SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Evidenčné číslo: FEI-16607-99535

NOVÝ PERTURBAČNÝ MODIFIKÁTOR SCHÉMY HFE

DIPLOMOVÁ PRÁCA

2023 **Bc. Sabina Daniela Pekareková**

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Evidenčné číslo: FEI-16607-99535

NOVÝ PERTURBAČNÝ MODIFIKÁTOR SCHÉMY HFE

DIPLOMOVÁ PRÁCA

Študijný program:	Aplikovaná informatika
Názov študijného odboru:	Informatika
Školiace pracovisko:	Ústav informatiky a matematiky
Vedúci záverečnej práce:	Ing. Viliam Hromada, PhD.

Bratislava 2023 **Bc. Sabina Daniela Pekareková**

Slovenská technická univerzita v Bratislave
Ústav informatiky a matematiky

Fakulta elektrotechniky a informatiky
Akademický rok: 2022/2023
Evidenčné číslo: FEI-16607-99535

ZADANIE DIPLOMOVEJ PRÁCE

Študentka:	**Bc. Sabina Daniela Pekareková**
ID študenta:	99535
Študijný program:	aplikovaná informatika
Študijný odbor:	informatika
Vedúci práce:	Ing. Viliam Hromada, PhD.
Vedúci pracoviska:	doc. Ing. Milan Vojvoda, PhD.
Miesto vypracovania:	Ústav informatiky a matematiky

Názov práce: **Nový perturbačný modifikátor schémy HFE**

Jazyk, v ktorom sa práca vypracuje: slovenský jazyk

Špecifikácia zadania:

Cieľom diplomovej práce je naštudovať a implementovať tzv. perturbačný modifikátor využívaný v tzv. kryptografii viacerých premenných. Ide o oblasť postkvantovej kryptografie využívajúcej polynómy viacerých neurčitých nad konečnými poliami. V roku 2022 navrhol kolektív autorov okolo J.C.Faugera nový modifikátor tzv. HFE schémy, ktorý by mal zvýšiť jej bezpečnosť. Úlohou DP je implementovať HFE schému bez/s modifikátorom a obe implementácie porovnať z hľadiska výpočtovej náročnosti.

Úlohy:

1. Naštudujte problematiku kryptografie viacerých premenných.
2. Naštudujte problematiku HFE zobrazenia a perturbačného modifikátora.
3. Implementujte systém HFE bez/s perturbačným modifikátorom.
4. Implementácie porovnajte a vyhodnoťte výpočtovú náročnosť modifikátora.

Zoznam odbornej literatúry:

1. DING, Jintai; PETZOLDT, Albrecht; SCHMIDT, Dieter S. Multivariate Public Key Cryptosystems. Springer Nature, 2020.
2. FAUGÈRE, Jean-Charles, et al. A New Perturbation for Multivariate Public Key Schemes such as HFE and UOV. Cryptology ePrint Archive, 2022.

Termín odovzdania diplomovej práce:	12. 05. 2023
Dátum schválenia zadania diplomovej práce:	05. 05. 2023
Zadanie diplomovej práce schválil:	prof. Dr. Ing. Miloš Oravec – garant študijného programu

SÚHRN

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Študijný program:	Aplikovaná informatika
Autor:	Bc. Sabina Daniela Pekareková
Diplomová práca:	Nový perturbačný modifikátor schémy HFE
Vedúci záverečnej práce:	Ing. Viliam Hromada, PhD.
Miesto a rok predloženia práce:	Bratislava 2023

Práca sa zaoberá novým perturbačným modifikátorom HFE schémy, ktorý v roku 2022 navrhol kolektív autorov okolo J.C.Faugera. Cieľom tejto práce bolo implementovať HFE schému bez tohto perturbačného modifikátora a s perturbačným modifikátorom a následne obe implementácie porovnať z hľadiska výpočtovej náročnosti.

Kryptosystém HFE bez a s perturbačným modifikátorom sme implementovali ako podpisovú schému. Implementácia je realizovaná pomocou knižnice NTL v jazyku C++. Podpisová schéma s perturbačným modifikátorom je implementovaná s dvomi možnými inverziami, ktoré navrhli autori článku z ktorého vychádza táto práca. Tieto dva spôsoby inverzie sú v tejto práci porovnané. Konkrétne sa porovnáva čas potrebný pre vygenerovanie verejného kľúča, vygenerovanie platného podpisu a overenie platnosti podpisu pre základnú HFE podpisovú schému a dve implementácie HFE podpisovej schémy s perturbačným modifikátorom.

Kľúčové slová: podpisová schéma, HFE, perturbačný modifikátor, postkvantová kryptografia

ABSTRACT

SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA
FACULTY OF ELECTRICAL ENGINEERING AND INFORMATION TECHNOLOGY

Study Programme:	Applied Informatics
Author:	Bc. Sabina Daniela Pekareková
Master's thesis:	New perturbation modifier of HFE scheme
Supervisor:	Ing. Viliam Hromada, PhD.
Place and year of submission:	Bratislava 2023

The thesis deals with a new perturbation modifier of the HFE scheme proposed in 2022 by a team of authors around J.C.Fauger. The aim of this work was to implement the HFE scheme without this perturbation modifier and with the perturbation modifier, and then to compare both implementations in terms of computational complexity.

We implemented the HFE cryptosystem without and with the perturbation modifier as a signature scheme. The implementation is implemented using the NTL library in C++. The signature scheme with the perturbation modifier is implemented with two possible inversions proposed by the authors of the paper on which this work is based. These two inversion methods are compared in this thesis. Specifically, the time required to generate a public key, generate a valid signature, and validate the signature for the basic HFE signature scheme and two implementations of the HFE signature scheme with the perturbation modifier are compared.

Keywords: signature scheme, HFE, perturbation modifier, postquantum cryptography

Poďakovanie

Chcem sa poďakovať vedúcemu práce, Ing. Viliamovi Hromadovi, PhD. za cenné rady, pripomienky, konzultácie počas celého akademického roka a hlavne za vypracované materiály a odporúčané zdroje, ktorými ma zasvätil do mágie zvanej matematika, ktorá stojí za návrhom postkvantových systémov a inšpiroval ma k tomu, aby som sa o ne zaujímala aj ďalej po dopísaní tejto diplomovej práce.

Ďalej by som sa rada poďakovala tzv. GIRLS GANGu - mojim kamarátkam Dominike, Nicole, Karolíne a Michaele, za dodanie odvahy zvoliť si ťažšiu cestu - štúdium BIS, za všetky hodiny strávené spoločným štúdiom i mimo neho, bez ktorých by zrejme päťročné štúdium na FEI nebolo najkrajších 5 rokov môjho života, ale najťažších 9 rokov môjho života.

V neposlednej rade by som sa chcela poďakovať mojim rodičom a priateľovi Petrovi, za neustálu motiváciu a podporu počas štúdia i mimo neho.

Obsah

Zoznam obrázkov a tabuliek

Zoznam skratiek

DSA	Digital Signature Algorithm
HFE	Hidden Field Equations
MQ	Multivariete Quadratic
NIST	National Institute of Standards and Technology
NP	Nondeterministic Polynomial
NTL	Number Theory Library
PQC	Post Quantum Cryptography
RSA	Rivest–Shamir–Adleman

Zoznam algoritmov

Úvod

S príchodom výkonného kvantového počítača prichádza riziko prelomenia bezpečnosti súčasnej asymetrickej kryptografie. Dnešné asymetrické kryptosystémy sú založené na náročnosti riešenia NP problémov, ako je problém faktorizácie čísel na prvočísla alebo problém diskrétneho logaritmu. Sú to problémy pre ktoré je možné overiť navrhnuté riešenie v polynomiálnom čase, avšak nie je známy algoritmus, ktorý by ich vedel efektívne vyriešiť. V roku 1994 [1] Peter Shor navrhol algoritmus (nazývaný Shorov algoritmus), ktorý dokáže vyriešiť problémy teórie čísel ako je práve problém faktorizácie celého čísla a problém diskrétneho logaritmu v polynomiálnom čase na kvantovom počítači. Kvantový počítač pracuje na iných princípoch ako bežný počítač. Využíva qubity, ktoré sa laicky povedané, môžu nachádzať v oboch stavoch (0/1) súčasne. Pre takýto počítač by pri použití Shorovho algoritmu nepomohlo ani zväčšenie parametrov súčasných kryptosystémov (napr. RSA [2] alebo DSA [3]). Tieto systémy by už neboli viac bezpečné. Z toho dôvodu je konštrukcia výkonného kvantového počítača hrozbou pre v súčasnosti používané asymetrické šifrovanie a je potrebný ďalší výskum v oblasti post-kvantovej kryptografie. Post-kvantová kryptografia (PQC) je termín používaný pre kryptografické algoritmy, ktoré by mali byť v post-kvantovej ére odolné voči krypto-analytickým útokom kvantového počítača. Ukázalo sa, že NP problémy (problém faktorizácie, problém diskrétneho logaritmu) by mohli byť vyriešené kvantovým počítačom v polynomiálnom čase. Tejto hrozby si je vedomý aj Americký národný inštitút pre štandardy a technológiu (NIST), ktorý vyhlásil súťaž s názvom *Post-Quantum Cryptography Standardization Process* s cieľom navrhnúť nové kryptografické štandardy odolné voči výkonným kvantovým počítačom. Toto bolo podnetom pre návrh nových štandardov a kryptosystémov. Vedci sa teraz zameriavajú na NP-úplné problémy, o ktorých sa predpokladá, že nie sú efektívne riešiteľné v polynomiálnom čase. Doterajšie návrhy rozdeľujeme do 5 rodín post-kvantových kryptosystémov, podľa odporúčaných matematických problémov, na ktorých sú založené:

- systémy založené na dekódovacom probléme
- systémy založené na hashovacích funkciách
- systémy založené na bodoch mriežky
- systémy založené na supersingulárnej izogénii eliptických kriviek
- systémy založené na sústave nelineárnych rovníc viacerých premenných nad konečným poľom

V tejto diplomovej práci sa zaoberáme novým perturbačným modifikátorom $HFE^{\hat{+}-}$ podpisovej schémy, ktorá patrí medzi systémy založené na sústave kvadratických rovníc. Tieto systémy sú podmnožinou poslednej spomínanej rodiny vyššie. Cieľom diplomovej práce bolo implementovať a porovnať základnú HFE podpisovú schému a schému s novým perturbačným modifikátorom $HFE^{\hat{+}-}$ a porovnať tieto dve implementácie z hľadiska časovej zložitosti.

Práca je štruktúrovaná následovne: V prvej kapitole sú zhrnuté základné pojmy súvisiace s teóriou MQ kryptosystémov a HFE schémou. Druhá kapitola približuje problematiku post-kvantovej kryptografie. V tretej kapitole sa venujeme definícii MQ kryptosystémov, princípom šifrovania a dešifrovania pomocou tohto systému a jeho využitiu ako podpisovej schémy. Štvrtá kapitola pojednáva o základných princípoch HFE schémy. V piatej kapitole sa venujeme novému perturbačnému modifikátoru HFE schémy a následne v siedmej kapitole sa zaoberáme invertovaním tejto HFE schémy s perturbačným modifikátorom. V siedmej kapitole sú uvedené predpokladané pamäťové nároky verejného a súkromného kľúča HFE podpisovej schémy s perturbačným modifikátorom. Ôsma kapitola sa venuje analýze bezpečnosti $HFE^{\hat{+}-}$ schémy. V deviatej kapitole tejto práce je zhrnutá implementácia základnej HFE podpisovej schémy a HFE podpisovej schémy s využitím perturbačného modifikátora, pričom sú uvedené implementácie dvoch možných inverzií. Desiata kapitola sa zaoberá vykonanými meraniami odporúčaných parametrov z článku z ktorého sme vychádzali, ako aj meraniami nových parametrov. V závere sme zhrnuli všetky merania a výsledky.

1 Základné pojmy

Táto diplomová práca sa zaoberá perturbačným modifikátorom HFE podpisovej schémy. HFE podpisová schéma patrí medzi kryptosystémy založené na sústave kvadratických rovníc s viacerými neurčitými nad konečným poľom.

1.1 Polynóm

Funkciu $P(x) = a_n x^n + a_{n-1}x^{n-1} ... + a_1 x + a_0$, kde $a_0 ... a_n \in R$, kde R je okruh a $a_n \neq 0$, nazývame polynóm stupňa n, $a_0 ... a_n$ nazývame koeficienty polynómu a x nazývame neurčitou polynómu.

1.2 Grupa

Definícia 1.1. Grupa je algebrická štruktúra, ktorú tvorí množina prvkov $\{a, b, c, ...\}$ a binárna operácia $\oplus$ definovaná na množine G, pre ktorú platia nasledovné vlastnosti [4]:

- Uzavretosť: pre ľubovoľné $a \in G$, $b \in G$ je prvok $a \oplus b$ v G.
- Asociatívnosť: pre ľubovoľné $a, b, c \in G$, $(a \oplus b) \oplus c = a \oplus (b \oplus c)$
- Identita: V G je prvok identity 0, pre ktorý $a \oplus 0 = 0 \oplus a = a$ pre všetky $a \in G$.
- Aditívny inverzný prvok: Pre každé $a \in G$ existuje inverzný prvok $(-a) \in G$ taký, že $a \oplus (-a) = 0$.

Definícia 1.2. Grupa G, pre ktorú platí $a \oplus b = b \oplus a$ pre všetky $a, b \in G$ sa nazýva abelovská grupa.

1.3 Pole

Definícia 1.3. Množinu s aspoň dvomi prvkami F, s dvomi operáciami $\oplus$ a $\cdot$, pre ktorú platí:

- Množina F tvorí abelovskú grupu (ktorej identita sa nazýva 0) pre operáciu $\oplus$.
- Množina $F\cdot = F - \{0\} = \{a \in F, a \neq 0\}$ tvorí abelovskú grupu (ktorej identita sa nazýva 1) v rámci operácie $\cdot$.
- Distributívny zákon: Pre všetky $a, b, c \in F, (a \oplus b) \cdot c = (a \cdot c) \oplus (b \cdot c)$.

nazývame pole a značíme $(F, +, \cdot)$.

Definícia 1.4. Pojmom konečné pole označujeme pole $(F, +, \cdot)$, pre ktoré platí, že množina F je konečná.

Pre jednoduchosť budeme ďalej $(F, +, \cdot)$ označovať ako F.

1.4 Rozšírenie konečného pola

Nech K je konečné pole a nech F je podmnožina K, ktorá tvorí taktiež pole vzhľadom na operácie definované na poli K. Potom F sa nazýva podpole poľa K a pole K sa nazýva rozšírením poľa (nadpoľom) F.

Rozšírenie konečného poľa môžme definovať pomocou ireducibilných polynómov. Majme konečné pole F a polynóm $p(x)$ (ireducibilný polynóm o jednej neurčitej x nad poľom F stupňa n), pre ktorý platí $p(x) \in F[x]$. Množinu K môžeme definovať ako $K = F[x]/(p(x))$, teda množinu pozostávajúcu z polynómov v jednej neurčitej x, s koeficientami z F, reprezentujúcimi zvyšky delenia polynómov $F[x]$ a polynómu $p(x)$. Operácie medzi prvkami K sú sčítanie a násobenie polynómov $mod p(x)$. Konečné pole $(K, +, \cdot)$ nazývame nadpoľom F, ktoré je n-tým stupňom rozšírenia poľa F [5].

1.5 Afinná transformácia

Afinná transformácia je zobrazenie, ktoré zachováva vzájomnú polohu bodov a paralelizmus rovnobežiek. Nech V a W sú dva vektorové priestory nad tým istým poľom, a nech $f : V \to W$ je zobrazenie. Ak existuje vektor $b \in W$ a lineárne zobrazenie $L : V \to W$, také, že pre každý vektor $v \in V$ platí $f(v) = L(v) + b$, potom f je afinná transformácia [6]. Afinnú transformáciu môžme reprezentovať pomocou matice a vektoru. Nech V a W sú dva vektorové priestory rovnakej veľkosti dimenzie n nad konečným poľom R a nech je daná afinná transformácia $f : V \to W$. Potom existujú matica $A \in R^{n \times n}$ a vektor $b \in R^n$, také, že pre každý vektor $v \in V$ platí [7]:

$$f(v) = Av + b$$

Matica A potom predstavuje lineárnu zložku afinnej transformácie a vektor b predstavuje posunutie afinnej transformácie. Vektor $v \in V$ teda môžeme transformovať na vektor $w \in W$ nasledovne:

$$w = f(v) = Av + b.$$

1.6 Trapdoor

Trapdoorová funkcia f s „trapdoorom“ t je jednosmerná funkcia. Jednosmerná funkcia f je funkcia, ktorú je ľahké vypočítať, ale veľmi ťažké invertovať bez znalosti tzv. „trapdooru“ (pasce). Trapdoor teda spočíva v tajnej informácii t, ktorá umožňuje jednoduché vypočítanie inverzie [8].

2 Post-kvantová kryptografia

Kryptografia je veda, ktorá sa zaoberá šifrovaním informácií, pričom odtajnenie šifrovanej informácie je možné iba so znalosťou špeciálneho kľúča. Je súčasťou našej každodennej komunikácie, využívame ju pri prehliadaní internetového obsahu, online bankovníctve a emailovej komunikácii. Cieľom kryptografie je zachovanie:

- Dôvernosti: Nelegitímny používateľ by nemal získať žiadne informácie o tajomstve správy.
- Autentifikácie entity: Potvrdenie identity entity.
- Autentifikácia údajov: Potvrdenie pôvodu údajov.
- Integrity údajov: Zabezpečenie toho, že informácie neboli zmenené neoprávnenou osobou subjektov

Kryptografiu delíme podľa spôsobu akým prebieha šifrovanie a dešifrovanie na:

- symetrickú - využíva na šifrovanie a dešifrovanie informácie rovnaký tajný kľúč, ktorý poznajú iba komunikujúce strany,
- asymetrickú - používa dva kľúče:
  - verejný kľúč, pomocou ktorého správy šifrujeme,
  - súkromný, pomocou ktorého správy dešifrujeme.

Verejný kľúč, používaný v asymetrickej kryptografii, teda poznajú všetci a môžu pomocou neho správu zašifrovať, ale súkromný kľúč pozná iba legitímny príjemca, ktorý ako jediný môže správu dešifrovať. Asymetrické kryptosystémy okrem klasického šifrovania správy umožňujú aj tvorbu digitálnych podpisov. Tie slúžia na overenie integrity dokumentu. Vlastník súkromného kľúča, pomocou neho podpíše správu (napr. odtlačok podpisovaného dokumentu) čím získa platný podpis. Podpis potom zverejní spolu so správou, pre ktorú bol vygenerovaný. Verejný kľúč je vygenerovaný na základe súkromného kľúča tak, aby pôsobil ako náhodná postupnosť a neprezrádzal nič o štruktúre súkromného kľúča. Pomocou verejného kľúča je možné overiť podpis a teda zistiť, či bol naozaj odvodený od daného odtlačku dokumentu. Vygenerovaním platného podpisu môžeme dokázať autorstvo daného dokumentu.

Post-kvantová kryptografia je kryptografia o ktorej sa predpokladá, že je odolná voči útokom kvantového počítača. Kvantový počítač funguje na iných princípoch ako

bežný počítač. Využíva javy z kvantovej mechaniky. Podľa kvantovej fyziky, kvantový systém môže existovať v kombinácii všetkých povolených stavov súčasne [9]. Toto miešanie stavov sa nazýva kvantová superpozícia. Základnou jednotkou kvantového počítača je qubit, ktorý môže existovať v super-pozícii svojich dvoch „základných“ stavov (0 alebo 1). Inými slovami, qubit sa môže nachádzať v oboch stavoch súčasne. Toto je kľúčom k sile kvantových počítačov.

Keďže pomocou Shorovho algoritmu [1] je možné riešiť problém faktorizácie čísel na celé čísla a problém diskrétneho logaritmu na kvantovom počítači v polynomiálnom čase, asymetrické kryptosystémy založené na týchto problémoch považujeme za prelomené a je potrebné sa venovať návrhu nových post-kvantových systémov.

3 MQ kryptosystém

Táto práca sa zaoberá podpisovou schémou HFE, ktorá patrí do rodiny post-kvantových kryptosystémov založených na sústave nelineárnych rovníc viacerých premenných nad konečným poľom, konkrétne do jej podmnožiny MQ systémov. MQ kryptosystémy sú založené na probléme riešenia sústavy kvadratických rovníc s viacerými premennými nad konečným poľom.

Ide o asymetrické šifrovanie, teda používajú sa súkromný a verejný kľúč. Verejný kľúč v MQ predstavuje sústava polynómov, v ktorej platí, že znaky otvoreného textu sú reprezentované samotnými neurčitými tejto sústavy a znaky zašifrovaného textu sú reprezentované hodnotami polynómov.

$$\textit{verejný kľúč} \begin{cases} x_1x_2 + x_2x_3 = y_1 \\ x_1x_3 + x_2 + 1 = y_2 \\ x_2x_3 + x_1 + x_2 + x_3 = y_3. \end{cases} \tag{1}$$

Zatiaľ čo verejný kľúč je sústava kvadratických polynómov, ktorá je v podstate náhodná, t.j. členy pozostávajú z rôznej kombinácie neurčitých, súkromný kľuč v MQ systémoch má špeciálnu štruktúru, nazývanú aj trapdoor, vďaka ktorej je možné efektívne nájsť riešenie príslušnej sústavy kvadratických rovníc s viacerými premennými. Hodnoty x_i v (1) sú znaky otvoreného textu a y_i sú znaky zašifrovaného textu.

3.1 Formálna definícia MQ kryptosystému

Definícia 3.1. [10] Pre jednoduchosť uvažujme konečné pole F_q a $n, m \in N$

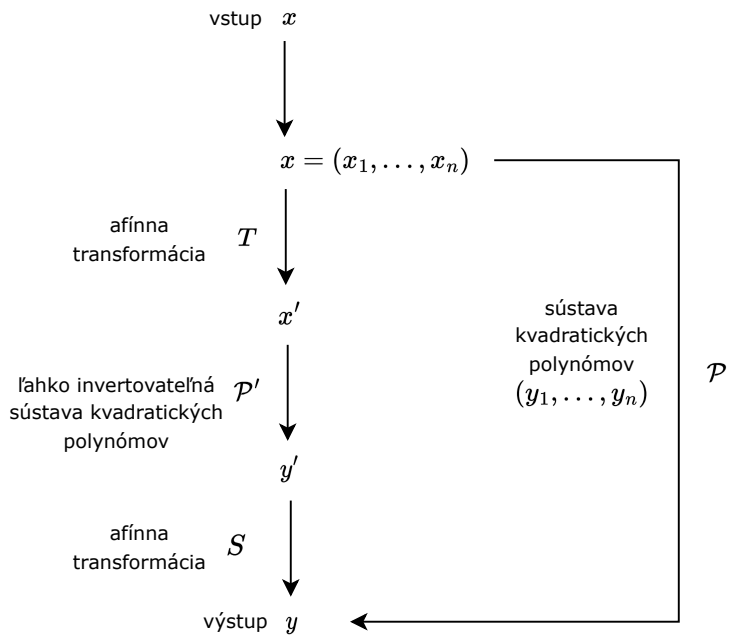
- Nech S je afinná transformácia, $S : F_q^m \to F_q^m$
- Nech T je afinná transformácia, $T : F_q^n \to F_q^n$
- Nech P' je ľahko invertovateľná sústava m kvadratických polynómov s n neurčitými nad základným poľom $GF(2)$.
- Nech P je sústava m kvadratických polynómov s n neurčitými, ktorá vznikne zložením $P = S \circ P' \circ T$, kde $P : F_q^n \to F_q^m$.

Pre MQ-kryptosystém platí, že afinné transformácie S, T a ľahko invertovateľná sústava P' tvoria súkromný kľúč a sústava P, ktorá vznikne ich zložením, predstavuje verejný kľúč. Pomocou zobrazení S a T zamaskujeme trapdoor P'. Sústava P predstavujúce verejný

kľúč, ktoré zverejníme, už teda pôsobí ako náhodná sústava polynómov, ktorá neprezrádza nič o súkromnom kľúči a bez znalosti tajných S a T je náročné ju invertovať.

3.2 Šifrovanie

Majme dve komunikujúce strany $\mathcal{A}$ a $\mathcal{B}$. Strana $\mathcal{A}$ zverejní svoj verejný kľúč, pričom znalosť S, T a P' má iba ona. V MQ kryptosystémoch je možné zašifrovať n-rozmerný vstup nad príslušným konečným poľom na m-rozmerný výstup nad príslušným konečným poľom. Strana $\mathcal{B}$ ma k dispozícii sústavu P, do ktorej za hodnoty neurčitých dosadí vstup veľkosti n, ktorý chce zašifrovať. Dosadením za neurčité v sústave P získa m-rozmerný vektor, ktorý predstavuje vypočítané hodnoty m polynómov. Na šifrovanie v MQ kryptosystémoch sa dá pozerať ako na zobrazenie P, ktoré vstup x zobrazí na výstup y, t.j. $P(x) = y$. Zobrazenie $P(x) = y.$ je ekvivalentné k postupnému aplikovaniu zobrazení S, T a P', teda $S(P'(T(x)) = y$, toto môžeme vidieť na obrázku 1. Keďže strana $\mathcal{B}$ nemá znalosť zobrazení S, T a P', pozná len finálny výsledok y a nepozná medzivýsledky x', y'.



Obr. 1: Šifrovanie.

3.3 Dešifrovanie

Pri dešifrovaní využívame znalosť súkromného kľúča pozostávajúceho zo zobrazení S, T a P', ktoré postupne invertujeme. Po obdržaní správy y, ktorú strana $\mathcal{B}$ získala dosadením vstupu x do $P(x) = y$, $\mathcal{A}$ dešifruje následovne:

1. Strana $\mathcal{A}$ obdržala správu y od strany $\mathcal{B}$.

2. Správu y dosadí do inverznej transformácie ku transformácii S, $S^{-1}(y) = y'$.

3. Medzivýsledok y' dosadí do inverzného zobrazenia k zobrazeniu P', $P'^{-1}(y') = x'$.

4. Medzivýsledok x' dosadí do inverznej transformácie ku transformácii T, $T^{-1}(x') = x$.

5. Výsledné x predstavuje dešifrovanú správu.

3.4 MQ kryptosystém ako podpisová schéma

V prípade, že MQ kryptosystém uvažujeme ako podpisovú schému, pri konštrukcii podpisu nejde o klasické šifrovanie a dešifrovanie ako je uvedené vyššie, ale o overenie podpisu a jeho vygenerovanie. Pri digitálnych podpisoch platí, že strana $\mathcal{A}$ ktorá disponuje znalosťou zobrazení S, T a P', t.j. súkromného kľúča, vie dokument podpísať (t.j. vygenerovať platný podpis) a strana $\mathcal{B}$, ktorá disponuje verejným kľúčom, vie overiť, či je daný podpis platný.

3.4.1 Vygenerovanie podpisu

Nech strana $\mathcal{A}$ vlastní súkromný kľúč nejakej MQ podpisovej schémy. Ak chce strana $\mathcal{A}$ vypočítať platný podpis pre správu y, postupuje následovne:

1. Strana $\mathcal{A}$ môže počítať podpis priamo pre správu y, alebo napríklad pre jej odtlačok.

2. Strana $\mathcal{A}$ dosadí hodnotu správy do inverznej afinnej transformácie ku transformácii S, $S^{-1}(y) = y'$.

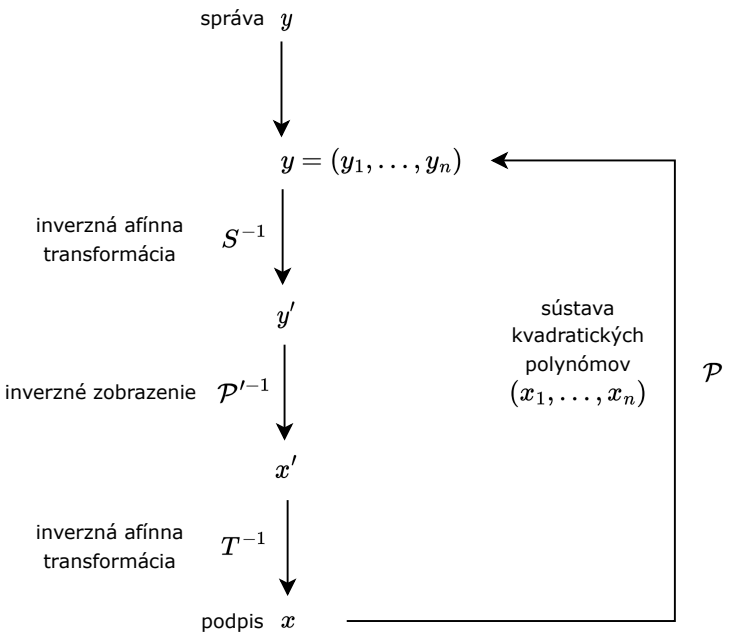
3. Medzivýsledok y' potom dosadí do inverzného zobrazenia k zobrazeniu P', $P'^{-1}(y') = x'$.

4. Medzivýsledok x' dosadí do inverznej afinnej transformácie ku transformácii T, $T^{-1}(x') = x$

5. Výsledné x predstavuje hodnotu príslušného digitálneho podpisu.

3.4.2 Overenie podpisu

Ak by strana $\mathcal{B}$ chcela overiť platnosť podpisu x pre správu y, stačí aby do verejného kľúča dosadila jeho hodnotu. Ak bol podpis x vygenerovaný pomocou správneho súkromného kľúča, potom musí platiť, že dosadením hodnôt podpisu x za hodnoty neurčitých v sústave polynómov P, hodnoty ktoré príslušné polynómy budú nadobúdať, sa budú rovnať hodnotám správy y.



Obr. 2: Generovanie a overenie podpisu.

4 HFE schéma

HFE schéma je typ kryptosystému s verejným kľúčom navrhnutý Jacquesom Patarinom [11] nadväzujúc na myšlienku Matsumotovho and Imaiovho systému [12]. Ide o typ trapdooru, ktorý využíva teóriu nadpolí, čo znamená, že používa základné pole F_q a jeho n-té rozšírenie F_{q^n}. Kanonická bijekcia $\phi_n : F_{q^n} \to F_q^n$ hovorí, že každý prvok z F_{q^n} môže byť zobrazený na nejaký n-prvkový vektor nad základným konečným poľom F_q. HFE kryptosystém môže byť implementovaný ako šifrovací systém alebo ako podpisová schéma. HFE trapdoor sa na začiatku nezadáva priamo ako systém polynómov, ale ako polynóm s jednou neurčitou nad rozšírením konečného poľa. Polynóm v HFE tvare pre pole rozšírenia $GF(2^n)$ vyzerá následovne:

$$P'(X) = \sum_{\substack{0 \leq i,j \leq d \\ 2^i + 2^j \leq d}} C_{i,j} X^{2^i + 2^j} + \sum_{\substack{0 \leq k \leq d \\ 2^k \leq d}} B_k X^{2^k} + A \tag{2}$$

Koeficienty $A, B_k, C_{i,j}$ a neurčitá X reprezentujú prvky z rozšírenia poľa $GF(2^n)$. Číslo d predstavuje stupeň polynómu $P'(X)$, ktoré má špeciálnu vlastnosť - musí byť malé. Ak by táto vlastnosť nebola zachovaná, polynóm by bol potencionálne veľkého stupňa a nebolo by možné ho efektívne invertovať. Členy polynómu sú mocniny X vynásobené koeficientom, teda prvkom z rozšírenia poľa. Platí, že každý polynóm nad rozšírením poľa $GF(2^n)$ je možné napísať ako ekvivalentnú sústavu n polynómov o n neurčitých nad poľom $GF(2)$, vďaka kanonickej bijekcii ϕ. Ak je navyše polynóm $P'(X)$ v tvare, že mocniny termov sú alebo mocnina dvojky (členy s koeficientami B_k) alebo súčty dvoch mocnín dvojky (členy s koeficientami $C_{i,j}$), výsledná ekvivalentná sústava n polynómov o n neurčitých nad poľom $GF(2)$ bude kvadratická..

Zatiaľ čo verejný kľúč je množina kvadratických polynómov, ktorá sa javí ako náhodná, súkromný kľúč je systémom zloženým z kvadratických polynómov, ktoré majú trapdoor riešený cez polynóm v HFE tvare, vďaka čomu je sústava rovníc efektívne riešiteľná. Ak chceme zamaskovať štruktúru trapdooru HFE vo verejnom kľúči P, na sústavu polynómov prislúchajúcu zobrazeniu P' musíme aplikovať afinné transformácie S a T.
Verejný kľúč P tvorí zloženie:

$$P = S \circ \phi \circ P' \circ \phi^{-1} \circ T \tag{3}$$

kde:

- S a T sú afinné transformácie,
- P' je polynóm v HFE tvare,

- $\phi_n : F_{q^n} \rightarrow F_q^n$, je zobrazenie prvku z F_{q^n} na ekvivalentný n-prvkový vektor nad základným poľom F_q.

Súkromný kľúč teda tvoria:

- afinné transformácie $S : F_q^n \rightarrow F_q^n$ a $T : F_q^n \rightarrow F_q^n$,
- $P' = F_q^n \rightarrow F_q^m$, ľahko invertovateľná sústava m polynómov s n neurčitými nad F_q.

V tejto práci uvažujeme implementáciu HFE ako podpisovej schémy.

5 Perturbačný modifikátor

V [13] Faugére a kol. navrhli nový perturbačný modifikátor HFE schémy. Táto schéma sa označuje $HFE^{\hat{+}-}$ keďže zavádza dva nové parametre. Prvým parametrom je parameter t, ktorý určuje rozmery novej perturbácie označovanej ako „Plus so strieškou“ ($\hat{+}$) a parameter a, ktorý vystupuje v modifikátore „Mínus“ ($-$).

5.1 Perturbácia $\hat{+}$

V tejto kapitole si vysvetlíme ako funguje perturbácia $\hat{+}$. Ako je známe, HFE pracuje nad 2 poliami: základným poľom F_q a jeho n-tým rozšírením F_q^n. Platí že každý prvok nad rozšírením poľa, je možné previesť na n-prvkový vektor nad základným poľom.

Postup pre vytvorenie modifikovaného HFE je následovný:

1. Zvolí sa HFE polynóm nad poľom F_{q^n}.
2. Zvolí sa parameter t.
3. Vygeneruje sa t náhodných kvadratických polynómov nad F_q, teda máme kvadratické polynómy o n neurčitých nad základným polom.
4. Každý z polynómov sa prevedie na polynóm $p_i{}'$, ktorý je ekvivalentom polynómu p_i avšak v poli F_{q^n}.
5. Následne vyberieme t náhodne zvolených prvkov z poľa F_{q^n}, ktoré označíme ako ako $\beta_1, ..., \beta_t$.
6. Polynóm Q, potom vznikne ako lineárna kombinácia polynómov $p'_1 ... p'_t$, kde koeficienty lineárnej kombinácie sú $\beta_1, ..., \beta_t$.

$$Q(x) = \Sigma_{i=1}^{t} \beta_i p\prime_i(x) \tag{4}$$

7. Výsledný trapdoor (polynóm F), teda vznikne pripočítaním polynómu Q ku HFE polynómu H.

$$F(x) = Q(x) + H(x) \tag{5}$$

Príklad 5.1. *Príklad perturbovaného HFE Majme základné pole $GF(2)$ a jeho rozšírenie $GF(2^3)$, ktoré je definované pomocou polynómu x^3+x+1 nad poľom $GF(2)$. Prvky rozšírenia poľa sú: $GF(2^3) = \{0, 1, \alpha, \alpha^2, \alpha^3 = \alpha+1, \alpha^4 = \alpha^2+\alpha, \alpha^5 = \alpha^2+\alpha+1, \alpha^6 = \alpha^2+1\}$. Platí, že každý prvok nad rozšírením poľa je možné previesť na n-prvkový vektor nad základným*

poľom. Toto zobrazenie sa nazýva aj kanonická bijekcia a autori článku [13] ho označujú $\phi_n : F_{q^n} \to F_q^n$. *Keďže ide o bijekciu, existuje aj inverzné zobrazenie* ${\phi_n}^-1 : F_q^n \to F_{q^n}$, *ktoré zobrazí n-rozmerný vektor nad základným poľom* F_q *na prvok z poľa* F_q^n. *Zobrazenia* ϕ *a* ${\phi_n}^{-1}$ *vyzerajú následovne:*

- $\phi_3(0) = (0,0,0), \quad {\phi_3}^{-1}(0,0,0) = 0$
- $\phi_3(1) = (1,0,0), \quad {\phi_3}^{-1}(1,0,0) = 1$
- $\phi_3(\alpha) = (0,1,0), \quad {\phi_3}^{-1}(0,1,0) = \alpha$
- $\phi_3(\alpha^2) = (0,0,1), \quad {\phi_3}^{-1}(0,0,1) = \alpha^2$
- $\phi_3(1+\alpha) = (1,1,0), \quad {\phi_3}^{-1}(1,1,0) = 1+\alpha$
- $\phi_3(\alpha+\alpha^2) = (0,1,1), \quad {\phi_3}^{-1}(0,1,1) = \alpha+\alpha^2$
- $\phi_3(1+\alpha+\alpha^2) = (1,1,1), \quad {\phi_3}^{-1}(1,1,1) = 1+\alpha+\alpha^2$
- $\phi_3(1+\alpha^2) = (1,0,1), \quad {\phi_3}^{-1}(1,0,1) = 1+\alpha^2$

Postupujme podľa algoritmu vyššie:

1. *Zvolí sa HFE polynóm nad poľom* F_{2^3}. *Majme teda polynóm v HFE tvare uvedenom v rovnici* (6). *Uvažujme parametre d=5 a t=2. Ako jednotlivé náhodne zvolené koeficienty HFE polynómu uvažujme:*

   - *Pre* $i = 0$, $j = 0$, *teda pre* $x^2 : \alpha_{0,0} = 0$
   - *Pre* $i = 0$, $j = 1$, *teda pre* $x^3 : \alpha_{0,1} = \alpha^2$
   - *Pre* $i = 0$, $j = 2$, *teda pre* $x^5 : \alpha_{0,2} = \alpha + 1$
   - *Pre* $i = 1$, $j = 1$, *teda pre* $x^4 : \alpha_{1,1} = 1$
   - *Pre* $i = 1$, $j = 2$, *už koeficient nevolíme, keďže* $2^1 + 2^2 > 5$

   Náhodne zvolený polynóm v HFE tvare bude vyzerať následovne:

   $$H(x) = \alpha^2 x^3 + x^4 + (1+\alpha)x^5 \tag{6}$$

   Polynóm $H(x)$ *vieme previesť na sústavu 3 kvadratických polynómov o 3 neurčitých* x_1, x_2, x_3 *nad základným poľom* F_2*:*

   - $h_1(x_1, x_2, x_3) = x_2x_3 + x_2 + x_3$

- $h_2(x_1, x_2, x_3) = x_1x_2 + x_1x_3 + x_2x_3 + x_1 + x_2$
- $h_3(x_1, x_2, x_3) = x_2x_3 + x_1 + x_3$

2. *Zvolíme si parameter* $t = 2$.

3. *Vygenerujeme náhodné kvadratické polynómy:*

- $p_1(x_1, x_2, x_3) = x_1x_3 + x_2x_3 + {x_2}^2$
- $p_2(x_1, x_2, x_3) = x_1x_2 + x_1x_3 + {x_1}^2 + {x_3}^2$

4. *Každý z polynómov prevedieme na jeho ekvivalentom v poli* F_2^n*:*

- $p_1(0,0,0) = 0$, *teda jeho ekvivalent* $p_1'(0) = 0$
- $p_1(1,0,0) = 0$, *teda jeho ekvivalent* $p_1'(1) = 0$
- $p_1(0,1,0) = 1$, *teda jeho ekvivalent* $p_1'(\alpha) = 1$
- $p_1(0,0,1) = 0$, *teda jeho ekvivalent* $p_1'(\alpha^2) = 0$
- $p_1(1,1,0) = 1$, *teda jeho ekvivalent* $p_1'(1+\alpha) = 1$
- $p_1(0,1,1) = 0$, *teda jeho ekvivalent* $p_1'(\alpha+\alpha^2) = 0$
- $p_1(1,1,1) = 1$, *teda jeho ekvivalent* $p_1'(1+\alpha+\alpha^2) = 1$
- $p_1(1,0,1) = 1$, *teda jeho ekvivalent* $p_1'(1+\alpha^2) = 1$

- $p_2(0,0,0) = 0$, *teda jeho ekvivalent* $p_2'(0) = 0$
- $p_2(1,0,0) = 1$, *teda jeho ekvivalent* $p_2'(1) = 1$
- $p_2(0,1,0) = 0$, *teda jeho ekvivalent* $p_2'(\alpha) = 0$
- $p_2(0,0,1) = 1$, *teda jeho ekvivalent* $p_2'(\alpha^2) = 1$
- $p_2(1,1,0) = 0$, *teda jeho ekvivalent* $p_2'(1+\alpha) = 0$
- $p_2(0,1,1) = 1$, *teda jeho ekvivalent* $p_2'(\alpha+\alpha^2) = 1$
- $p_2(1,1,1) = 0$, *teda jeho ekvivalent* $p_2'(1+\alpha+\alpha^2) = 0$
- $p_2(1,0,1) = 1$, *teda jeho ekvivalent* $p_2'(1+\alpha^2) = 1$

$p_1'(x) = (\alpha+1)x^6 + (\alpha^2+1)x^5 + (\alpha+1)x^4 + (\alpha^2+\alpha+1)x^3 + (\alpha^2+\alpha+1)x^2 + (\alpha^2+1)x$
$p_2'(x) = (\alpha^2+\alpha)x^6 + \alpha x^5 + (\alpha^2+1)x^4 + \alpha^2x^3 + (\alpha+1)x^2 + (\alpha^2+\alpha+1)x$

5. *Následne vyberieme* $t = 2$ *náhodne zvolených prvkov z poľa* $F_(2^n)$*:*

- $\beta_1 = \alpha + \alpha^2$
- $\beta_2 = \alpha$

.

6. *Polynóm Q, potom vznikne ako:*

$$Q(x) = \beta_1 p'_1 + \beta_2 p'_2 =$$
$$(\alpha+\alpha^2)((\alpha+1)x^6+(\alpha^2+1)x^5+(\alpha+1)x^4+(\alpha^2+\alpha+1)x^3+(\alpha^2+\alpha+1)x^2+(\alpha^2+1)x)+$$
$$+(\alpha)((\alpha^2+\alpha)x^6+\alpha x^5+(\alpha^2+1)x^4+\alpha^2x^3+(\alpha+1)x^2+(\alpha^2+\alpha+1)x) =$$
$$= (\alpha^2+\alpha)x^6+(\alpha^2+\alpha+1)x^5+(\alpha^2+\alpha+1)x^3+(\alpha+1)x^2+(\alpha^2+\alpha)x \quad (7)$$

7. *Výsledný trapdoor:*

$$F(x) = Q(x) + H(x) =$$
$$(\alpha^2+\alpha)x^6+(\alpha^2+\alpha+1)x^5+(\alpha^2+\alpha+1)x^3+(\alpha+1)x^2+(\alpha^2+\alpha)x+\alpha^2x^3+x^4+(1+\alpha)x^5 =$$
$$= (\alpha^2+\alpha)x^6+\alpha^2x^5+x^4+(\alpha+1)x^3+(\alpha+1)x^2+(\alpha^2+\alpha)x \quad (8)$$

K vzniknutému polynómu F prislúcha sústava polynómov nad základným poľom, ktorú autori článku [13] označujú ako $\hat{F}$:

- $f_1(x_1, x_2, x_3) = x_2x_3 + x_2 + x_3$
- $f_2(x_1, x_2, x_3) = x_1x_3 + x_3$
- $f_3(x_1, x_2, x_3) = x_1x_3 + x_1 + x_2 + x_3$

5.2 Modifikátor -

Modifikátor - zavádza parameter a, ktorý predstavuje počet polynómov, ktoré sa odstránia z verejného kľúča. Afinné transformácie S a T, sú zvolené náhodne, pričom platí že:

- $S : F_q^n \to F_q^{n-a}$
- $T : F_q^n \to F_q^n$

Aplikovaním afinných transformácii S a T na perturbovanú sústavu polynómov $\hat{F}$, by sme dostali sústavu kvadratických polynómov P, predstavujúcu verejný kľúč pre $HFE^{\hat{+}-}$ schému.

6 Inverzia HFE s perturbačným modifikátorom

Keďže stupeň polynómu $F(x)$, ktorý vznikol z HFE polynóm $H(x)$ použitím perturbácie $\hat{+}$, môže byť potencionálne veľkého stupňa, klasická metóda HFE inverzie faktorizáciou polynómu (napr. cez Berlekampov algoritmus [14]), by bola príliš zložitá. V pôvodnom článku [13] preto autori navrhli 2 spôsoby, ako je možné invertovať $HFE^{\hat{+}-}$, t.j. polynóm $F(x)$:

- prehľadávaním všetkých možností
- cez projekciu

Majme hodnotu $y \in F_{q^n}$ prislúchajúcu polynómu $F(x)$, t.j.

$$F(x) = y \tag{9}$$

Invertovať modifikované HFE teda znamená, že hľadáme také x, pre ktoré platí, že jeho dosadením do polynómu dostaneme hodnotu y. Pri polynómoch, ktoré sú nízkeho stupňa, ako je napríklad $H(x)$. Ak by sme ho chceli invertovať, uvažovali by sme polynóm $H(x) - y = 0$, ktorý by sme faktorizovali.

Pri faktorizácii polynómu $F(x)$, je dôležité mať na pamäti, že platia následovné skutočnosti:

- $F(x) - y = 0$
- $F(x) = H(x) + Q(x)$
- $Q(x) = \Sigma_{i=1}^{t} \beta_i p\prime_i(x) = \beta_1 p\prime_1 + ... + \beta_t p\prime_t$

Polynóm (9) teda môžeme zapísať ako:

$$F(x) - y = H(x) + \beta_1 p\prime_1 + ... + \beta_t p\prime_t - y = 0 \tag{10}$$

6.1 Inverzia cez prehľadávanie všetkých možností

Tento spôsob inverzie je založený na prehľadávaní všetkých možností hodnôt, ktoré môžu nadobúdať polynómy $p\prime_i(x)$. Postup je následovný:

Postup inverzie cez prehľadávanie všetkých možností:

1. Uvažuj polynóm $F(x) - y = H(x) + \beta_1 p\prime_1 + ... + \beta_t p\prime_t - y = 0$.

2. Vygeneruj náhodný vektor t hodnôt $(y_1, y_2, ..., y_t)$ nad poľom F_q.

3. Hodnoty z vektora dosaď za hodnoty perturbačných polynómov $p\prime_1(x) = y_1, ..., p\prime_t(x) = y_t$, teda uvažujme polynóm

$$H(x) + \beta_1 y_1 + ... + \beta_t y_t - y = 0 \tag{11}$$

4. Keďže polynóm v rovnici (11) je nízkeho stupňa, invertuj ho, hľadaj také x', pre ktoré platí $H(x') + \beta_1 y_1 + ... + \beta_t y_t = y$.

5. Ak sa také x' nepodarilo nájsť, vráť sa na krok č.2 a zopakuj postup.

6. Ak sa také x' podarilo nájsť, over, či pre všetky $i = 1, .., t$ platí, že $p\prime_i(x') = y_i$, teda či dosadením hodnoty x' do $p\prime_i(x)$ dostaneme hodnotu y_i, teda hodnotu na príslušnej i-tej pozícii vo vektore z kroku č.2.

7. Ak pre nejaké $i = 1, .., t$ neplatí, že $p\prime_i(x') = y_i$, vráť sa na krok č.2 a zopakuj postup.

8. Ak pre všetky $i = 1, .., t$ platí, že $p\prime_i(x') = y_i$, potom x' je hľadanou inverziou $F(x) = y$.

Táto práca uvažuje HFE schému implementovanú ako podpisovú schému. V takom prípade, podpisovú schému stačí nájsť jedno také x, pre ktoré platí vzťah z rovnice (9). Preto v kroku č. 2 stačí generovať vektor náhodných hodnôt a v prípade, že nájdeme platné riešenie môžeme skončiť. Ak by ale schéma HFE bola implementovaná ako šifrovanie, museli by sme nájsť všetky také x, pre ktoré platí vzťah z rovnice (9). V prípade šifrovania by sme teda museli krok č. 2 nahradiť cyklom, ktorý by prehľadával všetky možné vektory t hodnôt $(y_1, y_2, ..., y_t)$. V takomto cykle by sa teda prechádzalo q^t hodnôt.

Príklad 6.1. *Príklad invertovania pomocou prehľadávania všetkých možností. Majme polynóm $H(x)$ z rovnice (6), $\beta_1 = \alpha + \alpha^2$, $\beta_2 = \alpha$ a $F(x) = (\alpha^2 + \alpha)x^6 + \alpha^2 x^5 + x^4 + (\alpha + 1)x^3 + \alpha x^2 + (\alpha^2 + \alpha)x$ nad poľom F_{2^3} a majme hodnotu $y = \alpha^2 + 1$, pre ktorú ho chceme invertovať. Hľadáme také x, pre ktoré platí $F(x) = \alpha^2 + 1$.*

1. *Uvažujme polynóm $F(x) - y = H(x) + \beta_1 p\prime_1 + \beta_2 p\prime_2 - y = \alpha^2 x^3 + x^4 + (1 + \alpha)x^5 + (\alpha + \alpha^2)p\prime_1 + (\alpha)p\prime_2 - (\alpha^2 + 1) = 0$.*

2. *Vygenerujme vektor $t = 2$ náhodných hodnôt z F_2 ako $(y_1, y_2) = (0, 1)$.*

3. *Pre tieto hodnoty nadobudne polynóm $H(x) + \beta_1 y_1 + \beta_2 y_2 - y = 0$ hodnotu $\alpha^2 x^3 + x^4 + (1 + \alpha)x^5 + (\alpha + \alpha^2)0 + (\alpha)1 - (\alpha^2 + 1) = \alpha^2 x^3 + x^4 + (1 + \alpha)x^5 + (\alpha^2 + \alpha + 1)$.*

4. *Keďže ide o HFE polynóm nízkeho stupňa, použijeme faktorizáciu pomocou Berlekampovho algoritmu. Dostaneme rozklad:* $\alpha^2x^3 + x^4 + (1+\alpha)x^5 + (\alpha^2+\alpha+1) = (\alpha+1)(x+\alpha^2+\alpha+1)(x+\alpha+1)^2(x^2+x+\alpha^2+\alpha+1)$*, z čoho vidíme, že korene polynómu sú:*
   - $x' = \alpha^2+\alpha+1$ *(z faktoru* $(x+\alpha^2+\alpha+1)$*)*
   - $x' = \alpha+1$ *(z faktoru* $(x+\alpha+1)$*)*
5. *Našli sme kandidátov na* x'*, pokračujeme ďalej.*
6. *Overíme či po dosadení* x' *za* x*, platí že* $p\prime_1(x) = 0, p\prime_2(x) = 1$*.*
   - *Pre* $x' = \alpha^2+\alpha+1$*,* $p\prime_1(\alpha^2+\alpha+1) = 1$ *a* $p\prime_2(\alpha^2+\alpha+1) = 0$
   - *Pre* $x' = \alpha+1$*,* $p\prime_1(\alpha+1) = 1$ *a* $p\prime_2(\alpha^2+\alpha+1) = 0$
7. *Vidíme že po dosadení hodnôt* x'*, sme nedostali očakávané hodnoty (0, 1), preto sa vrátime na krok č. 2.*
8. *Vygenerujeme vektor* $t = 2$ *náhodných hodnôt z* F_2 *ako* $(y_1, y_2) = (1, 0)$*.*
9. *Pre tieto hodnoty nadobudne polynóm* $H(x) + \beta_1 y_1 + \beta_2 y_2 - y = 0$ *hodnotu* $\alpha^2x^3 + x^4 + (1+\alpha)x^5 + (\alpha+\alpha^2)1 + (\alpha)0 - (\alpha^2+1) = \alpha^2x^3 + x^4 + (1+\alpha)x^5 + (\alpha+1)$*.*
10. *Polynóm invertujeme pomocou Berlekampovej faktorizácie:* $\alpha^2x^3 + x^4 + (1+\alpha)x^5 + (\alpha+1) = (\alpha+1)(x+\alpha)(x^4+\alpha^2x^3+(\alpha^2+\alpha)x^2+(\alpha^2+\alpha+1)x+\alpha^2+1)$*. Vidíme, že korene polynómu sú:*
    - $x' = \alpha$ *(z faktoru* $(x+\alpha)$*)*
11. *Overíme či po dosadení* x' *za* x*, platí že* $p\prime_1(x) = 1, p\prime_2(x) = 0$*.*
    - *Pre* $x' = \alpha$*,* $p\prime_1(\alpha) = 1$ *a* $p\prime_2(\alpha) = 0$
12. *Vidíme že vektor* $(1, 0) = (y_1, y_2)$*, platí teda že* $F(\alpha) = \alpha^2 + 1$*. Podarilo sa nám teda invertovať perturbovaný HFE polynóm a riešenie pôvodného polynómu* $F(x) = y$ *bolo* $x = \alpha$*.*

6.2 Inverzia cez projekciu

Druhý navrhovaný spôsob inverzie z článku [13] je založený na aplikácii operátora projekcie. Definícia projekcie hovorí, že je to lineárna transformácia (resp. lineárny operátor) P na nejakom vektorovom priestore, pre ktorú platí, že $P \circ P = P$. Teda jej opakovaným aplikovaním na nejaký vektor by sme dostali vždy ten istý výsledok.

Príklad 6.2. *Majme trojrozmerný vektorový priestor nad R, t.j. R^3, klasický trojrozmerný vektor v priestore so všeobecným vyjadrením (x_1, x_2, x_3). Ak by sme si definovali zobrazenie vektorov, ktoré by zachovávalo iba prvé dve súradnice a z tretej súradnice urobí vždy nulu, dostali by sme príklad projekcie z priestoru do roviny, keďže z trojrozmerného vektora by sme dostali dvojrozmerný vektor s treťou súradnicou nulovou. Operátor projekcie P by sme vyjadrili pomocou násobenia matice ako:*

$$P(x) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ 0 \end{pmatrix}$$

Teda napríklad pre $P(1, 2, 3) = (1, 2, 0)$, $P(1, 2, -10) = (1, 2, 0)$.

Vieme že pre perturbované HFE platí:

- $F(x) - y = 0$
- $F(x) = H(x) + Q(x)$
- $Q(x) = \Sigma_{i=1}^{t} \beta_i p\prime_i(x) = \beta_1 p\prime_1 + ... + \beta_t p\prime_t$

Inverzia pomocou projekcia teda spočíva v hľadaní takého operátora projekcie Π_t, ktorý spĺňa dôležitú vlastnosť:

$$\Pi_t(F(x)) = \Pi_t(H(x)) \tag{12}$$

Z rovnice (12) vyplýva následovné:

1. Hľadáme lineárne zobrazenie ktorého aplikáciou na výsledné hodnoty polynómov $H(x)$ a $F(x)$, dostaneme vždy tú istú hodnotu, teda $\Pi_t(F(x)) = \Pi_t(H(x))$.
2. Keďže platí:
   - $\Pi_t(F(x)) = \Pi_t(H(x))$
   - $F(x) = H(x) + Q(x)$
   - Π_t je lineárny operátor,

   tak z rovnosti $\Pi_t(F(x)) = \Pi_t(H(x) + Q(x))$ vyplýva, že $\Pi_t(Q(x)) = 0$.
3. Projekcia $\Pi_t(x)$ bude teda také zobrazenie, ktoré všetky potencionálne hodnoty ktoré môže polynóm $Q(x)$ nadobúdať, bude zobrazovať na nulu.

V prípade perturbovaného HFE bude projekcia $\Pi_t(x)$ polynóm nad rozšírením F_{q^n}. Hľadáme teda polynóm $\Pi_t(x)$, ktorý zobrazuje všetky možné výsledky $Q(x)$ na nulu. Chceme teda nájsť polynóm, ktorý pre vybrané vstupy nadobúda nulové hodnoty. Nech hodnoty, pre ktoré má polynóm nadobúdať nulové hodnoty, sú $c_1, c_2, ..., c_n$. Potom polynóm $f(x)$, ktorý vznikne ako:

$$f(x) = (x - c_1)(x - c_2)...(x - c_n)$$

je polynóm n-tého stupňa, pre ktorý platí $f(c_1) = ... = f(c_n) = 0$. Hľadáme polynóm $\Pi_t(x)$, pre ktorý platí, že zobrazuje všetky možné výsledky $Q(x)$ na nulu. Pripomeňme si definíciu $Q(x)$:

$$Q(x) = \beta_1 p\prime_1 + ... + \beta_t p\prime_t$$

kde koeficienty β_i sú pevne dané a každý polynóm $p\prime_t$ môže nadobúdať q hodnôt z poľa F_q, tak potom všetkých hodnôt, ktoré môže polynóm $Q(x)$ nadobúdať, je maximálne q^t.Všetky hodnoty, ktoré môže nadobúdať $Q(x)$ sú vlastne všetky lineárne kombinácie prvkov $\beta_1, ..., \beta_t$, t.j. prvky, ktoré dostaneme, keď v predpise

$$\beta_1 p\prime_1 + ... + \beta_t p\prime_t$$

dosadíme za hodnoty polynómov $p\prime_1, ..., p\prime_t$ všetky možné hodnoty. Vypočítame si q^t prvkov (každý polynóm môže nadobúdať q hodnôt a týchto polynómov máme počet t). Tieto prvky si označíme ako $c\prime_1, ..., p\prime_{q^t}$, následne z nich vyrobíme polynóm:

$$\Pi_t(x) = (x - c_1)(x - c_2)...(x - c_{q^t}) = \prod_{j=1}^{q^t}(x - c_j) \tag{13}$$

Príklad 6.3. *Majme polynóm $H(x)$ z rovnice* (6)*, $\beta_1 = \alpha + \alpha^2$, $\beta_2 = \alpha$ a polynómy $p\prime_1, p\prime_2$:*

$$p\prime_1(x) = (\alpha + 1)x^6 + (\alpha^2 + 1)x^5 + (\alpha + 1)x^4 + (\alpha^2 + \alpha + 1)x^3 + (\alpha^2 + \alpha + 1)x^2 + (\alpha^2 + 1)x$$

$$p\prime_2(x) = (\alpha^2 + \alpha)x^6 + \alpha x^5 + (\alpha^2 + 1)x^4 + \alpha^2 x^3 + (\alpha + 1)x^2 + (\alpha^2 + \alpha + 1)x$$

Výsledné centrálne zobrazenie $F(x) = (\alpha^2 + \alpha)x^6 + \alpha^2 x^5 + x^4 + (\alpha + 1)x^3 + \alpha x^2 + (\alpha^2 + \alpha)x$. Aby sme našli polynóm projekcie, musíme najprv nájsť všetky lineárne kombinácie prvkov β_1, β_2, kde koeficienty lineárnej kombinácie sú prvky z poľa $F_q = F_2$. Hľadáme teda všetky prvky podľa predpisu:

$$\beta_1 p_1 + \beta_2 p_2$$

kde $p_1, p_2 \in F_2$. Dostaneme teda 4 následovné prvky:

1. $p_1 = 0$, $p_2 = 0 \rightarrow (\alpha + \alpha^2) * 0 + (\alpha) * 0 = 0$
2. $p_1 = 1$, $p_2 = 0 \rightarrow (\alpha + \alpha^2) * 1 + (\alpha) * 0 = 0 = \alpha + \alpha^2$
3. $p_1 = 0$, $p_2 = 1 \rightarrow (\alpha + \alpha^2) * 0 + (\alpha) * 0 = 1 = \alpha$
4. $p_1 = 1$, $p_2 = 1 \rightarrow (\alpha + \alpha^2) * 1 + (\alpha) * 0 = 1 = \alpha^2$

Zostrojíme polynóm $\Pi_t(x)$ podľa predpisu z rovnice (12), kde za $c_1 = 0$, $c_2 = \alpha + \alpha^2$,$c_3 = \alpha$,$c_4 = \alpha^2$, t.j.:

$$\Pi_t(x) = (x - 0)(x - (\alpha + \alpha^2))(x - \alpha)(x - \alpha^2) = x + x^2 + x^4$$

Polynóm $\Pi_t(x)$ je možné vypočítať iba jeden krát (pri generovaní kľúča), keďže je závislý na prvkoch $\beta_1, ..., \beta_t$ a jeho predpis sa nemení pre rôzne inverzie toho istého centrálneho zobrazenia. Pre invertovanie perturbovaného zobrazenia $F(x) = y$ využívame vlastnosť, že $\Pi_t(H(x)) = \Pi_t(y)$, keďže výraz $\Pi_t(H(x))$ je polynóm maximálne stupňa $q^t d$, teda síce väčšieho ako je polynóm $H(x)$, ale potencionálne menšieho ako $F(x)$. Ak vyjadríme $\Pi_t(H(x))$ ako polynóm (teda vypočítame $\Pi_t(x) \circ H(x)$) a zároveň vypočítame hodnotu $\Pi_t(y)$, tak sa môžeme pokúsiť o nájdenie riešenia vzťahu $\Pi_t(H(x)) - \Pi_t(y) = 0$, pomocou faktorizácie polynómu $\Pi_t(H(x)) - \Pi_t(y)$ cez Berlekampov algoritmus, z ktorej vidíme korene $\Pi_t(H(x)) - \Pi_t(y)$ a teda hodnoty pre ktoré $\Pi_t(H(x)) - \Pi_t(y) = 0$. Vo faktoroch ktoré získame faktorizáciou Berlekampovým algoritmom sa bude nachádzať aj hľadané riešenie pôvodného systému $F(x) = y$. Správnosť riešenia je nutné overiť podľa vzťahu $F(x) = y$.

Príklad 6.4. *Majme projekčný polynóm $\Pi_t(x) = x + x^2 + x^4$ a hodnotu $y = \alpha^2 + 1$. Invertujeme teda $F(x) = y$, pričom hľadáme také x, že $F(x) = \alpha^2 + 1$. Ako prvé vypočítame hodnotu $\Pi_t(x) = x + x^2 + x^4$, kde za x dosadíme hodnotu $H(x) = \alpha^2 x^3 + x^4 + (1 + \alpha)x^5$, teda:*

$$\Pi_t(H(x)) = (\alpha^2 x^3 + x^4 + (1 + \alpha)x^5) + (\alpha^2 x^3 + x^4 + (1 + \alpha)x^5)^2 + (\alpha^2 x^3 + x^4 + (1 + \alpha)x^5)^4$$

$$\Pi_t(H(x)) = x^6 + x^5 + x^4 + x^3 + x^2 + x$$

Aplikovaním projekčného polynómu na y dostaneme $\Pi_t(y) = \Pi_t(1 + \alpha^2) = 1$. Faktorizáciou polynómu $\Pi_t(H(x)) - \Pi_t(y)$ dostaneme:

$$x^6+x^5+x^4+x^3+x^2+x-1 = (x+\alpha)(x+\alpha+1)(x+\alpha^2)(x+\alpha^2+1)(x+\alpha^2+\alpha)(x+\alpha^2+\alpha+1)$$

Dostali sme 6 koreňov (α, $\alpha+1$, α^2, α^2+1, $\alpha^2+\alpha$, $\alpha^2+\alpha+1$), pre ktoré musíme overiť, že platí $F(x) = 1+\alpha^2$:

1. α, $F(\alpha) = \alpha^2 + 1$
2. $\alpha + 1$, $F(\alpha + 1) = 1$
3. α^2, $F(\alpha^2) = \alpha^2 + \alpha + 1$
4. $\alpha^2 + 1$, $F(\alpha^2 + 1) = \alpha^2 + 1$
5. $\alpha^2 + \alpha$, $F(\alpha^2 + \alpha) = \alpha + 1$
6. $\alpha^2 + \alpha + 1$, $F(\alpha^2 + \alpha + 1) = 1$

Našli sme teda 2 korene, pre ktoré platí $F(x) = 1 + \alpha^2$ a teda sú riešeniami inverzie centrálneho zobrazenia. Sú to korene $\alpha^2 + 1$ a α.

Postup pre hľadanie inverzie perturbovaného HFE pomocou inverzie cez projekciu by bol teda následovný:

1. Vypočítaj všetky lineárne kombinácie hodnôt $\beta_1, ..., \beta_t$ pomocou t koeficientov z poľa F_q, teda urč všetky hodnoty $\Sigma_{i=1}^{t} \beta_i p\prime_i, p_i \in F_q$. Vypočítaj množinu C prvkov z poľa F_{q^n}, ktorá je daná ako $C = c_j | c_j = \Sigma_{i=1}^{t} \beta_i p\prime_i, p_i \in F_q$.
2. Vypočítaj projekčný polynóm $\Pi_t(x) = \prod_{j \in C}(x - c_j) = (x - c_1)(x - c_{q^t})$.
3. Vypočítaj aplikáciu projekčného polynómu na HFE polynóm, $\Pi_t(H(x))$.
4. Vypočítaj aplikáciu projekčného polynómu na y, $\Pi_t(y)$.
5. Nájdi korene polynómu $\Pi_t(H(x)) - \Pi_t(y)$.
6. Korene $\Pi_t(H(x)) - \Pi_t(y)$ označ ako x'. Platí, že $\Pi_t(H(x')) - \Pi_t(y)$
7. Over ktoré x' z množiny nájdených x', je také, že platí $F(x') = y$. Dané x' je hľadanou inverziou systému $F(x) = y$.

7 Pamäťové nároky kľúčov

7.1 Verejný kľúč

Uvažujme konečné pole $GF(2)$, kde každý koeficient je 1 bit. Ďalej uvažujme odporúčané parametre z článku [13] $q = 2$, $n = 263$, $d = 65$, $t = 6$, $a = 7$. Verejný kľuč je tvorený sústavou $(n - a)$ kvadratických polynómov s n neurčitými. Každý z týchto polynómov obsahuje kvadratické, lineárne a absolútne členy. Aby sme mohli vypočítať dolný odhad veľkosti kľúča, musíme si uvedomiť, že každý z polynómov zo sústavy verejného kľúča, môže mať $\binom{n}{2}$ kvadratických, n lineárnych a 1 absolútny koeficient. Dolný odhad veľkosti verejného kľúča teda vypočítame ako *maximálny počet možných koeficientov* * *počet polynómov v sústave* pomocou výrazu:

$$\left(\binom{n}{2} + n + 1 \right) * (n - a) \tag{14}$$

Dosadením hodnôt $n = 263$ a $a = 7$ do výrazu (14) teda dostaneme:

$$\left(\binom{263}{2} + 263 + 1 \right) * (263 - 7) = 8887552 \text{ bitov}$$

Horný odhad veľkosti verejného kľúča $HFE^{\hat{+}-}$ schémy je teda 8887552 bitov, čo je približne 1,11 MB. V pôvodnom článku [13] autori uvádzajú, že pre nimi odporúčané parametre, ktoré sme uvažovali, je veľkosť verejného kľúča 1111 kB = 1,08 MB čo je približne rovné dolnému odhadu nášho verejného kľúča.

7.2 Súkromný kľuč

Súkromný kľúč pozostáva zo znalosti afinných transformácií S a T a centrálneho zobrazenia tvoreného HFE polynómom. Každá afinná transformácia pozostáva z matice veľkosti $n \times n$ a vektora veľkosti n, teda jej pamäťové nároky sú $n \times n + n$ bitov.

7.2.1 Súkromný kľúč pre $HFE^{\hat{+}-}$ s inverziou cez prehľadávanie všetkých možností

Pre výpočet súkromného kľúča pri použití invertovania cez prehľadávanie všetkých možností, musíme brať do úvahy okrem afinných transformácií S a T aj koeficienty $\beta_1, ..., \beta_t$, ktorých znalosť je potrebná aby sme mohli vykonať inverziu a vygenerovať platný podpis. Tieto koeficienty predstavujú prvky z poľa $GF(2^n)$. Ich pamäťové nároky sú $t \times n$ bitov. Polynóm v HFE tvare je stupňa d a každý koeficient je z poľa $GF(2^n)$, preto zaberá $(d + 1) \times n$ bitov. Na uloženie tohto súkromného kľúča teda potrebujeme:

$2 * (n^2 + n) + t * n + (d + 1) * n = 2 * (263^2 + 263) + 6 * 263 + (65 + 1) * 263 = 157800$ bitov.

Veľkosť súkromného kľúča pre $HFE^{\hat{+}-}$ schému s inverziou cez prehľadávanie všetkých možností je 157800 bitov, čo je rovné približne 19,73 kB.

7.2.2 Súkromný kľúč pre $HFE^{\hat{+}-}$ s inverziou cez projekciu

V prípade súkromného kľúča s inverziou cez projekciu zohľadňujeme projekčný polynóm a výpočet je následovný:

$$2 * (n^2 + n) + (d * q^t + 1) * n = 2 * (263^2 + 263) + (65 * 2^6 + 1) * 263 = 1233207 \text{ bitov}$$

Veľkosť súkromného kľúča pre $HFE^{\hat{+}-}$ schému s inverziou cez projekciu je 1233207 bitov, čo je rovné približne 154,15 kB.

8 Analýza bezpečnosti

V kapitole 4 z článku [13] sa autori venujú predpokladu bezpečnosti $HFE^{\hat{+}-}$ schémy. Podľa tohto článku z ktorého vychádza táto diplomová práca, sú odporúčané optimálne parametre pre 128-bitovú bezpečnosť $q = 2$, $n = 263$, $d = 65$, $t = 6$, $a = 7$. Na základe dosadenia do vzťahov ktoré uvádzajú, sme vybrali nové parametre:

- $q = 2$, $n = 263$, $d = 33$, $t = 8$, $a = 7$
- $q = 2$, $n = 263$, $d = 129$, $t = 5$, $a = 7$

ktoré sme tiež testovali, viac v kapitole 10.

8.1 Priame útoky

Autori počítajú bezpečnosť pre tzv. priame útoky. Jedná sa o algebraické útoky, ktoré sa snažia priamo invertovať sústavu $P(x) = y$. Tieto útoky spočívajú v riešení sústavy rovníc verejného kľúča. Zložitosť sa odvádza od stupňa regularity, ktorý sa v článku odhaduje na základe vzťahu:

$$D_{reg} = \frac{(q-1)(r+a+t-\epsilon)}{2} \tag{15}$$

kde r predstavuje hodnosť HFE polynómu, pričom r počítame ako $r = \lfloor log_q(d-1) \rfloor + 1$.

Dosadením hodnôt odporúčaných parametrov $q = 2$, $n = 263$, $d = 65$ ($r = 7$), $t = 6$, $a = 7$ do vzťahu (15) dostaneme stupeň regularity schémy $HFE^{\hat{+}-}$:

$$D_{reg} = \frac{(2-1)(7+7+6-0)}{2} = 10$$

Pre nové parametre, ktoré sme testovali sú hodnoty stupňa regularity nasledovné:

- pre $d = 33$ ($r = 6$), $t = 8$:

$$D_{reg} = \frac{(2-1)(6+7+8-0)}{2} = 10.5$$

- $d = 129$ ($r = 8$), $t = 5$:

$$D_{reg} = \frac{(2-1)(8+7+5-0)}{2} = 10$$

Keďže stupeň regularity pre nami testované parametre nie je nižší, tieto parametre by nemali byť voči priamym útokom zraniteľnejšie ako odporúčané parametre z článku.

8.2 Štrukturálne útoky

Na rozdiel od priamych útokov, kde vyriešením sústavy rovníc verejného kľúča by sme získali falošný podpis, štrukturálne útoky zamerané na súkromný kľúč umožňujú útočníkovi generovať viacero falošných podpisov. Myšlienkou v [15] a všetkých príbuzných „Minrank“ útokoch, je využitie predvolenej hodnosti pri hľadaní lineárnej kombinácie matíc nízkej hodnosti. Týmto spôsobom útoku na súkromný kľúč sa dá nájsť afínna transformácia. Autori vyššie spomínaného článku vychádzajú z troch vzťahov s odhadom zložitosti:

$$O\left(\binom{n+r+a+t+1}{r+a+t+1}^{\omega}\right) \tag{16}$$

$$O\left(\binom{n+r+t+1}{r+t+1}^{\omega}\right) \tag{17}$$

$$q^{(2r+1)t}\left(n^{\omega}\binom{2r+1}{r}^{\omega}\right) \tag{18}$$

V týchto vzťahoch vystupuje ω, reprezentujúca lineárnu algebraickú konštantu rovnú 2.37188.

Dosadením hodnôt odporúčaných parametrov $q = 2$, $n = 263$, $d = 65$ ($r = 7$), $t = 6$, $a = 7$ do vzťahov (16), (17) a (18) dostaneme hodnoty:

$$O\left(\binom{263+7+7+6+1}{7+7+6+1}^{2.37188}\right) = 2^{248}$$

$$O\left(\binom{263+7+6+1}{7+6+1}^{2.37188}\right) = 2^{182}$$

$$2^{(2*7+1)6}\left(263^{2.37188}\binom{2*7+1}{7}^{2.37188}\right) = 2^{139}$$

Pre nové parametre, ktoré sme testovali je zložitosť nasledovná:

- pre $d = 33$ ($r = 6$), $t = 8$:

$$O\left(\binom{263+6+7+8+1}{6+7+8+1}^{2.37188}\right) = 2^{256}$$

$$O\left(\binom{263+6+8+1}{6+8+1}^{2.37188}\right) = 2^{192}$$

$$2^{(2*6+1)8}\left(263^{2.37188}\binom{2*6+1}{6}^{2.37188}\right) = 2^{148}$$

- $d = 129$ $(r = 8)$, $t = 5$:

$$O\left(\binom{263+8+7+5+1}{8+7+5+1}^{2.37188}\right) = 2^{248}$$

$$O\left(\binom{263+8+5+1}{8+5+1}^{2.37188}\right) = 2^{182}$$

$$2^{(2*8+1)5}\left(263^{2.37188}\binom{2*8+1}{8}^{2.37188}\right) = 2^{138}$$

Úroveň zložitosti parametrov z článku je 2^{248}, 2^{182} a 2^{139}. Pre naše parametre $d = 129$, $t = 5$ je zložitosť rovná 2^{248}, 2^{182}, 2^{138} a pre $d = 33$, $t = 8$ je zložitosť 2^{256}, 2^{192}, 2^{148}. Z týchto výpočtov vyplýva, že úroveň bezpečnosti nových parametrov $d = 33$, $t = 8$ by nemala byť nižšia, ako úroveň parametrov odporúčaných v článku a úroveň parametrov $d = 129$, $t = 5$ by bola takmer rovnaká ako pre odporúčané parametre.

9 Implementácia

Implementovali sme HFE podpisovú schému s perturbačným modifikátorom a bez perturbačného modifikátora, pričom verzia s perturbačným modifikátorom je implementovaná s dvomi možnými inverziami. Implementácia je spravená v jazyku C++ s využitím knižnice NTL[1] na platforme Windows.

Knižnica NTL (Number Theory Library) je knižnica v jazyku C++, ktorá je určená pre prácu s algoritmami a dátovými štruktúrami z oblasti teórie čísel, algebraickej geometrie a kryptografie. Poskytuje implementácie mnohých algoritmov, ktoré sú dôležité pre kryptografiu a teóriu čísel, ako napríklad rýchle násobenie v konečných poliach, testovanie prvočíselnosti, faktorizáciu celých čísel. Taktiež poskytuje implementácie algoritmov na prácu s polynómami, eliptickými krivkami a algebraickými číslami. Pomocou tejto knižnice sme teda reprezentovali polynómy, prvky konečných polí a operácií, ktoré sme na ne aplikovali.

Keďže sme vychádzali z odporúčaných nastavení z článku [13], kde figuroval parameter $q = 2$, naša implementácia pracuje s fixným základným poľom $F_q = F_2$, teda poľom $GF(2)$. Nastaviteľné parametre, ktoré sme testovali boli:

- `modulus_deg` - predstavuje parameter n, parameter určujúci stupeň ireducibilného polynómu, ktorý definuje rozšírenie poľa F_{q^n},
- `hfe_deg` - parameter určujúci stupeň polynómu v HFE tvare,
- `t` - parameter perturbácie $\hat{+}$, určujúci počet perturbačných polynómov (p'_i z rovnice (4)),
- `a` - parameter modifikátora $-$, určujúci počet polynómov ktoré sa odstránia.

V našej implementácii uvažujeme základné pole $GF(2)$. Jeho prvky reprezentujeme pomocou triedy `GF2` z knižnice NTL.

9.1 Polynóm

Ako sme už v tejto práci spomínali, verejný kľúč HFE podpisovej schémy pozostáva zo sústavy kvadratických polynómov. Každý kvadratický polynóm obsahuje kvadratické členy, lineárne členy a absolútny člen. Na reprezentáciu polynómu sme navrhli šablónovú triedu `Polynomial`, s tromi atribútmi:

[1] https://libntl.org/

- `Mat<T> m_quadratic_coefficient` - maticu $n \times n$ rozmerov, predstavujúca kvadratické koeficienty polynómu,
- `Vec<T> m_linear_coefficient` - n-rozmerný vektor, predstavujúci lineárny koeficienty polynómu,
- `T m_constant` - číslo reprezentujúce absolútny člen polynómu.

Príklad 9.1. *Ak by sme teda mali polynóm nad $GF(2)$ v tvare:*

$$f(x_1, x_2, x_3) = {x_1}^2 + {x_3}^2 + x_1x_2 + x_1x_3 + x_3 + 1$$

jeho reprezentácia pomocou matice s kvadratickými koeficientmi, vektora s lineárnymi koeficientmi a číslom predstavujúcim absolútny člen, by bola nasledovná:

- $m\_quadratic\_coefficient = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$
- $m\_linear\_coefficient = (0, 0, 1)$
- $m\_coefficient = 1$

Sústavu s n takýmito polynómami, potom vieme jednoducho reprezentovať pomocou n-rozmerného vektora. Používame na to triedu `Vec<T>`.

9.2 Generovanie HFE polynómu

Polynóm v HFE tvare je vlastne polynóm, ktorého prvky sú koeficienty z rozšírenia poľa $GF(2^n)$. Keďže toto rozšírenie je definované ireducibilným polynómom, najprv si vygenerujeme ireducibilný polynóm so stupňom určeným parametrom `modulus_deg`. Náhodný ireducibilný polynóm stupňa `modulus_deg` vygenerujeme pomocou funkcie `BuildIrred(modulus, modulus_deg)`, kde `modulus` je ireducibilný polynóm, ktorý reprezentujeme NTL triedou `GF2X`, teda triedou pre polynómy nad konečným poľom GF(2). Pre rozšírenie konečného poľa $GF(2^n)$ využívame triedu `GF2E`. Hodnoty rozšírenia konečného poľa $GF(2^n)$ inicializujeme použitím NTL funkcie `GF2E::init(modulus)`. Táto funkcia berie ako vstupný argument ireducibilný polynóm, pretože prvky rozšírenia vznikajú použitím funkcie *mod* P (kde P je ireducibilný polynóm). Samotný HFE polynóm reprezentujeme triedou `GF2EX`, určenou pre polynómy nad rozšírením poľa.

Pre generovanie polynómu v HFE tvare sme si navrhli triedu `HFE`, ktorá obsahuje funkciu `generateHFEPolynomial(long modulus_deg, long hfe_deg)`. Jej vstupné argumenty sú stupeň ireducibilného polynómu, ktorý definuje rozšírenie poľa a stupeň

polynómu HFE, ktorý chceme vygenerovať. Funkcia postupuje podľa nasledovného algoritmu 1:

Algoritmus 1 Algoritmus generovania HFE polynómu

1: $HFE[0] \leftarrow$ náhodný prvok z $GF(2n)$ {$HFE[i]$ je koeficient pri člene i-tého stupňa v polynóme HFE}
2: $i \leftarrow 0$
3: **while** $2^i <$ stupeň polynómu **do**
4: $\quad HFE[2^i] \leftarrow$ náhodný prvok z $GF(2^n)$
5: $\quad i \leftarrow i + 1$
6: **end while**
7: $i \leftarrow 0$
8: **while** $2^i <$ stupeň polynómu **do**
9: $\quad j \leftarrow i + 1$
10: $\quad$ **while** $2^i + 2^j <$ stupeň polynómu **do**
11: $\quad\quad HFE[2^i + 2^j] \leftarrow$ náhodný prvok z $GF(2^n)$
12: $\quad\quad j \leftarrow j + 1$
13: $\quad$ **end while**
14: $\quad i \leftarrow i + 1$
15: **end while**

Uvažujme HFE polynóm z rovnice (2). Na riadku číslo 1 zavoláme funkciu, ktorá koeficient neurčitej X^0 z polynómu HFE (teda koeficient A z rovnice (2)) nastaví na náhodný prvok z rozšírenia poľa $GF(2^n)$. V cykle sa riadku číslo 3 sa postupne prechádzajú mocniny čísla. Potom na riadku číslo 4 nastavujeme koeficient B_k pri neurčitej X^{2^k} na náhodný koeficient z rozšírenia poľa $GF(2^n)$. Na riadku číslo 11 obdobne nastavujeme koeficient $C_{i,j}$ pri neurčitej $X^{2^i+2^j}$ na náhodný koeficient z rozšírenia poľa $GF(2^n)$. Takto vytvoríme polynóm, ktorý má koeficienty z $GF(2^n)$ nastavené iba pri X^0, X^{2^k} a $X^{2^i+2^j}$, teda polynóm v HFE tvare.

9.3 Prevod HFE polynómu na sústavu rovníc

Keď už máme vygenerovaný polynóm v HFE tvare, t.j. polynóm v jednej neurčitej nad $GF(2^n)$, potrebujeme ho previesť na ekvivalentnú sústavu polynómov nad $GF(2)$. Na tento prevod sme si vytvorili funkciu `hfeToSystemOfPolynomials(long modulus_deg, GF2EX hfe)`, ktorej argumentmi sú:

- počet polynómov na ktoré prevedieme HFE polynóm,

- polynóm v HFE tvare.

HFE polynóm vieme previesť na sústavu $n \times n$ polynómov, preto stupeň nových polynómov a ich počet v sústave sú rovnaké. Postupne prevádzame jednotlivé koeficienty HFE polynómu, na koeficienty polynómov z novej sústavy podľa algoritmov 2, 3, 4 .

Každý polynóm nad $GF(2)$ obsahuje hodnoty kvadratických, lineárnych a absolútneho koeficientu, reprezentovaných maticou, vektorom a číslom. Uvažujme označenie jednotlivých častí následovne:

- A_i skalár reprezentujúci absolútny člen i-tého polynómu zo sústavy,
- L_i vektor reprezentujúci lineárne členy i-tého polynómu zo sústavy,
- Q_i matica reprezentujúca kvadratické členy$i$-tého polynómu zo sústavy.

Algoritmus 2 Algoritmus pre prevod lineárnych koeficientov HFE polynómu na lineárne koeficienty polynómov zo sústavy

1: $linearne\_koeficienty \leftarrow$ nulové vektory
2: $i \leftarrow 0$
3: **while** $i <$ stupeň HFE polynómu **do**
4: **if** $2^i >$ stupeň polynómu **then**
5: **break**
6: **end if**
7: $B \leftarrow HFE[2^i]$
8: $j \leftarrow 0$
9: **while** $j <$ stupeň polynómu **do**
10: $A \leftarrow B * \alpha^{j*2^i}$
11: $k \leftarrow 0$
12: **while** $k <$ stupeň polynómu **do**
13: **if** $A[k] == 1$ **then**
14: $linearne\_koeficienty[k][j] \leftarrow linearne\_koeficienty[k][j] + 1$
15: **end if**
16: $k \leftarrow k + 1$
17: **end while**
18: $j \leftarrow j + 1$
19: **end while**
20: $i \leftarrow i + 1$
21: **end while**

Ak chceme získať lineárne koeficienty pre každý z polynómov zo sústavy, postupujeme podľa algoritmu 2. Prevodom lineárnych koeficientov HFE polynómu na lineárne koeficienty príslušných polynómov zo sústavy, dostaneme vektor obsahujúci lineárne koeficienty každého polynómu zo sústavy. Lineárne koeficienty každého polynómu sú reprezentované vektorom hodnôt z $GF(2)$. Výstupom je teda štruktúra `Vec<Vec<GF2>>`. Tento algoritmus postupne spracováva členy $B_i X^{2^i}$ z rovnice 2 a upravuje hodnoty $L_1, ..., L_n$. Každý z vektorov L_i je na začiatku nulový vektor. V hlavnom cykle teda priradíme hodnotu koeficientu pri X^{2^i} do premennej B. Vo vnorenom cykle na riadku číslo 10 si vypočítame hodnotu $B * \alpha^{j*2^i}$ predstavujúci nejaký prvok z poľa $GF(2^n)$, ktorý priradíme premennej A. V cykle na riadku číslo 13 potom prechádzame reprezentáciu prvku, ak sa na k-tej pozícii nachádza hodnota 1, pripočítame ju do premennej $linearne\_koeficienty$ na pozíciu k, j. Inými slovami, pripočítame príspevok prvku ku lineárnemu koeficientu x_j vo vektore L_k.

Algoritmus 3 Algoritmus pre prevod kvadratických koeficientov HFE polynómu na kvadratické koeficienty polynómov zo sústavy

```
1: kvadraticke_koeficienty ← nulové matice
2: i ← 0
3: while 2^i < stupeň HFE polynómu do
4:    j ← i
5:    while j^2 < stupeň HFE polynómu do
6:       if i! = j then
7:          if 2^i + 2^j > stupeň HFE polynómu then
8:             break
9:          else
10:            A ← HFE[index]
11:            r ← 0
12:            while r <= stupeň polynómu do
13:               r ← 0
14:               while s <= stupeň polynómu do
15:                  temp ← A * α^((r−1)*2^i+(s−1)*2^j)
16:                  k ← 0
17:                  while < stupeň polynómu do
18:                     if temp[k] == 1 then
19:                        koef ← kvadraticke_koeficienty[k][r − 1][s − 1]
20:                        koef ← koef + 1
21:                        kvadraticke_koeficienty[k][r − 1][s − 1] ← koef
22:                     end if
23:                     k ← k + 1
24:                  end while
25:                  s ← s + 1
26:               end while
27:               r ← r + 1
28:            end while
29:         end if
30:      end if
31:      j ← j + 1
32:   end while
33:   i ← i + 1
34: end while
```

Ak chceme získať kvadratické koeficienty pre každý z polynómov zo sústavy, postupujeme podľa algoritmu 2. Prevodom kvadratických koeficientov HFE polynómu na kvadratické koeficienty príslušných polynómov zo sústavy, dostaneme vektor obsahujúci kvadratické koeficienty každého polynómu zo sústavy. Kvadratické koeficienty každého polynómu sú reprezentované maticou koeficientov z $GF(2)$. Výstupom je teda štruktúra `Vec<Vec<GF2>>`. Tento algoritmus postupne spracováva členy $A_{i,j}X^{2^i+2^j}$ z rovnice 2 a upravuje hodnoty $Q_1, ..., Q_n$. Každá z matíc Q_i je na začiatku nulová matica. Na riadku číslo 1 si teda pripravíme premennú $kvadraticke\_koeficienty$, predstavujúcu vektor nulových matíc. Pripravíme si premenné A a $temp$, typu `GF2E`, teda budú to prvky z poľa $GF(2^n)$. Hlavný cyklus prechádza hodnoty pre i od 0, kým 2^i je menšie ako stupeň polynómu HFE. Vnorený cyklus na riadku číslo 4 prechádza hodnoty pre j od i, kým 2^j je menšie ako stupeň polynómu HFE. Ak súčet $2^i + 2^j$ je väčší ako stupeň polynómu, cyklus sa preruší. Ak je súčet $2^i + 2^j$ menší ako stupeň polynómu HFE, do premennej A priradíme hodnotu koeficientu pri $X^{2^i+2^j}$ z polynómu HFE. Pomocou dvoch vnorených cyklov na riadku číslo 12 a 14 prechádzame indexy r a s. Na riadku číslo 15 vypočítame hodnotu $A * \alpha^{(r-1)*2^i+(s-1)*2^j}$ predstavujúcu nejaký prvok z poľa $GF(2^n)$ a priradíme ju do premennej $temp$. V cykle na riadku číslo 18 potom prechádzame reprezentáciu prvku $temp$, ak sa na jeho k-tej pozícii nachádza hodnota 1, pripočítame ju do premennej $kvadraticke\_koeficienty$ na pozíciu $k, r-1, s-1$. Inými slovami, pripočítame príspevok prvku ku kvadratickému koeficientu na pozícii $(r-1, s-1)$, teda $x_{r-1,s-1}$ v matici Q_k.

Algoritmus 4 Algoritmus prevodu HFE polynómu na sústavu polynómov

1: $absolutne\_koeficienty \leftarrow HFE.AbsolutneKoeficienty()$
2: $linearne\_koeficienty \leftarrow HFE.LinearneKoeficienty()$
3: $kvadraticke\_koeficienty \leftarrow HFE.KvadratickeKoeficienty()$
4: $i \leftarrow 0$
5: **while** $i <$ stupeň polynómu **do**
6: $\quad abs\_koeficient \leftarrow absolutne\_koeficienty[i]$
7: $\quad lin\_koeficient \leftarrow linearne\_koeficienty[i]$
8: $\quad kvad\_koeficient \leftarrow kvadraticke\_koeficienty[i]$
9: $\quad novy\_polynom \leftarrow Polynom(abs\_koeficient, lin\_koeficient, kvad\_koeficient)$
10: $\quad$ pridaj $novy\_polynom$ do sústavy polynómov
11: $\quad i \leftarrow i + 1$
12: **end while**

Algoritmus 4 zobrazuje prevod HFE polynómu na ekvivalentnú sústavu polynómov.

Keďže absolútny člen HFE polynómu je práve v tvare $c_1 * \alpha^0 + c_2 * \alpha^1 + ... + c_n * \alpha^{n-1}$, kde $c_1, ..., c_n$ sú prvky základného poľa $GF(2)$, na riadku číslo 1 priradíme koeficienty absolútneho člena HFE polynómu do vektora prvkov z $GF(2)$. Ak by teda sme teda mali napríklad absolútny člen HFE polynómu $C = \alpha^2 + \alpha + 1$, bol by reprezentovaný vektorom $(1, 1, 1)$. Tento vektor by sme teda priradili do premennej $absolutne\_koeficienty$ (typu `Vec<GF2>`), kde každá hodnota koeficientu na i-tej pozícii, predstavuje hodnotu absolútneho koeficientu i-tého polynóm zo sústavy.

Na riadku číslo 2 použijeme algoritmus 2, aby sme dostali vektor vektorov s lineárnymi koeficientami pre sústavu polynómov na ktorú HFE prevádzame.

Podobne na riadku číslo 3 použijeme algoritmus 3, aby sme dostali vektor matíc reprezentujúcich kvadratické koeficienty pre sústavu polynómov na ktorú HFE prevádzame.

Následne už len prechádzame v cykle (začínajúcom na riadku číslo 5) vektory s absolútnymi, lineárnymi a kvadratickými koeficientmi, pričom vytvoríme nový polynóm s príslušnými hodnotami z vektorov na i-tých pozíciach. Polynóm pridáme do sústavy polynómov. Takto vytvorená sústava polynómov je ekvivalentná polynómu s jednou neurčitou v HFE tvare.

9.4 Generovanie verejného kľúča HFE podpisovej schémy

Verejný kľúč HFE podpisovej schémy tvorí zobrazenie P, o ktorom vieme, že vzniká ako zloženie $P = S \circ P' \circ T$. Úlohou afinných transformácií S a T je zamaskovať zobrazenie P' predstavujúce centrálne zobrazenie. Aplikáciou afinnej transformácie T na centrálne zobrazenie P' nám vznikne sústava polynómov TP'. Následnou aplikáciou afinnej transformácie S na TP' dostaneme verejný kľúč.

9.4.1 Afinná transformácie T

Afinnú transformácie T aplikujeme na každý polynóm z centrálneho zobrazenia samostatne. Každý z polynómov centrálneho zobrazenia je reprezentovaný maticou kvadratických koeficientov Q_i, vektorom lineárnych koeficientov L_i a skalárom A_i reprezentujúcim absolútny člen polynómu. Ak máme polynóm $f_i(x_1, ..., x_n)$, pre vektor neurčitých $x = (x_1, ..., x_n)$ platí:

$$f_i(x_1, ..., x_n) = xQ_ix^T + L_ix^T + A_i$$

Majme maticu transformácie T, ktorú označíme ako A_T a vektor tejto transformácie, ktorý označíme ako b_T. Pre T zobrazí vektor $x = (x_1, ..., x_n)$ na:

$$T(x_1, ..., x_n) = xA_T + b_T$$

Aplikácia transformácie T na polynóm potom vyzerá nasledovne:

$$f_i(T(x)) = (xA_T + b_T)Q_i(xA_T + b_T)^T + L_i(xA_T + b_T)^T + A_i$$

Keďže transformáciu T aplikujeme na každý polynóm zvlášť, pri každom polynóme postupujeme podľa algoritmu 5.

Algoritmus 5 Afinná transformácia T

1: $novy\_polynom \leftarrow$ vytvor novy polynom
2: $novy\_polynom.KvadratickyKoeficient \leftarrow A_T * p.KvadratickyKoeficient * (A_T)^T$
3: $novy\_polynom.LinearnyKoeficient \leftarrow ((b_T * p.KvadratickyKoeficient * (A_T)^T) + (b_T * (p.KvadratickyKoeficient)^T * (A_T)^T) + (p.LinearnyKoeficient * (A_T)^T))$
4: $novy\_polynom.AbsolutnyKoeficient \leftarrow b_T * p.KvadratickyKoeficien * b_T + p.LinearnyKoeficient * b_T + p.AbsolutnyKoeficient$

Najprv si vytvoríme si premennú *novy_polynom*, ktorá reprezentuje polynóm po aplikovaní T. Na riadku číslo 2 mu do kvadratickej časti, teda do matice kvadratických členov, priradíme hodnotu podľa nasledovného vzorca:

$$A_T * Q_i * (A_T)^T$$

Len pre zopakovanie, A_T je transformačná matica transformácie T, Q_i je matica kvadratických koeficientov polynómu a $(A_T)^T$ je transponovaná matica A_T. Lineárne koeficienty polynómu nastavujeme na riadku číslo 3, podľa vzorca:

$$b_T * Q_i * (A_T)^T + b_T * (Q_i)^T * (A_T)^T + L_i * (A_T)^T$$

Absolútny koeficient polynómu nastavujeme na riadku číslo 4, podľa vzorca:

$$b_T * Q_i * (b_T)^T + L_i * (b_T)^T$$

Takto aplikujeme postupne T na každý polynóm zo sústavy.

9.4.2 Afinná transformácia S

[16] Afinná transformácia S sa aplikuje na výsledok predchádzajúcej transformácie T. Na rozdiel od T sa neaplikuje na každý polynóm samostatne, ale na celú sústavu polynómov $f'_1, ..., f'_m$, ktorá vznikla aplikáciou T. Transformácia S je reprezentovaná maticou A_S a vektorom b_S. Aplikujeme ju na sústavu $f'_1, ..., f'_m$ následovne:

$$S(f'_1, ..., f'_m) = (f'_1, ..., f'_m)A_S + b_S$$

Transformáciu S vytvoríme ako súčet lineárnej kombinácie polynómov $f'_1, ..., f'_m$ podľa matice A_S a transformačného vektora b_S. Nech polynómy $f'_1, ..., f'_m$ tvoria:

- $Q'_1, ..., Q'_m$,
- $L'_1, ..., L'_m$,
- $A'_1, ..., A'_m$.

Kvadratické koeficienty $Q''_1, ..., Q''_m$, lineárne koeficienty $L''_1, ..., L''_m$ a absolútne koeficienty $A''_1, ..., A''_m$ dostaneme ako:

$$Q''_i = \sum_{j=1}^{m} a_{j,i} Q'_j$$

$$L''_i = \sum_{j=1}^{m} a_{j,i} L'_j$$

$$A''_i = \sum_{j=1}^{m} a_{j,i} A'_j + b_i$$

kde $a_{j,i}$ je prvok z matice A_s na j-tom riadku v i-tom stĺpci. Funkcia ktorá celý tento proces vykonáva, sa v našej implementácii volá `Vec<Polynomial<T>> affineTransformationS`.

9.5 Generovanie verejného kľúča $HFE^{\hat{+}-}$ podpisovej schémy

Verejný kľúč pre $HFE^{\hat{+}-}$ generujeme rovnako, ako to bolo v prípade základnej schémy HFE s tým rozdielom, že vstupným parametrom nie je obyčajná sústava polynómov ktorá vznikla z HFE polynómu $H(x)$, ale sústava, ktorá vznikla z perturbovaného HFE polynómu $F(x) = H(x) + Q(x)$. Pre perturbáciu polynómov sústavy P' sme v triede `HFE` spravili funkciu `perturbation()` , ktorej vstupom sú:

- parameter `t`, určujúci koľko perturbačných polynómov sa vygeneruje
- parameter `modulus_deg`, predstavuje veľkosť n v matici $n \times n$, ktorá reprezentuje kvadratické koeficienty perturbačného polynómu
- parameter `system_of_polynomials`, predstavuje $p_1, ..., p_n$ zo P'
- parameter `perturbation_polynomials`, predstavuje $p'_1, ..., p'_n$
- parameter `betas`, predstavuje náhodne zvolené prvky $\beta_1, ..., \beta_n$ z $GF(2^n)$

Posledné dva parametre potrebujeme nakoľko predstavujú premenné, do ktorých uložíme hodnoty vygenerovaných perturbačných polynómov a koeficientov $\beta_1, ..., \beta_n$, ktoré budeme potrebovať pri inverzii pri generovaní podpisu. Sústavu perturbačných polynómov generujeme podľa algoritmu 6:

Algoritmus 6 Algoritmus pre generovanie sústavy polynómov centrálneho zobrazenia s perturbáciou $\hat{+}$

Require: vstupnými parametrami sú vektor hodnôt t, $beta\_koeficienty$, $perturbacne\_polynomy$, $velkost\_n$, $sustava\_polynomov$

1: $perturbovana\_sustava \leftarrow sustava\_polynomov$
2: $i \leftarrow 0$
3: **while** $i < t$ **do**
4: pridaj nahodnu maticu z GF2 do $perturbacne\_polynomy$
5: pridaj nahodny prvok z $GF(2^n)$ do $beta\_koeficienty$
6: $i \leftarrow i + 1$
7: **end while**
8: $i \leftarrow 0$
9: **while** $i < t$ **do**
10: $alfa \leftarrow beta\_koeficienty[i].KonverziaNaVektorHodnotGF(2)$
11: $k \leftarrow 0$
12: **while** $k < alfa.Dlzka()$ **do**
13: **if** $alfa[k] == 1$ **then**
14: $koef \leftarrow perturbovana\_sustava[k].KvadratickyKoeficient$
15: $koef \leftarrow koef + perturbovane\_polynomy[i]$
16: $perturbovana\_sustava[k].KvadratickyKoeficient \leftarrow koef$
17: **end if**
18: $k \leftarrow k + 1$
19: **end while**
20: $i \leftarrow i + 1$
21: **end while**

Na riadku číslo 1 si prichystáme premennú $perturbovana\_sustava$, do ktorej priradíme hodnotu základnej vygenerovanej sústavy polynómov. V cykle na riadku číslo 4 vygenerujeme náhodnú maticu prvkov z poľa $GF(2)$ predstavujúcu hodnoty koeficientov pre perturbačný polynóm. Túto maticu pridáme do vektora predstavujúceho sústavu perturbačných polynómov, teda do vstupného parametra $perturbacne\_polynomy$. Týmto spôsobom si vygenerované perturbačné polynómy ukladáme, keďže ich potrebujeme pre generovanie podpisu. Na riadku číslo 5 vygenerujeme náhodný koeficient β_i z $GF(2^n)$ a pridáme ho do vektora $beta\_koeficienty$. V cykle na riadku číslo 10 prevedieme koeficient β_i (prvok z $GF(2^n)$), na jeho ekvivalent v podobe vektora prvkov z $GF(2)$, ktorý priradíme do premennej $alfa$. Vo vnorenom cykle začínajúcom na riadku číslo 12, potom prechádzame

jednotlivé koeficienty tohto vektora. Ak je koeficient na k-tej pozícii rovný 1, pripočítame k matici kvadratických koeficientov na k-tej pozícii v premennej *perturbovana_sustava* hodnotu matice . Obsah premennej *perturbovana_sustava* po skončení cyklu na riadku číslo 21 predstavuje sústavu trapdooru zamaskovanú pomocou perturbačných polynómov.

9.6 Generovanie podpisu

Pre vygenerovanie podpisu sme si vytvorili triedu `Signature` a v nej tri funkcie pre vygenerovanie podpisu, podľa toho či šlo o HFE alebo $HFE^{\hat{+}-}$ schému a ktorá inverzia sa použila:

- `generateSignature` - pre vygenerovanie podpisu základnej HFE schémy
- `generateSignaturePerturbed` - pre vygenerovanie podpisu $HFE^{\hat{+}-}$ schémy s inverziou pomocou prehľadávanie všetkých možností
- `generateSignaturePerturbedProjection` - pre vygenerovanie podpisu $HFE^{\hat{+}-}$ schémy s inverziou pomocou projekcie

Najprv sme skúšali implementovať faktorizáciu pomocou Berlekampovho algoritmu [14], ale zistili sme, že faktorizovanie polynómov vyššieho stupňa pomocou tohto algoritmu nebolo veľmi efektívne. Preto sme sa rozhodli faktorizáciu implementovať pomocou Cantorovho a Zassenhausovho algoritmu [17]. Algoritmy pre vygenerovanie podpisu vyzerali následovne:

9.6.1 Generovanie podpisu pre HFE schému

Algoritmus 7 popisuje priebeh generovanie platného podpisu pre základnú HFE podpisovú schému pomocou funkcie `generateSignature`. Vstupnými parametrami sú:

- `signature`, premenná do ktorej uložíme vygenerovaný podpis
- `a`, parameter ktorý určuje koľko polynómov sa odstráni z verejného kľúča,
- `hfe`, predstavujúci HFE polynóm,
- `matrix_T`, parameter predstavujúci maticu transformácie T,
- `vector_T`, parameter predstavujúci vektor transformácie T,
- `matrix_S`, transformačná matica S,
- `vector_S`, parameter predstavujúci maticu transformácie S,
- `y_without_a`, predstavuje správu bez a počtu prvkov,

- `modulus_deg`, stupeň ireducibilného polynómu.

Generovanie podpisu postupuje podľa algoritmu 7. Na riadku číslo 2 najprv vezmeme správy y, ktorej doplníme počet a náhodných prvkov z $GF(2)$. Na riadku číslo 2 aplikujeme inverznú afinnú transformáciu na správu y, čím vznikne vektor z prvkov $GF(2)$ TP. Na riadku číslo 3 prevedieme TP na prvok z $GF(2^n)$ a priradíme ho do premennej Y, aby sme tento prvok mohli odrátať od HFE polynómu. Na riadku číslo 4 teda odrátame prvok Y od polynómu HFE, na výsledný polynóm aplikujeme funkciu ktorá ho prevedie na monický polynóm, aby sme naň mohli použiť Cantor-Zassenahausov faktorizačný algoritmus [17]. Aplikáciou tohto algoritmu dostaneme korene polynómu. V cykle na riadku číslo 6 postupne prechádzame korene polynómu, pričom ak je stupeň koreňu rovný 1, na riadku číslo 8 priraďujeme konštantný člen c do premennej X, potom nastavujeme premennú *existuje_koren* na hodnotu true a nakoniec cyklus ukončíme. Tento krok je dôležitý, nakoľko ak sa premenná *existuje_koren* nenastaví na hodnotu true, je implicitne nastavená na hodnotu false po prejdení všetkých koreňov. Ak táto funkcia vráti hodnotu false, znamená to, že sústava nemala riešenie. Táto situácia môže nastať, keďže sa nejedná o bijekciu. Podpis teda nemusí byť možné pre danú správu vygenerovať. V takom prípade sa táto funkcia zavolá znovu, opäť sa teda vygeneruje počet a náhodných prvkov z $GF(2)$ na riadku číslo 1 a cyklus pokračuje. Ak sa podarí nájsť koreň stupňa 1, na riadku číslo 16 prebehne konverzia prvku X z $GF(2^n)$ na vektor hodnôt z $GF(2)$, ktorý priradíme premennej x. Na riadku číslo 17 potom aplikujeme afinnú transformáciu T na vektor x, čím vlastne vygenerujeme platný podpis.

Algoritmus 7 Algoritmus pre generovanie podpisu pre HFE schému

1: dopln y poctom a nahodnych prvkov $GF(2)$
2: $TP \leftarrow$ aplikuj inverznu afinnu transformaciu S na y
3: $Y \leftarrow$ konverzia TP na prvok z $GF(2^n)$
4: $korene \leftarrow CantorZassenhaus(Monic(HFE - Y))$
5: $existuje\_koren \leftarrow false$
6: **for** c in $korene$ **do**
7: **if** stupen $c == 1$ **then**
8: $X \leftarrow$ konstantny clen z c
9: $existuje\_koren \leftarrow true$
10: **break**
11: **end if**
12: **end for**
13: **if** $existuje\_koren == false$ **then**
14: **return** $false$
15: **end if**
16: $x \leftarrow$ konverzia X na vektor prvkov z $GF(2)$
17: $podpis \leftarrow$ aplikuj inverznu afinnu transformaciu T na x
18: **return** $true$

9.6.2 Generovanie podpisu pre $HFE^{\hat{+}-}$ schému s inverziou cez prehľadávanie všetkých možností

Generovanie podpisu pre $HFE^{\hat{+}-}$ s inverziou cez prehľadávanie všetkých možností postupuje podľa algoritmu 8. Podobne ako pri generovaní klasického HFE podpisu, najprv vezmeme správu y, ktorej doplníme počet a náhodných prvkov z $GF(2)$ a aplikujeme inverznú afinnú transformáciu na správu y, čím vznikne vektor z prvkov $GF(2)$ TP. Na riadku číslo 3 nastavíme premennú *existuje_koren* na hodnotu false. Na riadku číslo 4 si vygenerujeme všetky možné vektory t hodnôt z poľa $GF(2^n)$ a tieto vektory priradíme do premennej *vektory*. Následne prechádzame vektory z premennej *vektory*. Práve prechádzaný vektor sa nachádza v premennej *t_hodnoty*. Cyklus na riadku číslo 7 zabezpečí pripočítanie $\sum_{i=1}^{t} \beta_i {p'}_i$, kde za ${p'}_i$ dosádzame hodnoty z vektora *t_hodnoty*, ku HFE polynómu. Získame teda polynóm nízkeho stupňa $H(x) + \beta_1 y_1 + ... + \beta_t y_t$, ktorý je možné invertovať. Ďalej pokračujeme ako pri klasickom HFE generovaní podpisu. Na riadku číslo 11 prevedieme TP na prvok z $GF(2^n)$ a priradíme ho do premennej Y, aby sme tento prvok mohli odrátať od HFE polynómu. Po odrátaní na výsledný polynóm aplikujeme funkciu ktorá ho prevedie na monický polynóm, aby sme naň mohli použiť Cantor-Zassenahausov

faktorizačný algoritmus [17]. Aplikáciou tohto algoritmu dostaneme korene polynómu. V cykle na riadku číslo 13 postupne prechádzame korene polynómu, pričom ak je stupeň koreňu rovný 1, na riadku číslo 15 priraďujeme konštantný člen c do premennej X. Tento prvok X z rozšírenia poľa prevedieme na vektor prvkov z $GF(2)$. Na riadku číslo 18 prechádzame hodnoty od 0 po t a vytvárame vektor nazvaný *perturbované_korene*, ktorého hodnoty sú vlastne hodnoty dosadeného x'_i do $p'_i(x)$. Na riadku číslo 22 teda overujeme, či platí že pre všetky $i = 1, .., t$ je $p'_i(x') = y_i$, a teda či x' je hľadanou inverziou $F(x) = y$. Ak je táto podmienka splnená, nastavíme premennú *existuje_koren* na true a cyklus ukončíme, pretože sme už našli platné riešenie. Ak by *existuje_koren* bolo false, cyklus by sa ukončil a funkcia by sa musela zavolať znovu. Ak je ale *existuje_koren* true, na riadku číslo 35 vypočítame platný podpis aplikovaním afinnej transformácie T na *vektor_X*.

Algoritmus 8 Algoritmus pre generovanie podpisu pre $HFE^{\hat{+}-}$ schému s inverziou cez prehľadávanie všetkých možností

1: doplň y poctom a nahodnych prvkov $GF(2)$
2: $TP \leftarrow$ aplikuj inverznu afinnu transformaciu S na y
3: $existuje\_koren \leftarrow false$
4: $vektory \leftarrow$ vygeneruj vsetky mozne vektory t hodnot
5: **for** $t\_hodnoty$ in $vektory$ **do**
6: $i \leftarrow 0$
7: **while** $i < t$ **do**
8: $HFE \leftarrow HFE + betas[i] * t\_hodnoty[i]$
9: $i \leftarrow i + 1$
10: **end while**
11: $Y \leftarrow$ konverzia TP na vektor prvkov z $GF(2^n)$
12: $korene \leftarrow CantorZassenhaus(Monic(HFE - Y))$
13: **for** c in $korene$ **do**
14: **if** stupen $c == 1$ **then**
15: $X \leftarrow$ konstantny clen z c
16: $vektor\_X \leftarrow$ konverzia X na vektor prvkov z $GF(2)$
17: $i \leftarrow 0$
18: **while** $i < t$ **do**
19: $perturbovane\_korene[i] = vektor\_X * pert\_polynomy[i] * vektor\_X$
20: $i \leftarrow i + 1$
21: **end while**
22: **if** $perturbovane\_korene == t\_hodnoty$ **then**
23: $existuje\_koren \leftarrow true$
24: **break**
25: **end if**
26: **end if**
27: **end for**
28: **if** $existuje\_koren == true$ **then**
29: **break**
30: **end if**
31: **end for**
32: **if** $existuje\_koren == false$ **then**
33: **return** $false$
34: **end if**
35: $podpis \leftarrow$ aplikuj inverznu afinnu transformaciu T na $vektor\_X$

9.6.3 Generovanie podpisu pre $HFE^{\hat{+}-}$ schému s inverziou cez projekciu

Generovanie podpisu pre $HFE^{\hat{+}-}$ s inverziou cez prehľadávanie všetkých možností postupuje podľa algoritmu 9. Opäť najprv vezmeme správu y, ktorej doplníme počet a náhodných prvkov z $GF(2)$ a aplikujeme inverznú afinnú transformáciu na správu y, čím vznikne vektor z prvkov $GF(2)$ TP. Na riadku číslo 3 si vygenerujeme všetky možné vektory t hodnôt z poľa $GF(2^n)$ a tieto vektory priradíme do premennej $vektory$. Následne na riadku číslo 4 vypočítame všetky lineárne kombinácie $\beta_1, ..., \beta_t$ pre $vektory$ a hodnoty uložíme do vektora $linearne\_koeficienty$. Na riadku číslo 5 vypočítame projekčný polynóm pomocou NTL funkcie `BuildFromRoots()`. `BuildFromRoots()` je funkcia z knižnice NTL, ktorá na základe vektora prvkov poľa $GF(2^n)$ zostrojí polynóm, ktorého korene sú práve prvky daného vektora. Do premennej $TP\_projekcia$ priradíme hodnotu polynómu pre hodnoty TP. V cykle potom prechádzame hodnoty pre i od 0, kým i je menšie ako stupeň projekčného polynómu. Ak práve prechádzaný koeficient z projekčného polynómu nie je nulový, potom k $hfe\_projekcia$ pripočítame tento prechádzaný koeficient vynásobený s neurčitou V HFE polynóme na i-tej pozícii. Na riadku číslo 14 od polynómu $hfe\_projekcia$ odpočítame polynóm $TP\_projekcia$ a na výsledný polynóm aplikujeme funkciu $MakeMonic$ z NTL knižnice, aby sme dostali monický polynóm, ktorý môžeme faktorizovať pomocou Cantor Zassenhausovho algoritmu. Na riadku číslo 16 prechádzame korene polynómu $\Pi_t(H(x)) - \Pi_t(y)$, ktoré sme dostali faktorizáciou. Ak je práve prechádzaný koreň stupňa 1, konvertujeme konštantný člen c, reprezentovaný prvkom z rozšírenia, na vektor hodnôt zo základného poľa. Potom v cykle na riadkoch 21 až 28 overujeme, či je koreň platným riešením perturbačných polynómov. Teda overujeme ktoré x' z množiny nájdených x', je také, že platí $F(x') = y$. Po nájdení takého x' nastavíme premennej $existuje\_koren$ hodnotu true. Ak by táto premenná mala hodnotu false, cyklus by sa ukončil a funkcia by sa musela zavolať znovu. Ak je ale $existuje\_koren$ rovný true, na riadku číslo 36 vypočítame platný podpis aplikovaním afinnej transformácie T na vektor $vektor\_X$.

Algoritmus 9 Algoritmus pre generovanie podpisu pre $HFE^{\hat{+}-}$ schému s inverziou cez projekciu

1: dopln y poctom a nahodnych prvkov $GF(2)$
2: $TP \leftarrow$ aplikuj inverznu afinnu transformaciu S na y
3: $vektory \leftarrow$ vygeneruj vsetky mozne vektory t hodnot
4: $linearne\_kombinacie \leftarrow$ vypocitaj vsetky mozne linearne kombinacie koeficientov $\beta_1, ..., \beta_t$ pre $vektory$
5: $projekcny\_polynom \leftarrow BuildFromRoots(linearne\_kombinacie)$
6: $TP\_projekcia \leftarrow$ vyhodnot $projekcny\_polynom$ pre hodnoty TP
7: $i \leftarrow 0$
8: **while** $i <=$ stupen $projekcny\_polynom$ **do**
9: **if** $projekcny\_polynom[i].koeficient! = 0$ **then**
10: $hfe\_projekcia \leftarrow hfe\_projekcia + projekcny\_polynom[i].koeficient * HFE^i$
11: **end if**
12: $i \leftarrow i + 1$
13: **end while**
14: $hfe\_projekcia \leftarrow hfe\_projekcia - TP\_projekcia$
15: $korene \leftarrow CantorZassenhaus(Monic(hfe\_projekcia))$
16: **for** c in $korene$ **do**
17: **if** stupen $c == 1$ **then**
18: $X \leftarrow$ konstantny clen z c
19: $vektor\_X \leftarrow$ konverzia X na vektor prvkov z $GF(2)$
20: $i \leftarrow 0$
21: **while** $i <$ stupen polynomu **do**
22: $pert\_korene[i] \leftarrow vektor\_X * pert\_polynomy[i].KvadratickyKoeficient * vektor\_X$
23: $pert\_korene[i] \leftarrow pert\_korene[i] + vektor\_X * pert\_polynomy[i].LinearnyKoeficient$
24: $pert\_korene[i] \leftarrow pert\_korene + pert\_polynomy[i].AbsolutnyKoeficient$
25: **if** $pert\_korene == TP$ **then**
26: $existuje\_koren \leftarrow true$
27: **break**
28: **end if**
29: $i \leftarrow i + 1$
30: **end while**
31: **end if**
32: **end for**
33: **if** $existuje\_koren == false$ **then**
34: **return** $false$
35: **end if**
36: $podpis \leftarrow$ aplikuj inverznu afinnu transformaciu T na $vektor\_X$

9.7 Overenie podpisu

Pre overenie podpisu sme vytvorili funkciu `verifySignature`, so vstupnými parametrami:

- `a` - parameter modifikátora "mínus", určujúci koľko polynómov sa odstráni z verejného kľúča,
- `signature` - obsahuje podpis ktorý chceme overiť,
- `message` - obsahuje správu, pre ktorú bol vygenerovaný podpis, ktorý chceme overiť,
- `public_key` - verejný kľúč pomocou ktorého overujeme podpis,
- `modulus_deg` - určuje počet polynómov pred použitím modifikátora $-$.

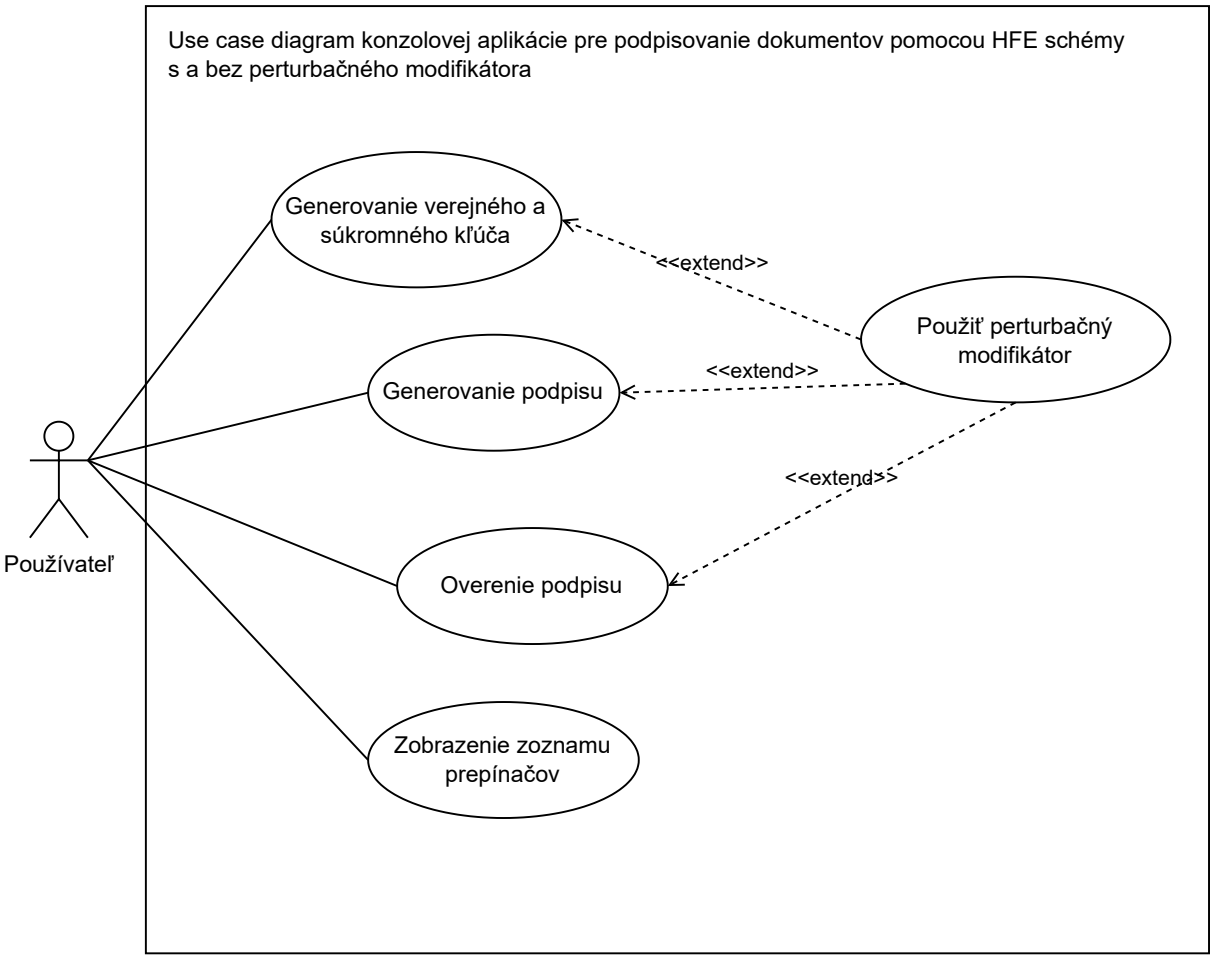
Algoritmus 10 Algoritmus pre overenie podpisu

1: $i \leftarrow 0$
2: **while** $i < stupen\ polynomu - a$ **do**
3: $\quad hodnoty\_PK[i] \leftarrow public\_key[i].AbsolutnyKoeficient$
4: $\quad hodnoty\_PK[i] \leftarrow PK[i] + public\_key[i].LinearnyKoeficient * podpis + podpis * public\_key[i].KvadratickyKoeficient * podpis$
5: $\quad i \leftarrow i + 1$
6: **end while**
7: **if** $sprava == hodnoty\_PK$ **then**
8: $\quad$ **return** $true$
9: **else**
10: $\quad$ **return** $false$
11: **end if**

Funkcia `verifySignature()` postupuje podľa algoritmu 10. V cykle prechádzame hodnoty pre i od 0 po veľkosť *stupňa polynómu* od ktorého odpočítame parameter a. To znamená že ignorujeme posledných a prvkov, keďže vieme, že tieto prvky boli náhodne vygenerované. V cykle prechádzame hodnoty verejného kľúča na i-tej pozícii a do premennej $hodnoty\_PK[i]$ postupne najprv uložíme absolútny člen, potom k nemu prirátame hodnoty vynásobenú hodnotu lineárnych koeficientov s vektorom podpisu a na záver prirátame aj súčin vektora podpisu s maticou kvadratických koeficientov verejného kľúča a transponovaného vektora podpisu. Na riadku číslo 7 overujeme či sa správa rovná hodnotám po aplikovaní verejného kľúča na podpis. Ak sa rovnajú, podpis je platný. Ak by sa nerovnali, podpis by bol neplatný.

9.8 Konzolová aplikácia

Súčasťou implementácie je aj konzolová aplikácia, ktorá umožňuje vygenerovať kľúče, následne podpísať dokument a overiť tento vygenerovaný podpis. Jednotlivé časti sa spúšťajú pomocou prepínačov, viac o ich použití je v prílohe B. Nakoľko implementácia $HFE^{\hat{+}-}$ schémy s inverziou cez projekciu sa v experimentoch ukázala ako neefektívna, rozhodli sme sa v konzolovej aplikácii implementovať iba podpisovú schému pre HFE a $HFE^{\hat{+}-}$ schémy s inverziou cez prehľadávanie všetkých možností. Funkcionálne požiadavky pre jednoduchú konzolovú aplikáciu sú definované na obrázku 3.



Obr. 3: Funkcionálne požiadavky na konzolovú aplikáciu pre podpisovú HFE schému s/bez perturbačného modifikátora.

10 Experimenty

V tejto kapitole si zhrnieme postup pri testovaní našej implementácie. Cieľom diplomovej práce bolo implementovať podpisovú schému HFE a $HFE^{\hat{+}-}$, porovnať tieto implementácie a vyhodnotiť výpočtovú náročnosť. Implementovali sme obe schémy, pričom pre schému $HFE^{\hat{+}-}$ boli implementované obe možné inverzie z kapitoly 6.

Konečné riešenie bolo vytvorené v release móde s použitím najvyššej úrovne optimalizácie kompilátora. Všetky testy bežali na počítači s 8-jadrovým procesorom AMD Ryzen 9 5900HS, so 16 GB RAM a s operačným systémom Windows.

10.1 Testovanie odporúčaných parametrov

V [13] autori navrhli optimálne parametre pre 128-bitové zabezpečenie: $q = 2$, $n = 263$, $d = 65$, $t = 6$, $a = 7$. Naším cieľom bolo porovnať klasickú schému HFE a schému HFE s novým perturbačným modifikátorom v zmysle výpočtovej náročnosti. Zmerali sme čas, ktorý procesor strávil vykonávaním nasledujúcich procesov:

- generovanie verejného kľúča
- generovanie platného podpisu
- overenie platnosti podpisu

a počet iterácií pre počet pokus pre vygenerovanie platného podpisu. Generovanie/overovanie pre jednotlivé nastavenia parametra $t = 0...1$, sme spúšťali pre 100 iterácií.

Použili sme odporúčané parametre pre q, n, a, d a t z [13], ale parameter t sme testovali pre hodnoty od 0 do 6. To znamená, že sme menili hodnotu parametra t, ktorý nastavuje vygenerovaný počet perturbačných polynómov, ktoré sa pripočítajú k sústave polynómov centrálneho zobrazenia HFE. Všetky výsledky pre schému HFE bez modifikátora si môžete pozrieť v tabuľke 1. Výsledky pre $HFE^{\hat{+}-}$ s použitím dvoch rôznych inverzií nájdete v tabuľke 2 a tabuľke 3.

Pretože pri generovaní platného podpisu mohla nastať situácia, kedy sústava polynómov nemala riešenie, bolo potrebné znovu vygenerovať náhodne zvolené hodnoty z F_q podľa parametra a. Priemerný počet iterácií, ktoré boli potrebné pre vygenerovanie platného podpisu sú taktiež zahrnuté v tabuľke, nachádzajú sa v stĺpci s názvom Pokusy.

Vykonané testy ukazujú, že so zvyšujúcim sa parametrom t, sa zvyšuje aj čas potrebný na vygenerovanie platných podpisov pre schému $HFE^{\hat{+}-}$ s oboma inverziami. Zatiaľ čo pre klasickú HFE schému sa čas, za ktorý sa vygeneroval platný podpis pre hodnoty $t = 0, ..., 6$

takmer nemenil, v tabuľke pre $HFE^{\hat{+}-}$ s inverziou cez prehľadávanie všetkých možností bol nárast hodnoty priemeru z 0,378 sekundy pre $t = 0$ na 17,190 sekúnd pre $t = 6$. Priemerná hodnota samozrejme závisí aj od počtu pokusov, potrebných pre vygenerovanie platného podpisu. Faktom ale ostáva, že inverzia $HFE^{\hat{+}-}$ schémy trvá podstatne dlhšie ako inverzia základnej HFE schémy.

Ako môžeme vidieť v tabuľke 3, pre parameter $t = 4$ bol priemerný čas na vygenerovanie platného podpisu pre $HFE^{\hat{+}-}$ s inverziou cez projekciou rovný 204,389s. Jednalo sa o veľký nárast časovej zložitosti. Takéto generovanie podpisov by nebolo použiteľné v reálnom svete, preto sme sa rozhodli netestovať $HFE^{\hat{+}-}$ s inverziou cez projekciou pre $t = 5$ a $t = 6$.

Pokiaľ ide o merania času potrebného pre vygenerovanie verejného kľúča pre klasické HFE a $HFE^{\hat{+}-}$, namerané výsledky ukázali, že pridanie vygenerovanie perturbačných polynómov a ich pripočítanie k sústave centrálneho zobrazenia nespôsobuje takmer žiadny nárast času.

Podobne ako v prípade času potrebného pre vygenerovanie verejného kľúča pre HFE schému a $HFE^{\hat{+}-}$ schému, ani v prípade času potrebného pre overenie platnosti podpisu nehralo zavedenie perturbačného modifikátora žiadnu úlohu.

Z výsledkov teda jasne vidno, že implementácia $HFE^{\hat{+}-}$ s inverziou cez projekciu nie je efektívna a že pre odporúčané parametre $q = 2$, $n = 263$, $r = 7(d = 65)$, $t = 6$, $a = 7$ zavedenie perturbačného modifikátora pre generovanie podpisu znamená časový nárast v priemere o 17 sekúnd.

	`Generovanie kľúča`	`Generovanie podpisu`	`Pokusy`	`Overenie podpisu`
$t = 0$	6.101 ± 0.124	0.390 ± 0.217	1.640 ± 0.893	0.006 ± 0.007
$t = 1$	6.245 ± 0.150	0.417 ± 0.262	1.720 ± 1.064	0.006 ± 0.007
$t = 2$	6.277 ± 0.178	0.402 ± 0.261	1.680 ± 1.110	0.005 ± 0.007
$t = 3$	5.058 ± 0.093	0.305 ± 0.208	1.580 ± 1.093	0.005 ± 0.007
$t = 4$	7.215 ± 2.054	0.439 ± 0.285	1.560 ± 0.820	0.007 ± 0.007
$t = 5$	6.319 ± 0.188	0.380 ± 0.200	1.580 ± 0.854	0.005 ± 0.007
$t = 6$	6.084 ± 0.555	0.362 ± 0.221	1.560 ± 0.946	0.006 ± 0.007

Tabuľka 1: Priemerné hodnoty namerané zo 100 spustení (spolu so štandardnou odchýlkou) pre HFE bez perturbačného modifikátora. Hodnoty sú udávané v sekundách (okrem `Pokusy`, tieto hodnoty reprezentujú počet pokusov o vygenerovanie platného podpisu).

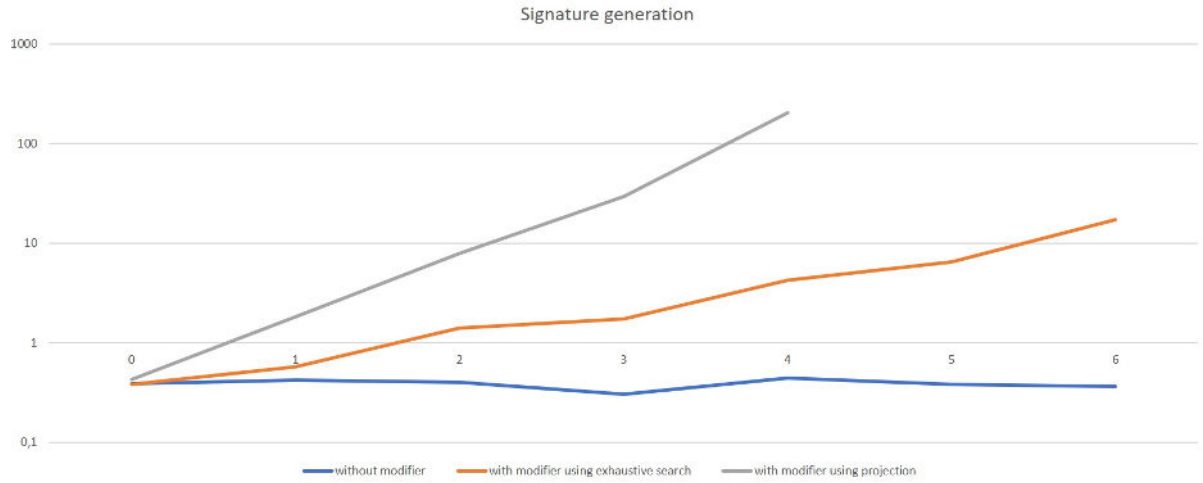
Rozdiel v náraste času potrebného na vygenerovanie platného podpisu pre 3 testované verzie HFE (HFE, $HFE^{\hat{+}-}$ s obomi inverziami) je možné vidieť na obrázku 4.

	`Generovanie kľúča`	`Generovanie podpisu`	`Pokusy`	`Overenie podpisu`
$t = 0$	6.130 ± 0.129	0.378 ± 0.257	1.610 ± 1.090	0.008 ± 0.007
$t = 1$	6.293 ± 0.133	0.573 ± 0.416	1.510 ± 0.810	0.006 ± 0.007
$t = 2$	6.326 ± 0.173	1.411 ± 1.264	1.920 ± 1.330	0.006 ± 0.007
$t = 3$	5.078 ± 0.083	1.738 ± 1.710	1.630 ± 1.050	0.005 ± 0.007
$t = 4$	7.286 ± 2.070	4.267 ± 3.994	1.490 ± 0.731	0.006 ± 0.007
$t = 5$	6.388 ± 0.192	6.514 ± 6.535	1.410 ± 0.766	0.005 ± 0.007
$t = 6$	6.155 ± 0.555	17.190 ± 15.557	1.770 ± 1.135	0.006 ± 0.007

Tabuľka 2: Priemerné hodnoty namerané zo 100 spustení (spolu so štandardnou odchýlkou) pre HFE s perturbačným modifikátorom s použitím inverzie cez prehľadávanie všetkých možností. Hodnoty sú udávané v sekundách (okrem `Pokusy`, tieto hodnoty reprezentujú počet pokusov o vygenerovanie platného podpisu).

	`Generovanie kľúča`	`Generovanie podpisu`	`Pokusy`	`Overenie podpisu`
$t = 0$	6.130 ± 0.129	0.429 ± 0.254	1.740 ± 1.050	0.006 ± 0.007
$t = 1$	6.293 ± 0.133	1.825 ± 1.023	1.610 ± 0.908	0.007 ± 0.007
$t = 2$	6.326 ± 0.173	7.894 ± 4.745	1.520 ± 0.915	0.007 ± 0.007
$t = 3$	5.078 ± 0.083	29.444 ± 16.472	1.610 ± 0.897	0.004 ± 0.007
$t = 4$	7.286 ± 2.070	204.389 ± 142.071	1.700 ± 1.184	0.008 ± 0.007
$t = 5$	6.388 ± 0.192	*	*	*
$t = 6$	6.155 ± 0.555	*	*	*

Tabuľka 3: Priemerné hodnoty namerané zo 100 spustení (spolu so štandardnou odchýlkou) pre HFE s perturbačným modifikátorom s použitím oboch metód inverzie cez projekciu). Hodnoty sú udávané v sekundách (okrem `Pokusy`, tieto hodnoty reprezentujú počet pokusov o vygenerovanie platného podpisu). * reprezentuje vynechané testy z dôvodu výpočtovej náročnosti



Obr. 4: Reprezentácia času ktorý procesor strávil generovaním platného podpisu HFE schémy s/bez perturbačného modifikátora.

10.2 Nové parametre

Okrem testovania nárastu časovej zložitosti so zvyšujúcou sa hodnotou parametra t, sme sa rozhodli ešte otestovať trade-off medzi parametrami t a d, teda medzi rozmerom perturbácie a stupňom HFE polynómu. Testovali sme dvojice hodnôt t a d:

- $t = 5$, $d = 129$
- $t = 8$, $d = 33$

Ostatné parametre z odporúčaných parametrov ostali nezmenené. Výsledky meraní pre klasickú HFE schému s týmito nastaveniami sú v tabuľke 4. Výsledky meraní pre klasickú $HFE^{\hat{+}-}$ schému s týmito nastaveniami sú v tabuľke 5.

	`Generovanie kľúča`	`Generovanie podpisu`	`Pokusy`	`Overenie podpisu`
$d = 129, t = 5$	6.364 ± 0.095	1.336 ± 0.802	1.470 ± 0.881	0.005 ± 0.007
$d = 33, t = 8$	4.282 ± 0.098	0.079 ± 0.047	1.580 ± 0.976	0.006 ± 0.008

Tabuľka 4: Priemerné hodnoty namerané zo 100 spustení (spolu so štandardnou odchýlkou) pre klasické HFE. Hodnoty sú udávané v sekundách (okrem `Pokusy`, tieto hodnoty reprezentujú počet pokusov o vygenerovanie platného podpisu).

Podobne ako pri odporúčaných parametroch, ani pre tieto parametre nebola implementácia s $HFE^{\hat{+}-}$ schémou s inverziou cez projekciu efektívna, preto sme sa rozhodli ju netestovať.

	Generovanie kľúča	Generovanie podpisu	Pokusy	Overenie podpisu
$d = 129, t = 5$	6.418 ± 0.105	31.982 ± 32.880	1.730 ± 1.053	0.004 ± 0.008
$d = 33, t = 8$	4.350 ± 0.086	11.823 ± 12.421	1.54 ± 0.979	0.006 ± 0.008

Tabuľka 5: Priemerné hodnoty namerané zo 100 spustení (spolu so štandardnou odchýlkou) pre HFE s perturbačným modifikátorom s použitím inverzie cez prehľadávanie všetkých možností. Hodnoty sú udávané v sekundách (okrem `Pokusy`, tieto hodnoty reprezentujú počet pokusov o vygenerovanie platného podpisu).

Z výsledkov v tabuľke 4 vidíme, že už samotné nastavenie parametrov na $t = 5$, $d = 129$ malo vplyv na čas potrebný pre vygenerovanie podpisu. Usudzujeme tak na základe toho, že stúpol čas potrebný pre vygenerovanie podpisu pomocou základnej HFE schémy. Priemerný nameraný čas pre odporúčané $t = 6$, $d = 65$ bol 0,362s a pre $t = 5$, $d = 129$ bol 1,336s. Preto nás neprekvapil nárast času potrebného pre generovanie podpisu pre schému $HFE^{\hat{+}-}$. Zatiaľ čo vygenerovanie pre odporúčané parametre trvalo 17,190s, pre $t = 5$, $d = 129$ to bolo až 31,982s.

Prekvapivé výsledky prinieslo testovanie parametrov $t = 8$, $d = 33$. Čas potrebný na vygenerovanie platného podpisu pre základnú HFE schému bol 0,079s. Je to menej ako 0,362s potrebných pre odporúčané $t = 6$, $d = 65$. Aj priemerný čas pre vygenerovanie kľúča bol o niečo nižší. Pre odporúčané parametre to bolo 6,084s a pre nové parametre 4,282s. Meranie času pre $HFE^{\hat{+}-}$ schému dopadlo podobne. Namerané hodnoty boli opäť nižšie. Čas potrebný pre vygenerovanie platného podpisu bol v priemere 11,823s, čo je o 5s menej ako v prípade odporúčaných parametrov, avšak táto hodnota závisí od počtu pokusov vygenerovania platného podpisu.

		Minimum	Maximum
$d = 65$	$t = 6$	0.453	64.703
$d = 129$	$t = 5$	0.859	140.578
$d = 33$	$t = 8$	0.188	74.859

Tabuľka 6: Maximálna a minimálna hodnota pre vygenerovanie podpisu namerané zo 100 spustení pre HFE s perturbačným modifikátorom s použitím inverzie cez prehľadávanie všetkých možností. Hodnoty sú udávané v sekundách.

V tabuľke 6 sa nachádza porovnanie minimálnej a maximálnej nameranej hodnoty času pre vygenerovanie platného podpisu $HFE^{\hat{+}-}$ schémy s inverziou cez prehľadávanie

všetkých možností. Najnižšia nameraná hodnota pre vygenerovanie podpisu bola 0,188s. Táto hodnota bola nameraná pre parametre $t = 8$, $d = 33$. Najvyššia nameraná hodnota pre vygenerovanie podpisu bola 140,578s. Táto hodnota bola nameraná pre parametre $t = 5$, $d = 129$.

Záver

V tejto práci sme sa zamerali na meranie výpočtovej zložitosti nového perturbačného modifikátora prezentovaného v [13], ktorý by mal zvýšiť bezpečnosť HFE schémy. Naštudovali a implementovali sme HFE schému a $HFE^{\hat{+}-}$ schému s modifikáciami, pričom sme implementovali a porovnali aj dve metódy inverzie navrhnuté v [13]. Naším cieľom bolo otestovať, ako pridanie tohto nového perturbačného modifikátora ovplyvní časovú zložitosť generovania verejného kľúča, generovania platného podpisu a overenia podpisu. Podľa našich zistení má implementácia $HFE^{\hat{+}-}$ s inverziou pomocou projekcie taký nárast časovej zložitosti, že by nebola reálne použiteľná v reálnom svete, keďže generovanie podpisu by mohlo trvať v najhoršom prípade viac ako 346 sekúnd len pre parameter $t = 4$. Autori článku [13] uvádzajú, že pre nimi odporúčané parametre, generovanie podpisu trvá 10 sekúnd, avšak neuvádzajú o akú implementáciu sa jedná, či používali inverziu cez projekciu alebo inverziu cez prehľadávanie všetkých možností, prípadne aké optimalizácie ešte vykonali. Pre našu implementáciu $HFE\hat{+}-$ schémy s inverziou cez prehľadávanie všetkých možností, trvalo generovanie podpisu v priemere cca 17 sekúnd. Z nameraných hodnôt pre generovanie verejného kľúča a overenie platnosti vyplýva, že zavedenie perturbačného modifikátora neprináša takmer žiadny nárast času ich trvania. Okrem odporúčaných parametrov sme hľadali kompromis medzi rýchlosťou a bezpečnosťou, keďže zvýšenie rýchlosti vedie k zníženiu bezpečnosti a naopak. Zaujímavé bolo vymenenie odporúčaných parametrov $t = 6$ a $d = 65$ za $t = 8$ a $d = 33$. Teda situácia, kedy sme ubrali z hodnoty parametra, ktorý určuje stupeň HFE polynómu a pridali sme na hodnote parametra, ktorý určuje rozmer perturbácie. Generovanie podpisu pri takto nastavených parametroch pre $HFE\hat{+}-$ schému s inverziou cez prehľadávanie všetkých možností, trvalo v priemere cca 12 sekúnd. Takto nastavené parametre by teda mohli priniesť potencionálne zníženie času potrebného pre vygenerovanie platného podpisu pomocou $HFE\hat{+}-$ schémy, so zachovaním úrovne bezpečnosti.

Zoznam použitej literatúry

1. SHOR, Peter W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*. 1999, roč. 41, č. 2, s. 303–332.
2. RIVEST, Ronald L, SHAMIR, Adi a ADLEMAN, Leonard. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*. 1978, roč. 21, č. 2, s. 120–126.
3. BRUCE, Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C.-2nd*. john wiley & sons, 1996.
4. LIDL, Rudolf a NIEDERREITER, Harald. *Introduction to finite fields and their applications*. Cambridge university press, 1994.
5. WOLF, Christopher a PRENEEL, Bart. Taxonomy of public key schemes based on the problem of multivariate quadratic equations. *Cryptology ePrint Archive*. 2005.
6. SAMUEL, Pierre a LEVY, Silvio. *Projective geometry*. Zv. 14. Springer, 1988.
7. STRANG, Gilbert. *Linear algebra and its applications*. Belmont, CA: Thomson, Brooks/Cole, 2006.
8. STINSON, Douglas Robert a PATERSON, Maura. *Cryptography: theory and practice*. CRC press, 2018.
9. EKERT, Artur a JOZSA, Richard. Quantum computation and Shor's factoring algorithm. *Reviews of Modern Physics*. 1996, roč. 68, č. 3, s. 733.
10. HROMADA, Viliam. Úvod do MQ kryptosystémov. 2022.
11. PATARIN, Jacques. Hidden fields equations (HFE) and isomorphisms of polynomials (IP): Two new families of asymmetric algorithms. In: *Advances in Cryptology—EUROCRYPT'96: International Conference on the Theory and Application of Cryptographic Techniques Saragossa, Spain, May 12–16, 1996 Proceedings 15*. Springer, 1996, s. 33–48.
12. MATSUMOTO, Kohji. Value-distribution of zeta-functions. In: *Analytic Number Theory: Proceedings of the Japanese-French Symposium held in Tokyo, Japan, October 10–13, 1988*. Springer, 1990, s. 178–187.
13. FAUGÈRE, Jean-Charles, PATARIN, Jacques, PERRET, Ludovic et al. A New Perturbation for Multivariate Public Key Schemes such as HFE and UOV. *Cryptology ePrint Archive*. 2022.

14. BERLEKAMP, Elwyn R. Factoring polynomials over large finite fields. *Mathematics of computation*. 1970, roč. 24, č. 111, s. 713–735.

15. TAO, Chengdong, PETZOLDT, Albrecht a DING, Jintai. Efficient key recovery for all HFE signature variants. In: *Advances in Cryptology–CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part I 41*. Springer, 2021, s. 70–93.

16. HROMADA, Viliam. Pár slov k implementácii MQ. 2022.

17. CANTOR, David G a ZASSENHAUS, Hans. A new algorithm for factoring polynomials over finite fields. *Mathematics of Computation*. 1981, roč. 36, č. 154, s. 587–592.

Prílohy

A Štruktúra elektronického nosiča

Príloha obsahuje nasledujúce súbory a adresáre:

- **dp.pdf** - diplomová práca vo formáte PDF.
- **merania/** - obsahuje výsledky meraní vo formáte .xlsx
- **src/** - adresár so zdrojovými kódmi aplikácie
- **HFE_APP.exe** - spustiteľná aplikácia

B Používateľská príručka

Súčasťou tejto práce bola aj implementácia konzolovej aplikácie, pomocou ktorej je možné vygenerovať verejný a súkromný kľúč pre podpisové schémy HFE a $HFE^{\hat{+}-}$. V prípade, že už disponujeme príslušným verejným a súkromným kľúčom (vygenerovanými touto aplikáciou), môžeme podpísať zvolený dokument. Ak máme k dispozícii vygenerovaný podpis, príslušný dokument, pre ktorý bol tento podpis vygenerovaný a príslušný verejný kľúč, potom môžeme overiť platnosť daného podpisu.

Všetky vyššie spomenuté úkony vieme spustiť pomocou príslušných prepínačov:

- `-h` pomocou tohto prepínača vieme zobraziť všetky definované prepínače aj s ich popisom,
- `-k` slúži na spustenie generovania súkromného a verejného kľúča,
- `-s` slúži na spustenie generovania podpisu za pomoci príslušného súkromného kľúča,
- `-v` slúži na spustenie overenia podpisu za pomoci príslušného verejného kľúča,
- `-n` nastavuje parameter n, teda veľkosť sústavy polynómov,
- `-d` nastavuje parameter d, teda stupeň HFE polynómu,
- `-t` nastavuje parameter t, teda rozmer perturbácie,
- `-a` nastavuje parameter a modifikátora „mínus“,
- `-i` nastavuje cestu k priečinku s kľúčmi (kde sa nachádzajú pri podpisovaní/overení, alebo kam sa vygenerované kľúče majú uložiť),
- `-f` nastavuje cestu súboru ktorý podpisujeme (alebo pre ktorý podpis overujeme),
- `-g` nastavuje cestu k priečinku, do ktorého sa uloží podpis (alebo kde nájdeme podpis ktorý overujeme),
- `-p` tento prepínač aktivuje perturbovanú verziu HFE schémy.

Majme teda spustiteľný súbor *HFE_APP.exe*. Príklad príkazu, ktorého zadaním do konzoly zobrazíme pomocný výpis všetkých definovaných prepínačov vidíme vo výpise B.1.

```
HFE_APP -h
```

Výpis č. B.1

Zadaním príkazu z výpisu B.1 sa zobrazí obsah výpisuB.2.

```
options:
  -h, --help              produce help message
  -k, --key_generation    generates keys
  -s, --sign              generates digital signature
  -v, --verify            verifies digital signature
  -n, --modulus_degree arg sets parameter n
  -d, --hfe_degree arg    sets parameter d
  -t, --perturbation arg  sets parameter t
  -a, --modifier arg      sets parameter a
  -i, --key_directory arg path to key directory
  -f, --file arg          path to file that should be signed
  -g, --signature arg     path to file with digital signature
  -p, --perturbation_mode ads perturbation to the scheme
```

Výpis č. B.2

Pokiaľ nenastavíme prepínače `-n`, `-d`, `-t` a `-a`, v aplikácii sú predvolené hodnoty parametrov rovné odporúčaným hodnotám z článku [13], teda $n = 263$, $d = 65$, $t = 6$, $a = 7$. Ak by sme chceli nastaviť parametre napríklad na nami testované hodnoty $n = 263$, $d = 33$, $t = 8$, $a = 7$, spravíme to zadaním prepínačov pre hodnoty, ktoré meníme voči predvoleným hodnotám, teda `-t` s číselnou hodnotou 8 a `-d` s číselnou hodnotou 33.

```
HFE_APP -t 8 -d 33
```

Výpis č. B.3

Vo výpise B.4 sa nachádza príkaz, pomocou ktorého sa generujú verejný a súkromný kľúč pre $HFE\hat{+}-$ schému s parametrami $n = 263$, $d = 33$, $t = 8$, $a = 7$. Generovanie kľúčov spúšťame prepínačom `-k`. Vygenerované kľúče sa uložia do priečinku priečinku *KEYS*. Ak by sme nezadali prepínač `-i` s cestou k súboru *KEYS*, kľúče by sa uložili do aktuálneho priečinku, v ktorom sa nachádza spustiteľný súbor *HFE_APP.exe*.

```
HFE_APP -k -p -d 33 -t 8 -i .\\KEYS
```

Výpis č. B.4

Keď už máme súkromný kľúč, môžeme podpísať dokument. Podpisovanie spúšťame prepínačom `-s`. Pri generovaní podpisu vždy zadávame všetky prepínače, ktoré sme použili pri generovaní kľúčov. Okrem nich ešte musíme použiť prepínač `-f`, pomocou ktorého špecifikujeme cestu kde sa nachádza podpisovaný súbor. Ak nenastavíme cestu k priečinku

do ktorého sa má podpis uložiť, podpis sa uloží do priečinku kde sa nachádza súbor *HFE_APP.exe*. Príklad vidíme vo výpise B.5.

```
HFE_APP -s -d 33 -t 8 -i .\\KEYS -p -f .\\priecinok_s_pdf\subor.pdf
```

Výpis č. B.5

Overovanie podpisu pre dokument spúšťame prepínačom `-v`. Musíme zadať cestu k priečinku s verejným kľúčom, cestu k dokumentu ktorý sme podpísali a ak sme nastavovali priečinok do ktorého sa uložil podpis, musíme zadať aj ten. Príklad vidíme vo výpise B.6.

```
HFE_APP -v -d 33 -t 8 -i .\\KEYS -p -f .\\priecinok_s_pdf\subor.pdf
```

Výpis č. B.6