**ZKRY 2024: sumarizacia**

Poznamka:

V tomto dokumente zosumarizujem, co by ste si z tohto predmetu mali odniest a na co sa mozem pytat na skuske.

**Ch1 – Introduction to Cryptography**

Mali by ste:

* vediet, co je kryptologia, co je kryptografia a co je kryptoanalyza
* poznat princip symetrickej kryptografie
* poznat Kerckhoffov princíp a vediet, preco ma zmysel ho dodrziavat
* poznat princip brute-force utoku na symetricke sifry
* vediet, ze na symetricke sifry mozu existovat aj efektivnejsie utoky ako brute-force (ako napriklad frekvencny utok v pripade substitucnej sifry)
* vediet, ze v sucasnosti sa odporuca pouzivat symetricke kluce s dlzkou aspon 128 bitov
* poznat Moorov zakon a vediet, preco je dolezity pri urcovani dlzky klucov pre symetricke sifry
* poznat principy modularnej aritmetiky tak, ako su popisane v sekcii 1.4 v knihe

**Ch2 – Stream Ciphers**

Mali by ste:

* vediet, ze XOR je to iste ako scitanie modulo 2
* vediet, ze AND je to iste ako nasobenie modulo 2
* vediet, ako sa prudove sifry lisia od blokovych sifier
* poznat princip sifrovania a desifrovania prudovou sifrou
* poznat vlastnosti, ktore musi splnat prud bitov (key stream) generovany v prudovej sifre
* vediet, ake typy generatorov nahodnych cisel pozname (PRNG, TRNG, CSPRNG), v com sa lisia a ktore z nich sa mozu pouzivat na kryptograficke ucely
* poznat definiciu Vernamovej sifry (one-time pad)
* poznat definiciu perfektnej bezpecnosti (unconditional security)
* vediet, ze Vernamova sifra je perfektne bezpecna
* vediet, preco je Vernamova sifra neprakticka
* vediet, ze prakticke prudove sifry pouzivaju na generovanie prudu bitov CSPRNG generatory
* vediet, ze niektore CSPRNG generatory pouzivaju posuvne registre s linearnou spatnou vazbou (LFSR)
* vediet popisat vseobecny LFSR
* vediet, co je stupen LFSR
* vediet, co je perioda LFSR
* vediet pre konkretny LFSR s konkretnym pociatocnym naplnenim urcit jeho periodu
* vediet, ze maximalna perioda LFSR stupna *m* je , a vediet preco
* vediet, ze LFSR mozeme definovat pomocou polynomu
* vediet pre konkretny LFSR urcit jeho polynom
* vediet pre konkretny polynom urcit jemu prisluchajuci LFSR
* vediet, ze LFSR registre s maximalnou periodou su definovane pomocou tzv. primitivnych polynomov (definiciu primitivnych polynomov si nemusite pamatat, ale mali by ste vediet, ze kazdy primitivny polynom je ireducibilny)
* vediet, ze ak je LFSR definovany pomocou ireducibilneho polynomu, potom jeho perioda nezavisi od jeho pociatocneho naplnenia (s vynimkou naplnenia samymi nulami)
* vediet, ze ak je LFSR definovany pomocou reducibilneho polynomu, potom jeho perioda zavisi od jeho pociatocneho naplnenia (aj ked neuvazujeme naplnenie samymi nulami)
* vediet pre konkretny LFSR urcit, ci jeho perioda zavisi od jeho pociatocneho naplnenia (naplnenie samymi nulami neuvazujte)
* vediet, ze jeden samostatny LFSR netvori kryptograficky bezpecny PRNG
* poznat princip utoku popisaneho v podkapitole 2.3.2 a mali by ste tento utok vediet zrealizovat pre LFSR maleho stupna
* vediet, ze prikladom v praxi pouzivanej prudovej sifry, ktora pouziva LFSR je Trivium. Detaily fungovania sifry Trivium poznat nemusite, ale mali by ste vediet, ze pouziva viacero LFSR registrov (a preto na nu utok z 2.3.2 nie je aplikovatelny), a ze poskytuje iba 80-bitovu uroven bezpecnosti a preto nie je vhodna na pouzitie v aplikaciach vyzadujucich vysoku uroven bezpecnosti.
* vediet, ze dalsimi prikladmi v praxi pouzivanych prudovych sifier su Salsa20 a ChaCha20. Detaily fungovania tychto sifier poznat nemusite, ale mali by ste vediet, ze narozdiel od sifry Trivium nepouzivaju LFSR registre, ale ze namiesto toho opakovane aplikuju operacie add, rotate a XOR na 32-bitovych slovach.

**Ch3 – DES**

Mali by ste:

* poznat princip konfuzie a princip difuzie
* poznat princip fungovania sucinovych sifier

**Ch4 – AES**

Mali by ste:

* vediet, ze AES je v sucasnosti najpouzivanejsou symetrickou sifrou
* vediet, ze AES je blokovou sifrou s velkostou bloku 128 bitov
* vediet, ze AES podporuje 3 velkosti klucov: 128, 192 a 256 bitov
* vediet, ze AES ma strukturu substitucno-permutacnej siete
* vediet, ze vsetky vypocty v AES su nad bajtami
* vediet, ze medzistav pocas sifrovania v AES je reprezentovany tabulkou bajtov o velkosti 4\*4
* vediet, ze v zavislosti od velkosti kluca AES obsahuje 10-14 kol
* vediet, ze v kazdom kole sa vykonavaju operacie SubBytes, ShiftRows, MixColumns a AddSubkey (s vynimkou posledneho kola, kde je MixColumns vynechane)
* vediet, ze SubBytes zabezpecuje konfuziu
* vediet, ze ShiftRows a MixColumns zabezpecuju difuziu
* vediet, ze v operaciach SubBytes a MixColumns sa s bajtami pocita ako s prvkami z GF(2^8)
* vediet pocitat s prvkami z GF(2^8) (Hodnotu ireducibilneho polynomu stupna 8, ktory sa v AES pouziva si ale nemusite pamatat. Ak si uloha na skuske bude vyzadovat takyto vypocet, bude v nej tento polynom explicitne uvedeny.)
* vediet, ze sucastou operacie SubBytes je vypocet multiplikativne inverznych prvkov v GF(2^8)
* vediet, ze multiplikativne inverzne prvky v GF(2^8) vieme hladat pomocou rozsireneho

Euklidovho algoritmu pre polynomy

* vediet, ze v operacii ShiftRows sa riadky medzistavu iba cyklicky posuvaju (nemusite vediet o kolko a kam sa posunie ktory riadok)
* vediet, ze v operacii MixColumns sa stlpce medzistavu nasobia maticou rozmeru 4\*4 a ze toto nasobenie sa vykonava nad GF(2^8) (hodnotu matice si nemusite pamatat)
* vediet, ze desifrovanie v AES vyzera velmi podobne ako sifrovanie a ze namiesto SubBytes, ShiftRows a MixColumns sa pouzivaju k nim inverzne operacie

**Ch5 – More About Block Ciphers**

Mali by ste:

* vediet, ako funguje ECB mod, a preco nie je vhodny na sifrovanie viacerych blokov plaintextu pomocou toho isteho kluca
* vediet, ako funguje CBC mod, a ako je v CBC mode odstraneny zakladny nedostatok ECB modu
* vediet, ako funguje CTR mod, a ako je v CTR mode odstraneny zakladny nedostatok ECB modu
* vediet, ze CTR mod funguje ako prudova sifra
* vediet, ze v sucasnosti sa casto preferuju mody sifrovania, ktore okrem dovernosti zabezpecuju aj autentizaciu dat (poznatky, ktore by ste mali vediet o takychto modoch sifrovania su zhrnute nizsie v sumarizacii pre Kapitolu 13)
* vediet, ze IV nemusi byt tajny, ale ze ten isty IV sa nesmie pouzit viackrat

**Ch6 – Introduction to Public-Key Cryptography**

Mali by ste:

* vediet, aku ulohu hra v asymetrickych sifrovacich schemach sukromny kluc a aku hra verejny kluc
* vediet, na co sluzia certifikaty verejnych klucov
* vediet, aka je vyhoda asymetrickych sifrovacich schem oproti symetrickym a aka je ich nevyhoda
* vediet, ze v praxi sa asymetricka sifrovacia schema casto pouzije iba na vymenu tajneho kluca pre symetricku sifrovaciu schemu a ze nasledne sa uz data sifruju pomocou symetrickej schemy
* vediet, ze okrem vymeny kluca sa asymetricka kryptografia casto vyuziva na tvorbu digitalnych podpisov
* vediet, co znamena, ze funkcia je jednosmerna („one-way“)
* vediet, co znamena, ze funkcia je „one-way trapdoor function“
* vediet, ze v asymetrickych sifrovacich schemach musi byt sifrovacia funkcia „one-way trapdoor function“ a ze „trapdoor“ v tomto pripade predstavuje znalost sukromneho kluca
* vediet, ze dnes pouzivane klasicke (t.j. nie postkvantove) asymetricke kryptosystemy su zalozene bud na probleme faktorizacie alebo na probleme diskretneho logaritmu
* vediet, ze velkost klucov v asymetrickych kryptosystemoch sa urcuje na zaklade najefektivnejsich znamych algoritmov na riesenie problemu faktorizacie, problemu diskretneho logaritmu alebo postkvantovych problemov
* ovladat poznatky z teorie cisel uvedene v podkapitole 6.3 v knihe
* vediet vypocitat hodnoty Eulerovej phi funkcie pre lubovolne prirodzene cislo vacsie ako 1
* vediet hladat multiplikativne inverzne prvky v  aj pomocou rozsireneho Euklidovho algoritmu aj pomocou Eulerovej vety

**Ch7 – RSA**

Mali by ste:

* vediet, ze RSA je v sucasnosti popularnou v praxi pouzivanou asymetrickou sifrovacou schemou
* vediet popisat RSA sifrovaciu schemu tak, ako je popisana v podkapitolach 7.2 a 7.3 v knihe (treba vediet popisat algoritmus generovania klucov, algoritmus sifrovania aj algoritmus desifrovania)
* vediet, ze desifrovanie v RSA funguje vdaka Eulerovej vete
* vediet, preco sa v algoritme sifrovania nemoze zvolit za spravu x vacsie cislo ako n-1
* vediet, ze bezpecnost RSA sifrovacej schemy sa opiera o narocnost problemu faktorizacie
* vediet sformulovat problem faktorizacie
* vediet, ze v sucasnosti sa v RSA odporuca pouzivat prvocisla p a q velkosti aspon 1024 bitov (a ze z toho vyplyva, ze n ma mat aspon 2048 bitov)
* vediet, ze prvocisla p a q dostatocnej velkosti pre RSA generujeme tak, ze opakovane nahodne generujeme velke neparne cislo a na toto neparne cislo aplikujeme test prvociselnosti, az kym vystup z testu prvociselnosti nehovori, ze testovane cislo je prvocislo
* vediet, ze v praxi pouzivane testy prvociselnosti maju tu vlastnost, ze mozu testovane cislo prehlasit za prvocislo, aj ked je zlozene
* vediet, ze ak v praxi pouzivane testy prvociselnosti opakujeme pre jedno testovane cislo dostatocne velakrat a vystup z testu zakazdym hovori, ze testovane cislo je prvocislo, tak pravdepodobnost, ze testovane cislo je zlozene bude zanedbatelne nizka (mensia ako )
* vediet, ze takto pre potreby RSA generujeme cisla p a q, ktore su s velkou pravdepodobnostou prvocisla
* vediet, popisat Fermatov test prvociselnosti tak, ako je popisany v ramceku na str.221 v knihe
* vediet, ze existuju zlozene cisla, ktore splnaju podmienku vo Fermatovom teste pre vela cisel „a“. (nemusite si ale pamatat, ze tieto zlozene cisla sa volaju Carmichaelove)
* vediet, ze z tohto dovodu sa namiesto Fermatovho testu pouziva iny test, ktory je ale zalozeny na podobnom principe (nemusite si pamatat, ze tento test sa vola Miller-Rabinov)
* ovladat square-and-multiply algoritmus
* vediet, ze sifrovaci exponent sa v RSA casto voli ako male cislo, co ma za nasledok velmi rychle sifrovanie
* vediet vykonat RSA desifrovanie pomocou cinskej zvyskovej vety
* vediet, ze problemom „schoolbook“ verzie RSA (t.j. verzie prezentovanej v podkapitolach 7.2 a 7.3) je, ze sifrovanie je deterministicke (rovnaky plaintext sa pomocou rovnakeho verejneho kluca vzdy zasifruje rovnako)
* vediet, ze v praxi sa RSA preto pouziva s paddingom, ktory do plaintextu pred sifrovanim vnesie nahodnost (nemusite ale presne vediet ako funguje OAEP padding)
* vediet, co znamena, ze sifrovacia schema je malleable
* vediet, ze schoolbook RSA je malleable, ale ze pouzitim vhodneho paddingu vieme tento nedostatok odstranit
* vediet, ako funguje KEM, na co sa pouziva, a v com sa lisi od vseobecnej asymetrickej sifrovacej schemy
* vediet, co je to KDF a aku rolu hra KDF pri vytvoreni zdielaneho tajneho kluca pomocou KEMu

**Ch8 – Cryptosystems Based on the Discrete Logarithm Problem**

Mali by ste:

* ovladat poznatky o grupach tak, ako su uvedene v *Pomocnom materialy o grupach* dostupnom na webstranke predmetu
* poznat definiciu problemu diskretneho logaritmu v
* vediet, ze kedze ma netrivialne podgrupy, casto sa v kryptosystemoch zalozenych na probleme diskretneho logaritmu namiesto generatora pouziva generator podgrupy s prvociselnou velkostou
* poznat definiciu problemu diskretneho logaritmu v lubovolnej konecnej cyklickej grupe
* poznat priklady grup, na ktorych je problem diskretneho logaritmu tazky
* poznat priklad grupy, na ktorej problem diskretneho logaritmu nie je tazky
* vediet popisat DHKE protokol nad tak, ako je popisany v podkapitole 8.1 v knihe (nezabudnite, ze α musi byt generator )
* vediet, preco v DHKE protokole nad generujeme hodnoty *a,b*  nahodne z a nie z
* vediet, ze pary *(a, A)* a *(b, B)* volame efemerne pary klucov a ze tieto pary sa generuju v kazdom behu DHKE protokolu nanovo
* poznat definiciu DH problemu v
* vediet, ze kedze ma netrivialne podgrupy, casto sa v kryptosystemoch zalozenych na DH probleme namiesto generatora pouziva generator podgrupy s prvociselnou velkostou
* poznat definiciu DH problemu v lubovolnej konecnej cyklickej grupe
* poznat priklady grup, na ktorych je DH problem tazky
* poznat priklad grupy, na ktorej DH problem nie je tazky
* vediet, co je zname o vztahu DH problemu a problemu diskretneho logaritmu
* vediet, ze na zlomenie DHKE protokolu staci vyriesit DH problem
* vediet popisat ElGamalovu sifrovaciu schemu tak, ako je popisana v podkapitole 8.5.2 v knihe (treba vediet popisat algoritmus generovania klucov, algoritmus sifrovania aj algoritmus desifrovania)
* vediet, preco v ElGamalovej sifrovacej scheme generujeme hodnoty *d,i*  nahodne z a nie z
* vediet, ze par *(d, β)* je dlhodoby klucovy par
* vediet, ze hodnoty sa pre kazdy novy plaintext generuju nanovo
* vediet, ze desifrovanie v ElGamalovej sifrovacej scheme si vieme zjednodusit tak, ze nebudeme pocitat ale rovno vypocitame podla vztahu na strane 265 v knihe
* vediet, ze narozdiel od schoolbook RSA vzniknu v ElGamalovej sifrovacej scheme pri opakovanom sifrovani rovnakeho plaintextu pomocou rovnakeho verejneho kluca rozne ciphertexty
* vediet, ze ElGamalova sifrovacia schema je malleable, a ze z tohto dovodu sa v praxi pouziva s paddingom (podobne ako RSA)
* vediet, ze v sucasnosti sa v DHKE nad aj v ElGamalovej sifrovacej scheme odporuca pouzivat prvocislo p velkosti aspon 2048 bitov

**Ch9 – Elliptic Curve Cryptosystems**

Mali by ste:

* vediet definovat elipticku krivku tak, ako je definovana v Definition 9.1.1 na str.279 v knihe (podmienku si nemusite pamatat)
* poznat geometricku interpretaciu scitania bodov na eliptickej krivke (Vztahy na vypocet koordinatov suctu si ale nemusite pamatat. Ak budem na skuske chciet, aby ste scitali dva body na eliptickej krivke, dostanete tieto vztahy k dispozicii.)
* vediet, ze elipticka krivka s operaciou scitania bodov tvori grupu (a vediet, ze za urcitych podmienok je tato grupa cyklicka)
* vediet, ako su v tejto grupe definovane inverzne prvky
* vediet, ze bod v nekonecne hra v tejto grupe rolu identickeho prvku
* vediet, ze ak je elipticka krivka definovana modulo p, tak pocet bodov na tejto eliptickej krivke je blizky cislu p (presne ohranicenia z Hasseho vety si ale nemusite pamatat)
* vediet definovat problem diskretneho logaritmu pre elipticke krivky
* vediet popisat ECDH protokol tak, ako je popisany v podkapitole 9.3 v knihe
* vediet, preco v ECDH protokole generujeme hodnoty *a,b*  nahodne z a nie z
* vediet, ze pary *(a, A)* a *(b, B)* volame efemerne pary klucov a ze tieto pary sa generuju v kazdom behu ECDH protokolu nanovo
* vediet ako funguje double-and-add algoritmus na nasobenie bodu eliptickej krivky skalarom
* vediet, ze niektore algoritmy, ktore vieme pouzit na riesenie problemu diskretneho logaritmu v , nie su aplikovatelne na riesenie problemu diskretneho logaritmu pre elipticke krivky
* vediet, ze z tohto dovodu mozeme v kryptosystemoch zalozenych na probleme diskretneho logaritmu pre elipticke krivky pouzivat ovela mensie parametre ako v kryptosystemoch zalozenych na probleme diskretneho logaritmu v alebo na probleme faktorizacie
* vediet, ze to ma za nasledok, ze kryptosystemy zalozene na probleme diskretneho logaritmu pre elipticke krivky su casto efektivnejsie ako kryptosystemy zalozene na probleme diskretneho logaritmu v alebo na probleme faktorizacie (vynimkou je sifrovanie v RSA a overovanie RSA podpisu, ktore je velmi efektivne vdaka malemu verejnemu exponentu e).
* vediet, ze v sucasnosti sa v kryptosystemoch zalozenych na probleme diskretneho logaritmu pre elipticke krivky odporuca na dosiahnutie 128 bitovej bezpecnosti pouzivat prvocislo p velkosti aspon 256 bitov

**Ch10 – Digital Signatures**

Mali by ste:

* poznat 4 zakladne kryptograficke ciele tak, ako su uvedene na str.303 v knihe
* vediet, ze nepopieratelnost vacsinou dosahujeme pomocou asymetrickej kryptografie (t.j. pomocou digitalnych podpisov)
* vediet popisat princip digitalnych podpisov tak, ako je vysvetleny v podkapitole 10.1.2 v knihe
* vediet, ze okrem nepopieratelnosti digitalne podpisy zabezpecuju aj integritu a autentizaciu sprav, ale ze nezabezpecuju dovernost sprav
* vediet popisat RSA podpisovu schemu tak, ako je popisana v podkapitole 10.2.1 knihe (treba vediet popisat algoritmus generovania klucov, algoritmus podpisovania aj algoritmus overovania podpisu)
* vediet, ze overenie podpisu v RSA funguje vdaka Eulerovej vete
* vediet, preco sa v algoritme podpisovania nemoze zvolit za spravu x vacsie cislo ako n-1
* vediet, ze bezpecnost RSA podpisovej schemy sa opiera o narocnost problemu faktorizacie
* vediet, ze v sucasnosti sa v RSA podpisovej scheme odporuca pouzivat prvocisla p a q velkosti aspon 1024 bitov (a ze z toho vyplyva, ze n ma mat aspon 2048 bitov)
* vediet, ze overovaci exponent sa v RSA podpisovej scheme casto voli ako male cislo, co ma za nasledok velmi rychle overovanie podpisu
* vediet, ako je na „schoolbook“ verziu RSA podpisovej schemy (t.j. verzie prezentovanej v podkapitole 10.1.2) mozne vykonat „existential forgery attack“
* vediet, ze tomuto utoku mozeme zabranit tak, ze budeme vyzadovat, aby sprava x mala specificky format (nemusite ale poznat detaily EMSA-PSS kodovania)
* vediet, ze okrem RSA su v praxi popularne podpisove schemy DSA (zalozena na probleme diskretneho logaritmu v ) a ECDSA (zalozena na probleme diskretneho logaritmu pre elipticke krivky)
* vediet, ze v porovnani s RSA ma ECDSA vyhodu v tom, ze produkuje ovela kratsie podpisy

**Ch11 – Hash Functions**

Mali by ste:

* vediet, ze z dovodu obmedzenej velkosti vstupu do podpisovacieho algoritmu sa vacsinou nepodpisuje povodna sprava ale jej hash
* poznat vlastnosti kryptografickych hashovacich funkcii tak, ako su uvedene v ramceku na spodku str.345 v knihe
* vediet, ako narocnost hladania 2. vzoru hrubou silou zavisi od dlzky vystupu z hashovacej funkcie
* vediet, ze hladanie kolizie je vypoctovo ovela menej narocne ako hladanie 2. vzoru
* vediet, ze ocakavany pocet sprav, ktore treba zahashovat, kym najdeme koliziu je priblizne , kde n je dlzka vystupu z hashovacej funkcie (presnejsi vztah (11.1), ktory je v knihe uvedeny na str. 344 si nemusite pamatat)
* vediet, ze tento jav sa vola narodeninovy paradox, a vediet preco sa takto nazyva
* vediet, ze kvoli narodeninovemu paradoxu potrebujeme na dosiahnutie 128 bitovej bezpecnosti hashovacie funkcie s vystupom dlzky 256 bitov
* vediet, ze v praxi popularne hashovacie funkcie su SHA-2 a SHA-3 (nemusite ale vediet podrobnosti o tom, ako tieto hashovacie funkcie funguju)

**Ch13 – MACs**

Mali by ste:

* vediet popisat princip MACov tak, ako je vysvetleny v podkapitole 13.1 v knihe
* vediet, ze MACy zabezpecuju integritu a autentizaciu sprav, ale ze nezabezpecuju ani dovernost ani nepopieratelnost sprav
* vediet, ze MACy su ovela rychlejsie ako digitalne podpisy
* poznat vlastnosti MACov tak, ako su uvedene v ramceku na str.467 v knihe
* vediet, ze narozdiel od hashovacich funkcii potrebuje utocnik pri hladani kolizii v MACoch asistenciu niekoho so znalostou tajneho kluca a ze preto stacia na dosiahnutie 128 bitovej bezpecnosti MACy s dlzkou vystupu iba 128 bitov
* vediet, ze velmi popularnym MACom v sucasnosti je HMAC (nemusite ale vediet podrobnosti o tom, ako HMAC funguje)
* vediet, ze v sucasnosti su casto preferovane mody sifrovania, ktore kombinuju funkcionalitu symetrickej sifry s funkcionalitou MACu a tak poskytuju sucasne dovernost aj autenticitu sprav
* vediet, ze taketo mody symetrickeho sifrovania nazyvame *autentizovane sifrovanie*
* vediet, ze autentizovane sifrovanieposkytuju mody GCM a CCM (nemusite ale vediet podrobnosti o tom, ako tieto mody funguju)

**Ch12 – PQC**

Mali by ste:

* vediet, ze na vykonnom kvantovom pocitaci by bolo mozne pomocou Groverovho algoritmu vykonat vyhladavanie v neusporiadanej databaze obsahujucej *N* prvkov v priblizne krokoch
* vediet, ake to ma dosledky pre bezpecne pouzivanie dnesnych symetrickych sifier v pripade, ze by ludstvo malo k dispozicii dostatocne vykonny kvantovy pocitac
* vediet, ze na vykonnom kvantovom pocitaci by bolo mozne pomocou Shorovych algoritmov efektivne riesit problemy faktorizacie a diskretneho logaritmu
* vediet, ake to ma dosledky pre bezpecne pouzivanie asymetrickych sifier zalozenych na problemoch faktorizacie a diskretneho logaritmu v pripade, ze by ludstvo malo k dispozicii dostatocne vykonny kvantovy pocitac
* vediet, ze v sucasnosti ludstvo nema k dispozicii kvantovy pocitac dostatocne vykonny na lamanie dnes pouzivanych asymetrickych sifier
* vediet, ze prevlada nazor, ze vyvoj takto vykonneho kvantoveho pocitaca bude trvat este minimalne 10 rokov
* vediet, preco je dolezite zacat sa pripravovat na prichod vykonnych kvantovych pocitacov uz dnes
* vediet, aky je rozdiel medzi kvantovou kryptografiou a postkvantovou kryptografiou
* vediet, ze uz existuju standardy pre postkvantovy KEM a pre postkvantove digitalne podpisy (z nazvov tychto standardov by ste si mali pamatat ML-KEM (aka Kyber), ostatne nazvy standardov si pamatat nemusite) a ze NIST je v procese tvorby dalsich postkvantovych kryptografickych standardov
* vediet, ze 3 najprominentnejsie rodiny postkvantovych kryptosystemov su kryptosystemy zalozene na mriezkach, kryptosystemy zalozene na kodoch a kryptosystemy zalozene na hashovacich funkciach
* vediet definovat mriezku s hodnostou (rankom) *n* v
* vediet, co je dimenzia mriezky
* vediet, ako v kryptografii zalozenej na mriezkach definujeme male prvky v
* vediet definovat LWE problem (na prednaske sme uviedli trochu inu verziu ako v knihe, staci, ak budete poznat lubovolnu z tychto dvoch verzii)
* vediet, ako LWE problem suvisi s mriezkami
* vediet, ze sa predpoklada, ze pre vhodne zvolene parametre je LWE problem tazky pre klasicke aj pre kvantove pocitace
* vediet, ze v pripade, ze by v LWE probleme nevystupoval chybovy vektor *e*, by sa LWE problem dal efektivne riesit pomocou Gaussovej eliminacie
* vediet definovat okruh tak, ako je definovany v Definition 12.2.3 na str. 399 v knihe, a vediet robit vypocty v tomto okruhu
* vediet definovat Ring-LWE problem tak, ako je definovany v Definition 12.2.4 na str.400 v knihe (Pozor, v definicii je vo vyraze nespravne umiestneny subscript . Spravne ma byt umiestneny pri *Z*tak, ako v definicii 12.2.3.)
* vediet, preco je Ring-LWE problem specialnym pripadom LWE problemu
* vediet, ze okrem LWE a Ring-LWE problemoch, este existuje aj Module-LWE problem (Module-LWE problem ale nemusite vediet definovat)
* vediet, ze na kazdom z tychto 3 problemov vieme zalozit asymetricku sifrovaciu schemu s podobnou konstrukciou ako je konstrukcia ElGamalovej sifrovacej schemy (detaily schem prezentovanych v podkapitolach 12.2.2 a 12.2.4 si ale nemusite pamatat)
* vediet, ze schemy zalozene na Ring-LWE alebo na Module-LWE probleme maju typicky mensie parametre ako schemy zalozene na LWE probleme.
* vediet, ze NIST standardizoval postkvantovy KEM zalozeny na Module-LWE probleme a ze tento KEM sa vola ML-KEM (je tiez znamy ako Kyber)
* vediet, akym praktickym problemom sa zaobera teoria kodovania
* vediet, ze binarny linearny kod dlzky n je linearny podpriestor vektoroveho priestoru
* vediet, co je dimenzia binarneho linearneho kodu
* vediet, co je generujuca matica binarneho linearneho kodu, a ako vieme pomocou generujucej matice vygenerovat vsetky kodove slova
* vediet, kedy je generujuca matica v systematickom tvare
* vediet, co je kontrolna matica binarneho linearneho kodu, a ako vieme pomocou kontrolnej matice urcit, ci nejaky vektor je kodovym slovom
* vediet z generujucej matice vypocitat kontrolnu maticu
* vediet, co je syndrom vektora, ktory vznikol z kodoveho slova pridanim chyboveho vektora *e*, a vediet, ze hodnota syndromu nezavisi od povodneho kodoveho slova ale iba od chyboveho vektora *e*
* vediet, ako funguje kodovanie pomocou binarneho linearneho kodu (obrazok 12.7 na str. 411 v knihe)
* vediet, ako funguje detekcia chyby pomocou binarneho linearneho kodu (obrazok 12.7 na str. 411 v knihe)
* vediet definovat minimalnu vzdialenost binarneho linearneho kodu
* vediet, ze pre binarne linearne kody mozeme minimalnu vzdialenost vypocitat tak, ako je uvedene na slajde 34 v prezentacii o PQC
* vediet, ako suvisi minimalna vzdialenost binarneho linearneho kodu s poctom chyb, ktore kod dokaze opravit
* vediet pre binarny linearny kod kratkej dlzky vykonat dekodovanie tak, ako je uvedene v Example 12.7 na stranach 415-417 v knihe (to znamena, ze z kodoveho slova ovplyvneneho takym poctom chyb, ktory kod dokaze opravit, treba urcit povodnu spravu)
* vediet, preco je metoda dekodovania popisana v Example 12.7 neefektivna pre binarne linearne kody schopne opravovat vacsi pocet chyb
* vediet definovat MDD problem tak, ako je definovany na slajde 37 v prezentacii o PQC
* vediet, ze sa predpoklada, ze pre vhodne zvolene parametre je MDD problem tazky pre klasicke aj pre kvantove pocitace
* vediet, ze v pripade, ze by v MDD probleme nevystupoval chybovy vektor *e*, by sa MDD problem dal efektivne riesit pomocou Gaussovej eliminacie
* vediet, ze v MDD probleme je dolezite, aby kod C bol nahodne generovany
* vediet, ze existuju specialne konstrukcie binarnych linearnych kodov, ktore umoznuju efektivne opravit aj vacsi pocet chyb (viac o takychto konstrukciach sa mozete dozvediet na predmete Teoria kodovania)
* si byt vedomi podobnosti medzi MDD problemom a LWE problemom a vediet, v com sa tieto problemy lisia
* si byt vedomi, ze kvoli podobnostiam medzi MDD problemom a LWE problemom existuju mnohe paralely medzi code-based kryptosystemami a lattice-based kryptosystemami