol{y}) = g_{y_n} \circ g_{y_{n-1}} \circ \cdots \circ g_{y_1}(\boldsymbol{x}),$$

potom platí nasledujúca nerovnosť

$$\mathbf{Adv}_{F^g}^{prf}(t, r) \leq mr\mathbf{Adv}_g^{prng}(t + r(m+1)T_g^{2n}).$$

Dôkaz. Funkcia F^g definuje množinu funkcií

$$\mathcal{F}^g = \{f_y; \boldsymbol{x} \in \mathrm{GF}(q)^n\},$$

pročom platí $f_x(\boldsymbol{y}) = F^g(\boldsymbol{x}, \boldsymbol{y})$. T. j. f_x je funckia z $\{0,1\}^m$ do $\mathrm{GF}(q)^n$.

[5] z bezpečnostných parametrov PRNG odvodíme bezpečnostné parametre zostrojenej PRF

Pre každé $0 \leq i \leq m$ definujeme množinu funkcií $\mathcal{F}_i^g$ z $\{0,1\}^m \to \mathrm{GF}(q)^n$ nasledujúcim spôsobom

$$-\mathcal{F}_0^g = \{f_{x_0}; \boldsymbol{x}_0 \in \mathrm{GF}(q)^n\}\,, \text{ kde}$$
$$f_{x_0}(y_1,\ldots,y_m) = g_{y_m} \circ g_{y_{m-1}} \circ \cdots \circ g_{y_1}(\boldsymbol{x}_0);$$

$$-\mathcal{F}_1^g = \{f_{x_0,x_1}; \boldsymbol{x}_0, \boldsymbol{x}_1 \in \mathrm{GF}(q)^n\}\,, \text{ kde}$$
$$f_{x_0,x_1}(y_1,\ldots,y_m) = g_{y_m} \circ g_{y_{m-1}} \circ \cdots \circ g_{y_2}(\boldsymbol{x}_{y_1});$$

$$-\mathcal{F}_i^g = \left\{f_{x_0,\ldots,x_{2^i-1}}; \boldsymbol{x}_0,\ldots,\boldsymbol{x}_{2^i-1} \in \mathrm{GF}(q)^n\right\}, \text{ kde}$$
$$f_{x_0,\ldots,x_{2^i-1}}(y_1,\ldots,y_m) = g_{y_m} \circ g_{y_{m-1}} \circ \cdots \circ g_{y_{i+1}}(\boldsymbol{x}_{y_1\ldots y_i})$$
$$\text{(pričom výraz } \boldsymbol{x}_{y_1\ldots y_i} \text{ reprezentuje vektor } \boldsymbol{x}_{\sum_{t=1}^{i} y_t 2^{t-1}})\ ;$$

$$-\mathcal{F}_m^g = \left\{f_{x_0,\ldots,x_{2^m-1}}; \boldsymbol{x}_0,\ldots,\boldsymbol{x}_{2^m-1} \in \mathrm{GF}(q)^n\right\}, \text{ kde}$$
$$f_{x_0,\ldots,x_{2^m-1}}(y_1,\ldots,y_m) = \boldsymbol{x}_{y_1\ldots y_n}.$$

Vidíme, že pre takto definované množiny platí, že množina $\mathcal{F}_0^g$ je totožná s množinou $\mathcal{F}^g$ a postupne cez množiny $\mathcal{F}_i^g$ prechádzame k množine $\mathcal{F}_m^g$, ktorá je totožná z množinou náhodných funkcií $\mathcal{F}_{m,n}^*$ z $\{0,1\}^m \to \mathrm{GF}(q)^n$.

Predpokladajme, že existuje algoritmus $\mathbf{A}$, ktorý v čase t rozlišuje náhodne vybranú funkciu z množiny $\mathcal{F}^g$ od náhodne vybranej funckie z množiny $\mathcal{F}_{m,n}^*$ pomocou r orakulovských dotazov s výhodou

$$\left|\Pr_{f\in_R\mathcal{F}^g}(\mathbf{A}(f_y)=1) - \Pr_{f\in_R\mathcal{F}_{n,m}^*}(\mathbf{A}(f)=1)\right| = \varepsilon.$$

Ďalej označme $\Pr_{f\in_R\mathcal{F}_i^g}(\mathbf{A}(f)=1) = p_i$. Z čoho dostávame $\varepsilon = |p_0 - p_n|$. Našim cieľom je pomocou algoritmu $\mathbf{A}$ zostrojiť algoritmus $\mathbf{B}$, ktorý dokáže rozlišiť r výstupov $(g(x_1),\ldots,g(x_r)), x_i \in_R \mathrm{GF}(q)^n$ (prvých $2n$ znakov) z generátora g od r náhodne vybraných vektorov z $\mathrm{GF}(q)^{2n}$.

Algoritmus $\mathbf{B}$, ktorého vstupom je $(\boldsymbol{z_1},\ldots,\boldsymbol{z_r})$, kde $z_i \in \mathrm{GF}(q)^{2n}$ definujme nasledujúcim spôsobom

1. Náhodne vyberie hodnotu $0 \leq i \leq m-1$.
2. Zavolá algoritmus $\mathbf{B_i}$ so vstupom $(\boldsymbol{z_1},\ldots,\boldsymbol{z_r})$.
3. Ak skončí $\mathbf{B_i}$, tak skončí aj $\mathbf{B}$ a vráti výsledok z algoritmu $\mathbf{B_i}$

Algoritmus $\mathbf{B_i}$, ktorého vstupom je $(\boldsymbol{z_1},\ldots,\boldsymbol{z_r})$, definujeme nasledujúcim spôsobom

1. Zavolá algoritmus $\mathbf{A}$, ktorý položí r orakulovských dotazov
$$\boldsymbol{x}^j = (x_1^j,\ldots,x_m^j) \in_R \{0,1\}^m\,.$$

2. Definujme funkciu $\alpha : \{0,1\}^i \to \mathrm{GF}(q)$, ktorá je na začiatku prázdna a budeme ju dodefinovávať postupne v behu algoritmu $\mathbf{B_i}$. Pre každý dotaz $\boldsymbol{x}^j$ algoritmus $\mathbf{B_i}$ zostrojí $\boldsymbol{o}^j \in \mathrm{GF}(q)^n$, ktorý obdrží algoritmus $\mathbf{A}$ ako odpoveď na dotaz $\boldsymbol{x}^j$. Odpoveď $\boldsymbol{o}^j$ je zostrojená nasledujúcim spôsobom.

   - Z prvých i znakov vektoru $\boldsymbol{x}^j$ určí hodnotu $k \in \mathrm{GF}(q)$ ako hodnotu funkcie $\alpha(x_1^j, \ldots, x_i^j)$, ak funkcia α ešte nieje v tomto bode definovaná, dodefinujeme ju náhodnou hodnotou z $\mathrm{GF}(q)$.
   - Z hodnoty x_{i+1}^j definujeme vektor $\boldsymbol{y} \in \mathrm{GF}(q)^n$ takto: Ak $x_{i+1}^j = 0$ bude $\boldsymbol{y}$ rovné prvým n znakov z vektoru $\boldsymbol{z}_k$ a v prípade, že $x_{i+1}^j = 1$ bude $\boldsymbol{y}$ rovný posledným n znakov z vektora $\boldsymbol{z}_k$.
   - Zo zvyšných znakov $x_{i+2}^j, \ldots, x_m^j$ vypočítame $\boldsymbol{o}^j$ ako
   $$\boldsymbol{o}^j = g_{x_m^j} \circ g_{x_{m-1}^j} \circ \cdots \circ g_{x_{i+2}^j}(\boldsymbol{y}).$$

3. Ak skončí algoritmus $\mathbf{A}$ skončí aj algoritmus $\mathbf{B}_i$ a vratí výsledok z algoritmu $\mathbf{A}$.

V prípade, že vstupom algoritmu $\mathbf{B}_i$ je $(g(\boldsymbol{a_1}), \ldots, g(\boldsymbol{a_r}))$, kde vektory $\{0,1\}^m$ platí, že odpoveď, ktorú obdrží algoritmus $\mathbf{A}$ na orakulovský dotaz má rovnakú pravdepodobnosť ako odpoveď z náhodne vybranej funckie z množiny $\mathcal{F}_i^g$

$$\Pr(\mathbf{B_i}(g(\boldsymbol{a_1}), \ldots, g(\boldsymbol{a_r})) = 1) = \Pr_{f \in \mathcal{F}_i^g}(\mathbf{A}(f)) = 1) = p_i.$$

V prípade, že vstupom algoritmu $\mathbf{B_i}$ je r náhodne vybraných vektorov z $\mathrm{GF}(q)^{2m}$ má odpoveď, ktorú obdrží algoritmus $\mathbf{A}$ na orakulovský dotaz rovnakú pravdepodobnosť ako odpoveď z náhodne vybranej funckie z množiny $\mathcal{F}_{i+1}^g$

$$\Pr(\mathbf{B_i}(\boldsymbol{z}_1, \ldots \boldsymbol{z}_r) = 1) = \Pr_{f \in \mathcal{F}_{i+1}^g}(\mathbf{A}(f)) = 1) = p_{i+1}.$$

Z čoho pre algoritmus $\mathbf{B}$ dostávame nasledujúcu rovnosť

$$|\Pr(\mathbf{B}(g(\boldsymbol{a_1}), \ldots, g(\boldsymbol{a_r})) = 1) - \Pr(\mathbf{B}(\boldsymbol{z}_1, \ldots \boldsymbol{z}_r) = 1)| =$$

$$= \left| \frac{1}{m} \sum_{i=0}^{m-1} p_i - \frac{1}{m} \sum_{i=0}^{m} p_i \right| = \frac{1}{m} |p_0 - p_m| = \frac{\varepsilon}{m}.$$

Algoritmus $\mathbf{B}$ pracuje v čase $t + rmT_g^{2n}$ a preto pre lubovoľný algoritmus $\mathbf{A}$ platí nerovnosť

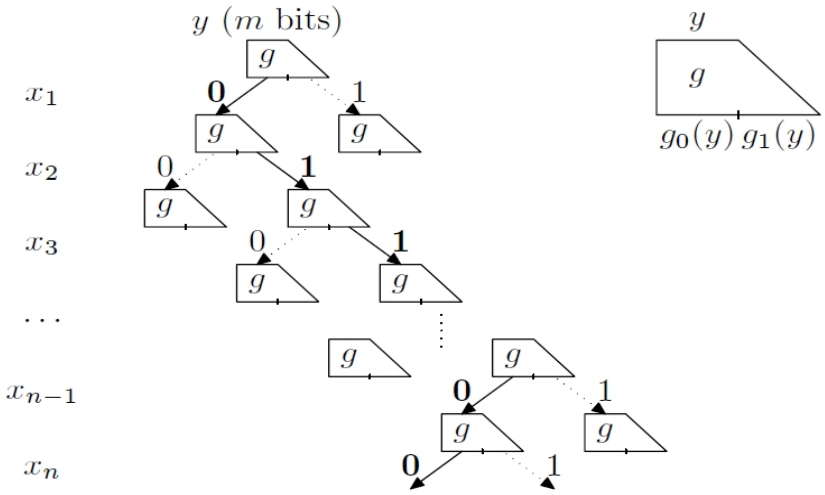
$$\mathbf{Adv}_{F^g}^{\mathrm{prf}}(\mathbf{A}) \leq \mathbf{Adv}_g^{\mathrm{prng}}(\mathbf{B}) \leq m\mathbf{Adv}_g^{\mathrm{prng}}(t + rmT_g^{2n}, r).$$

Nakoniec použitím lemy 6.8 dostávame

$$\mathbf{Adv}_{F^g}^{\mathrm{prf}}(t, r) \leq mr\mathbf{Adv}_g^{\mathrm{prng}}(t + r(m+1)T_g^{2n}).$$

□

Obrázok 3. znázorňuje túto konštrukciu ako binárny strom, kde inicializačný vektor určuje cestu v tomto strome.



Obr. 3: Výpočet funkcie $F^g(\boldsymbol{x}, \boldsymbol{y})$

6.2.4 Konštrukcia bezpečnej prúdovej šifry

V predošlej časti sme ukázali ako zostrojiť z generátora pseudonáhodných čísel g generátor pseudonáhodných funkcií F^g pomocou stromovej konštrukcie a ukázali sme, že bezpečnosť takto zostrojeného generátora pseudonáhodných funkcií F^g závisi len od bezpečnosti generátora g. Ku konštrukcii prúdovej šifry s `IV` budeme potrebovať dva generátory pseudonáhodných čísel

1. $g : \mathrm{GF}(q)^m \to \mathrm{GF}(q)^L$, ktorý bude slúžiť ako samotný generátor keystreamu z inicializačnej hodnoty $\boldsymbol{x}_0$;

2. $g^{'} : \mathrm{GF}(q)^m \to \mathrm{GF}(q)^{2m}$ z ktorého pomocou stromovej konštrukcie (konštrukcia pomocou binárneho stromu) zostrojíme generátor $F^{g^{'}}$, ktorý z $(\boldsymbol{K}, \mathtt{IV})$ vygeneruje inicializačnú hodnotu $\boldsymbol{x}_0$ z .

A teda prúdovú šifru (generátor keystreamu G) môžeme prezentovať, ako zloženú funkciu $G = g \circ F^{g^{'}}$. Priamou aplikáciou vety 6.9 a vety 6.11 dostávame nasledujúce tvrdenie o bezpečnosti takto zostojenej prúdovej šifry.

Veta 6.12. *Nech $g : \mathrm{GF}(q)^n \to \mathrm{GF}(q)^L$ je PRNG, ktorý vygeneruje L znakov za čas T_g^L a $g' : \mathrm{GF}(q)^n \to \mathrm{GF}(q)^{2n}$ je PRNG. Definujme prúdovú šifru $G = g \circ F^{g'}$, kde $F^{g'}(\boldsymbol{x}, \boldsymbol{y}) = g'_{y_m} \circ g'_{y_{m-1}} \circ \cdots \circ g'_{y_1}(\boldsymbol{x})$. (Funkcia zostrojená pomocou stromovej konštrukcie.) Potom funkcia G je PRF a platí*

$$\mathbf{Adv}_G^{prf}(t, r) \leq mr\mathbf{Adv}_{g'}^{prng}(t + rT_g^L + r(m+1)T_{g'}^{2n}) + r\mathbf{Adv}_g^{prng}(t + rT_g^L).$$

V prípade, že $g = g'$ dostávame

$$\mathbf{Adv}_G^{prf}(t, r) \leq r(m+1)\mathbf{Adv}_g^{prng}(t + r(m+2)T_g^L).$$

6.2.5 Aplikácia na šifru QUAD

V tejto podkapitole aplikujeme predchádzajúcu teóriu o bezpečnej prúdovej šifre s IV na šifru QUAD. Ukážeme, že prúdová šifra QUAD je generátorom pseudonáhodných funkcií. V úvode si pripomenieme ako funguje prúdová šifra šifra QUAD.

Nech $\boldsymbol{S}$ a $\boldsymbol{S'}$ značí náhodne vygenerovaný kvadratický systém n premenných nad telesom GF(q), kde

$$\boldsymbol{S} = (\underbrace{Q_1, Q_2, \ldots, Q_n}_{\boldsymbol{S_{\text{int}}}}, \underbrace{Q_{n+1}, \ldots Q_{kn}}_{\boldsymbol{S_{\text{out}}}}) = (\boldsymbol{S_{\text{int}}}, \boldsymbol{S_{\text{out}}})$$

$$\boldsymbol{S'} = (\underbrace{P_1, P_2, \ldots, P_n}_{\boldsymbol{S'_0}}, \underbrace{P_{n+1}, \ldots P_{2n}}_{\boldsymbol{S'_1}}) = \left(\boldsymbol{S'_0}, \boldsymbol{S'_1}\right)$$

V **1. kroku** (algoritmus 2) vygenerujeme z kľúča $\boldsymbol{K} \in \text{GF}(q)^n$ a inicializačného vektora $\boldsymbol{IV} \in \{0,1\}^m$ hodnotu $\boldsymbol{x_0}' \in \text{GF}(q)^n$. Jedná sa práve o konštrukciu pomocu binárneho stromu definovanú v oddiely (6.2.3) aplikovanej na generátor $\boldsymbol{S'}$

$$F^{S'}(\boldsymbol{K}, \texttt{IV}) = S'_{\texttt{IV}_n} \circ S'_{\texttt{IV}_{n-1}} \circ \cdots \circ S'_{\texttt{IV}_1}(\boldsymbol{K}) = \boldsymbol{x_0}'.$$

(Na $\boldsymbol{S'}$ sa pozeráme ako na PRNG z GF(q)^n do GF(q)^{2n}.) Na záver hodnotu $\boldsymbol{x_0}'$ ešte $m = |\texttt{IV}|$ - krát prepíšeme na hodnotu $\boldsymbol{S_{\text{int}}}(\boldsymbol{x_0}')$. A tým získame finálnu inicializačnú hodnotu $\boldsymbol{x_0} = \boldsymbol{S_{\text{int}}}^m(\boldsymbol{x_0}')$.

V **2. kroku** (algoritmus 1) z inicializačnej hodnoty $\boldsymbol{x_0}$ vygenerujeme samotný keystream o dĺžke $L = \lambda(k-1)n$

$$\boldsymbol{S_{\text{out}}}(\boldsymbol{x_0})|| \underbrace{\boldsymbol{S_{\text{out}}}(\boldsymbol{S_{\text{int}}}(\boldsymbol{x_0}))}_{n(k-1) \text{ znakov z GF}(q)} ||\cdots||\boldsymbol{S_{\text{out}}}(\boldsymbol{S_{\text{int}}}^{\lambda}(\boldsymbol{x_0})),$$

čo je ekvivaletné zápisu (zároveň takto definujeme generátor g_{real})

$$g_{\text{real}}(\boldsymbol{x_0}') = \boldsymbol{S_{\text{out}}}(\boldsymbol{S_{\text{int}}}^m(\boldsymbol{x_0}'))||\boldsymbol{S_{\text{out}}}(\boldsymbol{S_{\text{int}}}^{m+1}(\boldsymbol{x_0}'))||\cdots||\boldsymbol{S_{\text{out}}}(\boldsymbol{S_{\text{int}}}^{\lambda+m}(\boldsymbol{x_0}')).$$

Šifra QUAD je zobrazenie z GF(q)$^n \times \{0,1\}^m$ do telesa GF(q)^L definované ako

$$\text{QUAD}(\boldsymbol{K}, \texttt{IV}) = g_{\text{real}} \circ F^{S'}(\boldsymbol{K}, \texttt{IV}).$$

Z vety 6.12 aplikovanej na zložené zobrazenie QUAD, dostávame nasledujúci odhad

$$\begin{aligned}\mathbf{Adv}^{\text{prf}}_{\text{QUAD}}(T, r) \leq mr\mathbf{Adv}^{\text{prng}}_{S'}(T + rT^L_{g_{\text{real}}} + r(m+1)T_{S'}) + \\ + r\mathbf{Adv}^{\text{prng}}_{g_{\text{real}}}(T + rT^L_{g_{\text{real}}}).\end{aligned}$$

Kvadratický systém $\boldsymbol{S'}$ je PRNG, ktorý z $\boldsymbol{x} \in \text{GF}(q)^n$ vygeneruje postupnosť (keystream) znakov z telesa GF(q) dĺžky $2n$ rovný $\boldsymbol{S'}(\boldsymbol{x})$.

Ak predpokladame, že $k \geq 2$, t. j. náhodne vybraný kvadratický systém $\boldsymbol{S}$ obsahuje aspoň toľko polynómov ako $\boldsymbol{S'}$, platí, že

$$\mathbf{Adv}_{S'}^{\mathrm{prng}}(T) \leq \mathbf{Adv}_{S}^{\mathrm{prng}}(T). \tag{6.1}$$

Pretože, ak existuje algoritmus $\mathbf{A}$, ktorý rozlišuje výstup z kvadratického systému $2n$ polynómov n premenných $\boldsymbol{S'}(\boldsymbol{x})$ od náhodnej $2n$ znakovej postupnosti z $\mathrm{GF}(q)$ v čase T, potom pomocou algoritmu $\mathbf{A}$ dokážeme zostrojiť algoritmus $\mathbf{B}$, ktorý rozlíši výstup $\boldsymbol{S}(\boldsymbol{x})$, t. j. „väčšieho“ kvadratického systému kn polynómov n premenných, od náhodnej kn znakovej postupnosti. Algoritmus $\mathbf{B}$ môžeme definovať nasledujúcim spôsobom

$$\mathbf{B}(y_1, \ldots, y_{2n}, \ldots, y_{kn}) = \mathbf{A}(y_1, \ldots, y_{2n}).$$

Pre takto definovaný algoritmus $\mathbf{B}$ dostávame

$$\mathbf{Adv}_{S'}^{\mathrm{prng}}(\mathbf{A}) = \mathbf{Adv}_{S}^{\mathrm{prng}}(\mathbf{B}) \leq \mathbf{Adv}_{S}^{\mathrm{prng}}(T),$$

kedže táto nerovnosť, platí pre každé $\mathbf{A}$, dostávame uvedenú nerovnosť (6.1). Pre prípad $1 < k < 2$, t. j. kvadratický systém $\boldsymbol{S'}$ obsahuje aspoň toľko polynómov ako $\boldsymbol{S}$, platí nerovnosť $\mathbf{Adv}_{S'}^{\mathrm{prng}}(T) \geq \mathbf{Adv}_{S}^{\mathrm{prng}}(T)$.

Definujme generátor pseudonáhodných čísel $\widetilde{g}^S$ ako zobrazenie

$$\widetilde{g}^S(\boldsymbol{x}) = \underbrace{\boldsymbol{S}_{\mathbf{out}}(\boldsymbol{x})||\cdots||\boldsymbol{S}_{\mathbf{out}}({\boldsymbol{S}_{\mathbf{int}}}^{m}(\boldsymbol{x})))||\cdots||\boldsymbol{S}_{\mathbf{out}}({\boldsymbol{S}_{\mathbf{int}}}^{m+\lambda}(\boldsymbol{x}))}_{m(k-1)n+L=(\lambda+m)(k-1)n \text{ znaková postupnosť}}.$$

Vidíme, že ak vynecháme prvých $n(k-1)m$ znakov z keystreamu vygenerovaného pomocou $\widetilde{g}^S$ dostaneme rovnaký keystream aký získame pomocou generátora g_{real}. Ak by existoval algoritmus, ktorý rozlíši výstup z generátora g_{real}, tak by tento algoritmus rozlišoval aj výstup z generátora $\widetilde{g}^S$, čo by sme zdvôvodnili podobne ako v prípade nerovnosti (6.1). A teda platí

$$\mathbf{Adv}_{g_{real}}^{\mathrm{prng}}(T) \leq \mathbf{Adv}_{\widetilde{g}^S}^{\mathrm{prng}}(T). \tag{6.2}$$

Takže podľa definície generátora $\widetilde{g}^S$ vidíme, že keystream $\widetilde{g}^S(\boldsymbol{x})$, dostaneme vykonaním $(\lambda + m)$ prechodov konštruckie šifry QUAD asociovanej s kvadratickým systémom $\boldsymbol{S}$ (algoritmus 1). Aplikáciou vety 6.1 na generátor $\widetilde{g}^S$, ktorý vyprodukuje keystream dĺžky $L = (\lambda + m)(k-1)n$ dostávame, že

$$\mathbf{Adv}_{\widetilde{g}^S}^{\mathrm{prng}}(T) \leq (\lambda + m)\mathbf{Adv}_{S}^{\mathrm{prng}}(T + (\lambda + m)T_{\boldsymbol{S}}). \tag{6.3}$$

Použitím horných odhadov z (6.1), (6.2) a (6.3) a lemy 2.6 dostávame po úprave

$$\begin{aligned}
\mathbf{Adv}_{\mathrm{QUAD}}^{\mathrm{prf}}(T, r) &\leq mr\mathbf{Adv}_{S}^{\mathrm{prng}}(T + rT_{g_{\mathrm{real}}}^{L} + r(m+1)T_{\boldsymbol{S}}) + \\
&+ r(\lambda + m)\mathbf{Adv}_{S}^{\mathrm{prng}}(T + rT_{g_{\mathrm{real}}}^{L} + (\lambda + m)T_{\boldsymbol{S}}) \leq \\
&\leq (2m + \lambda)r\mathbf{Adv}_{S}^{\mathrm{prng}}(T + rT_{g_{\mathrm{real}}}^{L} + r(m+1)T_{\boldsymbol{S}}) = \\
&= (2m + \lambda)r\mathbf{Adv}_{S}^{\mathrm{prng}}(T + r(\lambda + m)T_{\boldsymbol{S}} + r(m+1)T_{\boldsymbol{S}})
\end{aligned}$$

Tento odhad nám hovorí, že ak existuje algoritmus **A**, ktorý v čase T, pomocou r orakulovských dotazov, rozlíši funkciu QUADod náhodnej funckie s výhodou ε, potom existuje algoritmus **B**, ktorý rozlíši $\boldsymbol{S}(\boldsymbol{x})$ od náhodnej postupnosti dĺžky kn v čase $T' = T + r(\lambda + 2m + 1)T_{\boldsymbol{S}}$ s výhodou $\varepsilon/(2m+\lambda)r$.

Ak by existoval algoritmus **B**, ktorý v čase T' s výhodou $\varepsilon/(2m+\lambda)r$ rozlišuje $\boldsymbol{S}(\boldsymbol{x})$, kde $\boldsymbol{x} \in_R \mathrm{GF}(q)^n$ je neznámy náhodne vybraný vektor, od náhodnej kn znakovej postupnosti z $\mathrm{GF}(q)$, tak z vety 6.2 a vety 6.6 dostávame existenciu algoritmu **D**, ktorý dokáže invertovať zobrazenie $\boldsymbol{S}(\boldsymbol{x})$ s pravedepodobnosťou $\varepsilon/4qr(2m+\lambda)$ v čase T'', kde

$$T'' \leq 2^{15}\frac{nq^6r^5(2m+\lambda)^5}{\varepsilon^5}\log^2\left(\frac{2qnr(2m+\lambda)}{\varepsilon}\right)\left(T' + 2T_{\boldsymbol{S}}\right) +$$

$$+\left(1-\frac{1}{q}\right)^2\frac{4q^2r^2(2m+\lambda)^2}{\varepsilon^2}T_{\boldsymbol{S}}.$$

Toto tvrdenie zhrnieme do nasledujúcej vety.

Veta 6.13. *Nech algoritmus* **A** *pomocou r orakulovských dotazov rozlišuje funkciu* $\mathrm{QUAD}(k,n,q)(\boldsymbol{K},\cdot) : \{0,1\}^m \to \mathrm{GF}(q)^{L=n(k-1)\lambda}$, *kde $\boldsymbol{K}$ je neznámy náhodne vybranný kľúč, od náhodnej funkcie $f : \{0,1\}^m \to \mathrm{GF}(q)^L$, v čase T s výhodou ε. Potom existuje algoritmus* **B**, *ktorý zo vstupu $(\boldsymbol{S}, \boldsymbol{S}(\boldsymbol{x}))$, kde $\boldsymbol{x} \in_R \mathrm{GF}(q)^n$ je neznámy náhodne vybraný vektor, nájde $\boldsymbol{x}$ s pravdepodobnosťou ε' v čase T', kde*

$$\varepsilon' \geq \frac{\varepsilon}{4qr(2m+\lambda)},$$

$$T' \leq 2^{15}\frac{nq^6r^5(2m+\lambda)^5}{\varepsilon^5}\log^2\left(\frac{2qnr(2m+\lambda)}{\varepsilon}\right)\left(T + r\Big(\lambda+2m+1\Big)T_S + 2T_S\right) +$$

$$+\left(1-\frac{1}{q}\right)^2\frac{4q^2r^2(2m+\lambda)^2}{\varepsilon^2}T_{\boldsymbol{S}}.$$

Pre prípad telesa $\mathrm{GF}(2)$ *dostávame*

$$\varepsilon' \geq \frac{\varepsilon}{8r(2m+\lambda)};$$

$$T' \leq \frac{2^7n^2r^2(\lambda+2m)^2}{\varepsilon^2}\Bigg(T + (r(\lambda+2m+1)+2)\,T_{\boldsymbol{S}} +$$

$$+\log\left(\frac{2^7nr^2(\lambda+2m)^2}{\varepsilon^2}\right) + 2\Bigg) + \frac{2^7nr^2(\lambda+2m)^2}{\varepsilon^2}T_{\boldsymbol{S}}.$$

7 Klasifikácia šifry QUAD podľa hodnôt parametrov

V tejto kapitole rozdelíme šifru QUAD, podľa parametrov pre ktoré je šifra QUAD prelomiteľná, nedokázatená alebo (ne)preukazateľne bezpečná.

7.1 Rozdelenie paramaterov

- Šifru QUAD s parametrami pre, ktoré útočník dokáže vypočítať v prijateľnom čase ($< 2^{80}$) tajnú inicializačnú hodnotu $\boldsymbol{x}$, bez ohľadu na to ako bola vygenerovaná, už po niekoľkých prechodov šifry QUAD, nazveme **prelomiteľná**.
- Parametre pre, ktoré existuje uskutočniteľný ($< 2^{80}$) útok na príslušný MQ problém, na ktorom je postavená šifra QUAD. Šifra QUAD, pre tieto parametre nemôže byť nikdy preukazateľne bezpečná, aj keby redukcia šifry QUAD na MQ problém bola priliehavejšia. Šifru QUAD pre tieto parametre nazveme **nedokazateľná**.
- Šifru QUAD s parametrami, pre ktoré pomocou vety 6.5 nedokážeme zaručiť bezpečnosť šifry. To samozrejme neznamená, že šifra QUAD je pre tieto parametre prelomiteľná. Je možne, že ak by redukcia šifry QUAD na MQ bola priliehavejšia, pre mnohé z týchto parametrov by sme bezpečnosť už vedeli zaručiť. Tieto parametre nazveme **nepreukazateľne bezpečná** .
- Šifru QUAD s parametrami, pre ktoré bezpečnosť šifry QUAD dokážeme zaručiť pomocou vety 6.5, nazveme **preukazateľne bezpečná**.

7.2 QUAD s prelomiteľnou bezpečnosťou

Bez ohľadu na to, ako bol inicializačný vektor $\boldsymbol{x_0}$ vygenerovaný. Keystream šifry QUAD sa rovná

$$\boldsymbol{S_{\text{out}}}(\boldsymbol{x_0})|| \underbrace{\boldsymbol{S_{\text{out}}}(\boldsymbol{S_{\text{int}}}(\boldsymbol{x_0}))}_{n(k-1)\text{ znakov z GF}(q)} ||\cdots||\boldsymbol{S_{\text{out}}}(\boldsymbol{S_{\text{int}}}^{\lambda}(\boldsymbol{x_0})).$$

Mohli by sme sa pokúsiť vyriešiť rovno $\boldsymbol{S_{\text{out}}}(\boldsymbol{x_0}) = \boldsymbol{y_1}$, kde $\boldsymbol{y_1}$ značí prvých $n(k-1)$ znakov keystreamu. Ale, už aj pre malý systém s 20 kvadratickými rovnicami 20 premenných by to trvalo pomocou algoritmu XL (Wiedemann) viac ako 2^{80} cyklov. Skúsime namiesto jednej rovnici riešiť dve rovnice

$$\begin{aligned}\boldsymbol{S_{\text{out}}}(\boldsymbol{x_0}) &= \boldsymbol{y_1}\\ \boldsymbol{S_{\text{out}}}(\boldsymbol{S_{\text{int}}}(\boldsymbol{x_0})) &= \boldsymbol{y_2}.\end{aligned}$$

Útočník má k dispozícii $\boldsymbol{S_{\text{out}}}$ a $\boldsymbol{S_{\text{int}}}$, pretože sú verejné. Ďalej predpokladajme, že útočník získal prvých $2 \times n(k-1)$ znakov keystreamu $\boldsymbol{y_1}, \boldsymbol{y_2}$. Tieto hodnoty

mohol získať napríklad po úspešnom uhádnutí začiatku otvorenej správy. Rovnica $\boldsymbol{S}_{\mathbf{out}}(\boldsymbol{x}_{\mathbf{0}}) = \boldsymbol{y}_{\mathbf{1}}$ pozostáva z $(k-1)n$ kvadratických rovníc n premenných. Rovnica $\boldsymbol{S}_{\mathbf{out}}(\boldsymbol{S}_{\mathbf{int}}(\boldsymbol{x}_{\mathbf{0}})) = \boldsymbol{y}_{\mathbf{2}}$ pozostáva z $(k-1)n$ kvartických rovníc n premenných. Utočník po vyriešení tohoto kvarticko-kvadratického systému získa inicializačnú hodnotu $\boldsymbol{x}_{\mathbf{0}}$, pomocou ktorej dokáže vypočítať celý zvyšný keystream $\boldsymbol{y}_{\mathbf{3}}, \boldsymbol{y}_{\mathbf{4}}, \ldots$. Algoritmus XL definovaný v tejto práci, pracuje nad systémom kvadratických rovníc. Tento algoritmus sa da zovšeobecniť aby pracoval nad systémom polynomiálnych rovníc $l_1, l_2, \ldots l_m$, kde stupeň jednotlivých polynómov $\deg(l_i)$ môže byť rôzny od 2. A to nasledujúcou modifikáciou 1. kroku z definície algoritmu XL.

1. *(e**X**tend): Pre každý monóm stupňa $\leq D - d_{max}$, $\boldsymbol{x}^{\boldsymbol{b}} \in \mathcal{T}^{D-d_{max}}$, vygenerujeme množinu rovníc $\boldsymbol{x}^{\boldsymbol{b}} l_i(\boldsymbol{x}) \in \mathcal{T}^D$, kde $1 \leq i \leq m$. Systém všetkých týchto rovníc, označíme $\mathcal{R} = \mathcal{R}^D$.*

K odhadu najmenšieho D, pre ktoré zovšeobecnená verzia algoritmu XL funguje, nám poslúži nasledujúca veta.

Veta 7.1. *Pre semiregulárny systém polynómov, kde $D \leq \min(q, D_{XL}^{\infty})$ platí*

$$T - I = [t^D]\left((1-t)^{-n-1}\prod_{j=1}^{m}(1-t^{d_i})\right),$$

Pričom $D_{XL}^{\infty} = \min\left(D:\ [t^D]G(t) \leq 0\right)$, kde $G(t) = \left((1-t)^{-n-1}\prod_{j=1}^{m}(1-t^{d_i})\right)$.

Hodnota I sa môže jedine znížiť, ak systém nie je semiregulárny. Nech D_0 značí najmenšie D, pre ktoré XL funguje. Predpokladá sa [21], že náhodne vybraný polynomiálny systém je s veľkou pravdepodobnostou semiregulárny a teda platí,

$$D_0 = \min\left\{D : [t^D]G(t) \leq \min(D, q-1)\right\}.$$

S veľkou pravdepodobnosťou platí, že $D_{\mathrm{XL}}^{\infty} = D_0$.

Poznámka 7.2. Parameter $D_0 = D_{\mathrm{XL}}^{\infty}$, pre náhodne vybraný kvadratický systém nad končným telesom $\mathrm{GF}(q)$, je stupeň prvého nekladného koeficientu v rade

$$\left((1-t)^{-n-1}\prod_{j=1}^{m}(1-t^{d_i})\right\},$$

čo vyplýva priamo z definície D_{XL}^{∞}.

Tvrdenie 7.3. *Pre $q > 10$, semi-regulárny náhodne vybraný sytém 20 kvadratických a 20 kvartických rovníc 20 premenných nad konečným telesom $\mathrm{GF}(q)$ je vyriešiteľný pomocou $2^{63}\mathfrak{m}$ (násobení v telese $\mathrm{GF}(q)$).*

Dôkaz. Podľa vety 7.1 je

$$D_{\mathrm{XL}}^{\infty} = \min\left(D:\ [t^D]\frac{(1-t^2)^{20}(1-t^4)^{20}}{(1-t)^{21}} \leq 0\right) = 10.$$

Parameter algoritmu XL je $D_{\mathrm{XL}} = 10$. Ďalej $T = \binom{n+D}{D} = \binom{30}{10} = 2^{25}$ a $R = 20 \times \times \binom{28}{8} + 20 \times \binom{26}{6} = 2^{36}$. Po linearizácii získame systém R lineárnych rovníc v T neznámych. Celkový počet členov v sústave $\mathcal{R}$ je $\tau = \binom{22}{2}20\binom{28}{8} + \binom{24}{4}20\binom{26}{6} = 2^{36}$. Z tvdenia 4.12 dostávame

$$C_{\mathrm{XL}} \approx 3\tau T\,\mathfrak{m} \lesssim 2^{63}\mathfrak{m}.$$

□

Príklad 7.4. Napríklad QUAD$(2, 40, 16)$ nie je prelomiteľný pomocou tohoto útoku. V tomto prípade prvý nekladný koeficient v $(1-t^2)^{40}(1-t^4)^{40}(1-t)^{-41}$ dostávame pre $D_{\mathrm{XL}} = 14$. Takže pre konečne teleso GF(q), kde $q > 14$ vyriešenie semiregulárneho systému 40 kvadratických a 40 kvartických rovníc nad telsom GF(q) zaberie pomocou XL (Wiedemann) $\lesssim 2^{95}$ násobení v telese GF(q). Čo znamená, že bezpečnosť šifry QUAD$(2, 40, q)$, pre $q > 14$, je značne pod 128-bitovou úrovňou zabezpečenia, ale zatiaľ nepoznáme žiaden útok pod 80-bitovú úrovňou zabezpečenia.

V tvrdení 7.3 a v príklade 7.4 sme používali predpoklad: *Pre náhodný systém* $\boldsymbol{S}_{\mathbf{out}}$ *a* $\boldsymbol{S}_{\mathbf{int}}$ *je systém rovníc* $\boldsymbol{S}_{\mathbf{out}}(\boldsymbol{x}_{\mathbf{0}}) = \boldsymbol{y}_{\mathbf{1}}$ *a* $\boldsymbol{S}_{\mathbf{out}}(\boldsymbol{S}_{\mathbf{int}}(\boldsymbol{x}_{\mathbf{0}})) = \boldsymbol{y}_{\mathbf{2}}$ *semiregulárny.*

7.3 QUAD s nedokázateľnou bezpečnosťou

V tejto časti si uvedieme príklad na šifru QUAD s parametrami o ktorej nikdy nemôže byť dokázaná preukázateľná bezpečnosť.

Vo vete sme dokázali, že ak dokážeme prelomiť šifru QUAD v čase T, potom dokážeme vyriešiť MQ problém v čase $T' = LT$, kde $L \geq 1$. Snaha je vždy navrhnuť, čo najpriliehavejšiu redukciu, t. j. hodnotu L čo najviac priblížiť k 1.

Pre QUAD s parametrami, pre ktoré príslušný MQ problém nie je ťažký ($\leq 2^{80}$), nie je možné ani v prípade dokonale priliehavej redukcii ($L = 1$) o tejto šifre dokázať preukzateľnú bezpečnosť. Uveďme dva príklady.

Príklad 7.5 (QUAD$(2, 20, 256)$)**.** V tomto prípade bezpečnosť šifry stojí na obťažnosti riešenia 40 kvadratických rovníc 20 premenných nad telesom GF(256). Tento problém je možné v čase $\leq 2^{80}$ vyriešiť a to pomocou algoritmu XL (Wiedemann). Pre algoritmus XL dostávame z vety (4.8) $D_{\mathrm{XL}} = 5$

$$T = \binom{25}{5}; \quad \tau = 20\binom{22}{2}\binom{23}{3}$$

Na vyriešenie 40 kvadratických rovníc 20 premenných je podľa vety 4.12 potrebných $C_{\mathrm{XL}} \approx 3\tau T\,\mathfrak{m} \lesssim 2^{40}$ násobení v telese GF(256).

Príklad 7.6 (QUAD$(2, 40, 16)$)**.** Pre systém 80 kvadratických rovníc 40 premenných dostávame

$$D_{\mathrm{XL}} = 8; \quad T = \binom{48}{8}; \quad \tau = 80\binom{42}{2}\binom{46}{6}$$

Z čoho dostávame, že $C_{\mathrm{XL}} \lesssim 2^{69}$ násobení v GF(16).

Pri porovnaní týchto príkladov s tvrdením 7.3 a príkladom 7.4 vidíme roziel medzi útokom na šifru QUAD a útokom na príslušný MQ problém. Ukázať, nedokázateľnosť je ľahšie ako ukázať prelomiteľnosť

7.4 QUAD s (ne)preukázateľnou bezpečnosťou

Najprv budeme uvažovať šifru QUAD len ako PRNG, t. j. bez generovania inicializačnej hodnoty z $\boldsymbol{K}$ a IV. Ukážeme, že pre QUAD$(2, n, 2)$, pre $n > 350$ by existencia algoritmu (útočníka), ktorý by v čase lepšom ako 2^{80} s výhodou $0,01$ rozlíšil 2^{40}-bitový keystream od náhodnej postupnosti by nám dávalo existenciu (z vety 6.5) algoritmu na riešenie kvadratických rovníc v čase lepšom ako všetky doteraz známe algoritmy na riešenie takýchto sústav.

V oddiely 7.4.2 budeme uvažovať šifru QUAD vrátane generovania inicializačnej hodnoty. Ukážeme, že pre QUAD$(2, n, 2)$, kde $n > 530$ dokážeme zaručit 80-bitovú úroveň zabezpečenia.

7.4.1 Parametre pre generátor keystreamu šifry QUAD

Ukážeme, že pre určité nastavenia parametrov k, n, q a L je prúdová šifra QUAD, ktorá vygeneruje z neznámej náhodne vybranej inicializčnej hodnoty $\boldsymbol{x} \in \mathrm{GF}(q)$, keystream dĺžky L, preukázateľne bezpečná prúdová šifra. Inač povedané generátor keystreamu šifry QUAD je bezpečný PRNG a hodnota

$$\mathbf{Adv}^{\mathrm{prng}}_{\mathrm{QUAD}}(T)$$

nám garantuje dostatočnú odolnosť proti prípadným útokom. V dôkaze budeme postupovaž sporom. Ukážeme, že ak by existoval algoritmus $\mathbf{A}$, ktorý v čase T s výhodou ε rozlišuje keystream vygenerovaný dĺžky L vygenerovaný šifrou QUAD(k, n, q) od náhodne vybranej postupnosti z GF(q) dĺžky L, tak z vety 6.7 prípadne z vety 6.5 dostávame algoritmus $\mathbf{D}$, ktorý rieši kvadratický systém kn polynómov o n premenných nad telesom GF(q) v čase lepšom ako ako všetky doteraz známe algoritmy na riešenie kvadratických systémov kn polynómov o n premenných nad telesom GF(q).

Pre zjednodušenie sa obmedzíme na prípad telesa GF(2). Parametre šifry QUAD zvolíme tak, aby sme pomocou vety 6.5, zaručili bezpečnostný stupeň šifry QUAD, minimálne $T = 2^{80}$ a $\varepsilon = 1/100$. Čo znamená, že požadujem, aby neexistoval algoritmus, ktorý by v čase lepšom ako 2^{80} s výhodou $1/100$ rozlišoval keystream vygenerovaný šifrou QUAD$(k, n, 2)$ od náhodne vybranej postupnosti z GF$(2)^L$.

Maximálna dĺžka keystreamu L, sa môže pohybovať od 2^{10} až po 2^{40}, v závislosti od použitia tejto šifry. Zvoľme $L = 2^{40}$ a $k = 2$ pre túto hodnotu dopočítame hodnotu parametru n.
Ak by existoval algoritmus, ktorý rozlíši 2^{40}-bitový kystream vygenerovaný šifrou QUAD z náhodne v vybranej inicializačnej hodnoty $\boldsymbol{x_0}$, od náhodne vybranej 2^{40}-bitovej postupnosti v čase lepšom ako T a výhodou ε, dostávame z vety 6.5 existenciu algoritmu $\mathbf{D}$, ktorý v čase T' dokáže riešiť MQ problém s pravdepodobnosťou úspechu najemnej ε'.
Algoritmus $\mathbf{D}$ je pravdepodobnostný algoritmus na riešenie kvadratických systémov kn polynómov n premenných, pre $k > 1$. Preto algoritmu $\mathbf{D}$ opakujeme, kým nezískame správnu hodnotu $\boldsymbol{x}$. Očakávaný počet opakovaní je $1/\varepsilon'$. Algoritmus $\mathbf{D}$, preto v čase $T_{\mathbf{D}} = T'/\varepsilon'$ rieši systém kn kvadratických rovníc n premenných. Pre získanie sporu chceme ukázať, že algoritmus $\mathbf{D}$ pracuje efektívnejšie ako najrýchlejší doteraz známy algoritmus, XL, na riešenie kvadratického systému, t. j. aby platilo

$$C_{XL} > T_{\mathbf{D}} = \frac{T'}{\varepsilon'}.$$

Stanovíme 80-bitovú úroveň zabezpečenia šifry QUAD a teda $T = 2^{80}$ a $\varepsilon = 1/100$. Ďalej predpokladajme, že existuje algoritmus, ktorý v čase $T \leq 2^{80}$ rozlíši keystream dĺžky 2^{40} vygenerovaný šifrou $\textsc{quad}(2, n, 2)$ z náhodne vybranej inicializačnej hodnoty $\boldsymbol{x_0}$, od náhodne 2^{40}-bitovej postupnosti s výhodou $\varepsilon \geq 0,01$. Z vety 6.5 dostávame algoritmus $\mathbf{D}$, ktorý v čase $T' \leq 2^{230}/n$ vyrieši systém $2n$ kvadratických rovníc o n neznámych.
Spor získame pre $n > 350$. Pre $n > 350$ je $T_{\mathbf{D}} \leq 2^{222}$. Časová zložitosť algoritmus XL (Wiedemann) na nájdenie riešenia pre 700 kvadratických rovníc o 350 neznámych nad telesom GF(2) čase je $C_{\mathrm{XL}} \lesssim 2^{263}$ násobení v telese GF(2), kde z vety 4.8 je $D_{\mathrm{XL}} = 19$.
Pre hodnoty $k = 2$ a $n = 256$ získame spor pre $L \leq 2^{22}$.

7.4.2 Parametre pre šifru QUAD s `IV`

Uvažujeme prúdovú šifru QUAD vrátane generovania inicializačnej hodnoty $\boldsymbol{x_0}$ z tajného kľúča $\boldsymbol{K}$ a inicializačného vektoru `IV`.

Obmedzíme sa na prípad telesa GF(2). Položme $L = 2^{40}$, $T = 2^{80}$, $\varepsilon = 0,01$ a $r = 2^{40}$. Dĺžka inicializačného vektora `IV` sa rovná n. (Pre praktické použitie je inicializačný vektor 80 až 128 bitový vektor.) Predpokladajme, že existuje $\mathbf{A}$, ktorý pomocou 2^{40} orakulovských prístupov rozlíši (pseudonáhodnú funkciu) šifru QUAD od náhodnej funkcie v čase 2^{80} s výhodou $0,01$. Potom z vety 6.13 pravdepodobnostný algoritmus $\mathbf{D}$, ktorý s pravdepodobnosťou ε' v čase T' rieši systém kvadratických systémov kn polynómov n premenných, pre $k > 1$. Pre získanie sporu požadujeme aby platilo

$$C_{XL} > \frac{T'}{\varepsilon'} = T_{\mathbf{D}}.$$

Pre vyššie stanovené hodnoty a pre $k = 2$ získame z vety 6.13 a zo zložitosti algoritmu XL, spor pre $n > 530$. Pre hodnoty $k = 2$ a $n = 512$ získame spor pre $L \leq 2^{35}$.

8 Záver

V tejto práci sme predstavili novú synchrónnu prúdovú šifru QUAD. Ukázali sme dôkaz bezpečnosti tejto šifry, v konkrétnom bezpečnostnom modely.

K návrhu šifry QUAD, konkrétne ku generátoru inicializačnej hodnoty, by som uviedol nasledujúcu poznámku. Význam kroku, kedy hodnotu získanú z pseudonáhodnej funckie $\boldsymbol{x_0}' = F^{S'}(\boldsymbol{K}, \mathtt{IV})$ prepíšeme na hodnotu $\boldsymbol{S_{\mathbf{int}}}^{|\mathtt{IV}|}(\boldsymbol{x_0}')$, som v inač veľmi jednoduchom a prehľadnom návrhu prúdovej šifry QUAD nepochopil. Vynechaním tohoto kroku, by dôkaz bezpečnosti bol prehľadnejší. Dokonca aj redukcia bezpečnosti šifry QUAD na MQ problém by bola priliehavejšia.

Dôkaz bezpečnosti pre celú šifu QUAD publikovaný autormi šifry v článku [6] sa trochu odlišuje od dôkazu v tejto práci. Hlavný rozdiel nastáva pri odhade nasledujúcej nerovnosti

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathrm{QUAD}}(t,r) \leq mr\mathbf{Adv}^{\mathrm{prng}}_{S'}(T_1) + r\mathbf{Adv}^{\mathrm{prng}}_{\widetilde{g}^S}(T_2). \tag{8.1}$$

V článku [6] kapitola 6 je nasledujúce značenie $\boldsymbol{S'} = g^{S'}$ a $m = 2$. V článku sa píše: „T*ieto 2 generátory pseudonáhodných čisel* $\widetilde{g}^S$ *a* $\boldsymbol{S'}$ *fungujú na interacii náhodne vybraných kvadratických systémov kn rovníc n premenných (k = 2 pre S'). Ako bolo ukázané vo vete 6.5 bezpečnosť takýchto generátorov sa dá zredukovať na problém riešenia kn kvadratických rovníc n premenných.*“ Čo je samozrejme pravda.

Zároveň chcem podotknúť, že generátor $\boldsymbol{S'}$ a $\widetilde{g}^S$, nie sú generátory rovnakého typu. Keystream generátora $\boldsymbol{S'}$ vygenerovaný z $\boldsymbol{x}$ je rovný hodnote kvadratického systému $\boldsymbol{S'}(\boldsymbol{x})$ a keystream vygenerovaný $\widetilde{g}^S$, dostaneme po $\lambda+m$ prechodov konštrukcie šifry QUAD (algoritmus 2).

K odhadu, ktorý následne autori uvádzajú som sa ani väčším úsilím nedopracovať. Odhad uvádzaný autormi

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathrm{QUAD}}(t,r) \leq 3r\mathbf{Adv}^{\mathrm{prng}}_{g^S}(T), \tag{8.2}$$

kde g^S je generátor keystreamu šifry QUAD (λ prechodov) a T je čas uvedený v článku [6], ktorého hodnota nás teraz nebude zaujímať.

Z môjho prístupu v tejto práci, oddiel 6.2.5, som najprv aplikáciou vety 6.1 zhora odhadol hodnotu

$$\mathbf{Adv}^{\mathrm{prng}}_{\widetilde{g}^S}(T_2) \leq (\lambda + m)\mathbf{Adv}^{\mathrm{prng}}_{S}(T_2 + (\lambda + m)T_S).$$

A tým získal dva rovnaké typy generátorov $\boldsymbol{S}$ a $\boldsymbol{S'}$. Následným „sčítaním“ a použitím nerovnice 6.1 som získal

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathrm{QUAD}}(t,r) \leq (\lambda + 2m)r\mathbf{Adv}^{\mathrm{prng}}_{S}(T_2 + (\lambda + m)T_S).$$

Následnou aplikáciou vety 6.2 a vety 6.4 dostávame síce dva rozdielne výsledky, no pre praktické hodnoty je táto redukcia na riešenie MQ problému u oboch prípadoch takmer rovnaká.

Literatúra

[1] Come Berbain, Henri Gilbert, Jacques Patarin: *QUAD: a Practical Stream Cipher with Provable Security*, EUROCRYPT 2006, LNCS 4004, str. 109-128.

[2] Yang, B.-Y. Chen, O. C.-H. Bernstein, D. J. Chen, J.-M.: *Analysis of QUAD*, LNCS 4593, str. 290-308.

[3] Michael Feng-Hao Liu Chi-Jen Lu Bo-Yin Yang Jintai Ding: *Secure PRNGs from Specialized Polynomial Maps over Any* GF(q), October 23, 2007.

[4] Gregory V. Bard: *Algebraic Cryptanalysis*, Springer, 2009.

[5] Douglas R. Stinson: *Cryptography Theory and Practice*, CRC Press, Inc, 1995.

[6] Come Berbain and Henri Gilbert: *On the Security of IV Dependent Stream Ciphers*, Springer-Verlag Berlin Heidelberg 2007.

[7] Oded Goldreich, Shafi Goldwasser, and Silvo Micali: *How to Construct Random Function*, J. ACM 33(4): 792-807, 1986.

[8] M. Bellare and S. Goldwasse: *The complexity of decision versus search*, SIAM J. on Computing, Vol. 23, No. 1, February 1994.

[9] Nicolas Courtois, Louis Goubin, Willi Meier, and Jean-Daniel Tacier: *Solving Underdefined Systems of Multivariate Quadratic Equations*, CP8 Crypto Lab, SchlumbergerSema 36-38 rue de la Princesse, BP45F-78430 Louveciennes Cedex, France.

[10] Magali Bardet: *Étude des systémes algébriques surdéterminés. Applications aux codes correcteurs et á la cryptographie*, PhD thesis, Université Paris VI, 2004.

[11] Gregory V. Bard, Nicolas T. Courtois, and Chris Jefferson: *Efficient Methods for Conversion and Solution of Sparse Systems of Low-Degree Multivariate Polynomials over* GF(2) *via SAT-Solvers.*

[12] Nicolas Courtois, Alexander Klimov, Jacques Patarin, and Adi Shamir: *Efficient algorithms for solving overdefined systems of multivariate polynomial equations*, In Bart Preneel, editor, *Advances in Cryptology* EUROCRYPT 2000, volume 1807 of *Lecture Notes in Computer Science*, pages 392-407. Springer-Verlag, 2000.

[13] Nicolas Courtois and Jacques Patarin: *About the XL Algorithm over* GF(2), In Marc Joye, editor, *Topics in Cryptology* CT-RSA 2003, volume 2612 of *Lecture Notes in Computer Science*, pages 141-157. Springer-Verlag, 2003.

[14] M. Bardet, J.-C. Faugére, and B. Salvy: *On the complexity of gröbner basis computation of semi-regular overdetermined algebraic equations*, In Proceedings of the International Conference on Polynomial System Solving, pages 71–74, 2004.

[15] B.-Y. Yang and J.-M. Chen: *On Theoretical analysis of XL over small fields*, In ACISP 2004, volume 3108 of *Lecture Notes in Computer Science*, pages 277–288. Springer, 2004.

[16] Mneal Koblitz and Alfred J. Meneze: *Another Look at „Provable Security“ II.*, Journal of Cryptology , 2004.

[17] A. Kipnis, J. Patarin, L. Goubin: *Unbalanced Oil and Vinegar Signature Schemes II.*, Advances in Cryptology – EUROCRYPT'99, Proceedings, J. Stern (Ed.), *Lecture Notes in Computer Science*, Springer Verlag, vol. 1592, pp. 206 - 222.

[18] Y. Hashimoto: *Algorithms to solve massively under-defined systems of multivariate quadratic equations*, Proceedings (Industrial Track) of the 8th International Conference on Applied Cryptography and Network Security, pp.26–37, 2010.

[19] Aviad Kipnis and Adi Shamir: *Cryptanalysis of the HFE Public Key Cryptosystem by Relinearization*, Proceeding CRYPTO '99 Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology.

[20] Johan Hastad and Mats Naslund: *Bmgl: Synchronous key-stream henerator with provable security.* – *A*submitted to Nessie Project, 2000.

[21] C. Diem: *The XL-algorithm and a conjecture from commutative algebra*, In Advances in Cryptology – A*SIACRYPT 2004*, volume 3329 of L*ecture Notes in Computer Science*, pages 323-337. Pil Joong Lee, ed., Springer, 2004. ISBN 3-540-23975-8.