Specijalni kurs

Kriptografija

1. Navesti razlike između simetričnih i asimetričnih kriptosistema

Kriptosistem je par koji čine algoritam za kriptovanje i algoritam za dekriptovanje. Algoritmi su najčešće svima poznati i uvek zavise od parametra koji se zove **ključ** i koji se potpuno ili delimično čuva u tajnosti. U komunikaciji obično imamo sledeći scenario: Alisa šalje poruku Bobanu, a Cica krade šifrat i pokušava da ga dekriptuje ne znajući ključ, što se naziva **kriptoanaliza**. Kriptosistemi mogu biti:

- **simetrični** - Alisa i Boban koriste isti ključ za (de)kriptovanje. Unapred dogovore ključ, čuvaju ga u tajnosti i periodično ga menjaju. Ovi kriptosistemi su nepraktični kada imamo veliki broj korisnika gde svako komunicira sa svakim. Nedostaci su i to što je potreban dodatni kanal komunikacije za razmenu ključa, kao i to što su ovi sistemi lakši za kriptoanalizu.
- **asimetrični** (**kriptosistemi sa javnim ključem**) - Alisa i Boban prave sopstvene ključeve, pri čemu ključ za kriptovanje objavljuju, a ključ za dekriptovanje čuvaju u tajnosti. Nedostatak ovih kriptosistema je sporo izvršavanje pa se obično koriste za prenos kratke poruke kada je potrebna visoka bezbednost.

Najbolje rezultate zapravo daje kombinovanje ova dva pristupa tako što se za slanje poruke koristi simetrični kriptosistem, a njegov ključ se razmenjuje asimetričnim kriptosistemom.

2. Cezarova i afina šifra i njihova kriptoanaliza

Cezarova šifra: Slova A-Z kodiramo sa $\{0, \ldots, 25\} = \mathcal{Z}_{26}$ i koristimo protočnu šifru $f(P) = P + b \; mod \; 26$, gde je $P \in \mathcal{Z}_{26}$ kodirani simbol, a b tajni ključ. Algoritam za dekodiranje bi bio $f^{-1}(C) = C - b \; mod \; 26$. Na primer, ako je tajni ključ $b = 16$ i Alisa šalje poruku "PAYMENOW" dobijamo šifrat:
"PAYMENOW" $\rightarrow$ 15 0 24 12 4 13 14 22 $\xrightarrow{f}$ 5 16 14 2 20 3 4 12 $\rightarrow$ "FQOCUDEM"

Afina šifra predstavlja uopštenje Cezarove šifre:

$$f(P) = aP + b \; mod \; 26, \; f^{-1}(C) = a'C + b' \; mod \; 26,$$

gde je $a' = a^{-1}$ inverz od a u $\mathcal{Z}_{26}^*$ i $b' = -a^{-1}b$.

Kriptoanaliza: poznato je da je najfrekventnije slovo u tekstu na engleskom jeziku slovo 'E'. Cica pronalazi najfrekventnije slovo u šifratu (npr. slovo 'K') i pretpostavlja da je f('E') = 'K', odnosno f^{-1}('K') = 'E'. Slično uspostavlja vezu između drugog najfrekventnijeg slova u tekstu na engleskom jeziku i drugog najfrekventnijeg slova u šifratu (npr. f('T') = 'D'). Cica

rešavanjem sistema pronalazi ključ. Ako sistem nema rešenje ili ono nije jedinstveno, umesto drugog najfrekventnijeg slova može da koristi treće, četvrto i slično.

3. Jednokratna šifra (One Time Pad)

Jednokratna šifra (One Time Pad) je najjednostavnija protočna šifra. Poruka M kodira se binarno. Ključ K, koji je takođe zapisan binarno, mora biti iste dužine kao M. Kriptovanje se vrši bit po bit, sabiranjem bita iz M i bita iz K po modulu 2. Dekriptovanje se radi identično sa istim ključem. Ključ sme da se koristi samo jednom, a ponovna upotreba istog ključa dovodi do curenja podataka.

4. Matrično kriptovanje digrafa

Umesto sa pojedinačnim slovima, bolje je raditi sa većim blokovima slova. **Digrafovi** par slova kodiraju brojem iz skupa $\{0,\ \ldots,\ 26^2 - 1 = 675\}$. Na primer, digraf "NO" se kodira sa 26 · 'N' + 'O' = 26 · 13 + 14 = 352, a zatim se kriptuje sa 159 · 352 + 580 = 440 mod 676 što je ekvivalent "QY", pri čemu su 159 i 580 neki ključevi.

Neka je $A \in M_2(\mathcal{Z}_n)$ invertibilna matrica, tj. takva da je $\det A$ invertibilno u $\mathcal{Z}_n$. Algoritam za kriptovanje i dekriptovanje digrafa onda može biti:

$$\begin{pmatrix} x \\ y \end{pmatrix} \xrightarrow{f} A \begin{pmatrix} x \\ y \end{pmatrix} \ i \ \begin{pmatrix} x \\ y \end{pmatrix} \xrightarrow{f^{-1}} A^{-1} \begin{pmatrix} x \\ y \end{pmatrix}$$

Na primer, neka je $n = 26$ i $A = \begin{pmatrix} 2 & 3 \\ 7 & 8 \end{pmatrix}$ i neka Alisa šalje poruku "NO|AN|SW|ER":
$\begin{pmatrix} 2 & 3 \\ 7 & 8 \end{pmatrix} \begin{pmatrix} 13 & 0 & 18 & 4 \\ 14 & 13 & 22 & 17 \end{pmatrix} = \begin{pmatrix} 16 & 13 & 24 & 7 \\ 21 & 0 & 16 & 8 \end{pmatrix}$ što je "QVNAYGHI". Sa druge strane, ako Boban dobije poruku "FW|MD|IQ" on će je pročitati pomoću $A^{-1} = \begin{pmatrix} 14 & 11 \\ 17 & 10 \end{pmatrix}$:
$\begin{pmatrix} 14 & 11 \\ 17 & 10 \end{pmatrix} \begin{pmatrix} 5 & 12 & 8 \\ 22 & 3 & 16 \end{pmatrix} = \begin{pmatrix} 0 & 19 & 2 \\ 19 & 0 & 10 \end{pmatrix}$ što je "ATTACK".

5. Jednosmerne funkcije. Navesti primer jednosmerne funkcije

Kažemo da je funkcija $f : X \to Y$ **jednosmerna** ako se za poznato $x \in X$ lako (brzo) može izračunati $f(x)$, ali je za poznato $y \in Y$ teško (neizvodljivo u realnom vremenu) izračunati $f^{-1}(y)$. Dovoljno je da je f injekcija, a najčešće je bijekcija. Primer korišćenja jednosmerne funkcije f: Neki internet servis za svakog korisnika čuva par $(N,\ f(P))$ gde je N korisničko ime, a P šifra. Kada se prilikom prijavljivanja unesu N i P' lako se računa $f(P')$ i proverava da li se poklapa sa $f(P)$. U slučaju da Cica ukrade podatke $(N,\ f(P))$, ona ne može da izračuna P. Najčešće je vremenska složenost funkcija f i f^{-1} redom $O(n^k)$ i $O(l^n)$. Kriptosistemi sa javnim ključem zasnivaju se na jednosmernim funkcijama.

6. Difi-Helmanov algoritam za usaglašavanje ključeva

Ako je p prost broj, tada je $(\mathcal{Z}_p,\ +_p,\ \cdot_p)$ polje, gde je $\mathcal{Z}_p$ = $\mathcal{Z}/(p\mathcal{Z})$ = $\{0,\ 1,\ \ldots,\ p-1\}$. Multiplikativna grupa $\mathcal{Z}_p^*$ = $(\mathcal{Z}_p \backslash \{0\},\ \cdot_p)$ je ciklična, tj. postoji generator (primitivni koren)

$g \in \mathcal{Z}_p \backslash \{0\}$ tako da se svi elementi $\mathcal{Z}_p \backslash \{0\}$ mogu videti kao stepeni g. **Difi-Helmanova razmena ključa** zasniva se na:
♣ ako znamo $g \in \mathcal{Z}_p^*$ i $n \in N$ lako je odrediti g^n
♠ ako znamo g i g^n teško je odrediti n

Algoritam:

- Alisa i Boban biraju prost broj p (približno 200-cifren) i generator $g \in \mathcal{Z}_p^*$ i objavljuju p i g
- Alisa bira svoj tajni ključ $a_A \in N$, računa i objavljuje javni ključ $g^{a_A} \; mod \, p$
- Boban bira svoj tajni ključ $a_B \in N$, računa i objavljuje javni ključ $g^{a_B} \; mod \, p$
- Alisa i Boban mogu izračunati usaglašeni ključ $K = (g^{a_A})^{a_B} \; mod \, p$ = $(g^{a_B})^{a_A} \; mod \, p$

Cica zna samo $p, \; g, \; g^{a_A}$ i g^{a_B} i ne može lako (brzo) da odredi K.

7. Algoritam za stepenovanje ponovljenim kvadriranjem

Brzi algoritam za ♣ zasniva se na **ponovljenom kvadriranju** i sadrži sledeće korake (sve operacije su po modulu p):

1. Redukujemo stepen na $n < p - 1$ zbog Male Fermaove teoreme: $g^{p-1} = 1$ u $\mathcal{Z}_p$
2. Zapisujemo n binarno:

$$n = \overline{n_r n_{r-1} \dots n_1 n_0} = \sum_{i=0}^{r} n_i \cdot 2^i, \; n_i \in \{0, \; 1\}$$

3. Računamo

$$1, \; g, \; g^2, \; (g^2)^2 = g^{2^2}, \; \dots, \; (g^{2^{r-1}})^2 = g^{2^r}$$

4. g^n je proizvod onih stepena g za koje je $n_i = 1$:

$$g^n = g^{\sum_{i=0}^{r} n_i \cdot 2^i} = \prod_{i=0}^{r} g^{n_i \cdot 2^i} = \prod_{0 \leq i \leq r, \; n_i = 1} g^{2^i}$$

Za množenje k-bitnog broja a i l-bitnog broja b treba kl operacija, gde je $k \sim \log a$ i $l \sim \log b$, a isto važi i za celobrojno deljenje i uzimanje ostatka po modulu. Vremenska složenost algoritma je onda:

1. $O(\log^2 p)$ jer je $\log p$ broj bitova broja p
2. $O(r \log n) = O(\log^2 p)$ jer r puta ponavljamo deljenje sa 2 i ostatak po modulu 2, a $r \sim \log n \leq \log p$
3. r kvadriranja, a za svako kvadriranje treba po $O(\log^2 (g^{2^i})) = O(\log^2 p)$ operacija
4. najviše r množenja koja zahtevaju po najviše $O(\log^2 p)$ operacija

Dakle, ukupno $O(r \log^2 p) = O(\log^3 p)$. Za množenje $g^n = g \dots g$ bi trebalo $O(n \log^2 p)$ operacija. Algoritam radi i za modul m koji nije prost, ali može da radi sporije jer se umesto Male Fermaove koristi Ojlerova teorema.

8. Definisati diskretni logaritam. Navesti 3 kriptosistema koji se zasnivaju na problemu diskretnog logaritma

Neka je G grupa (npr. F_q^*) i neka su $a,\ g \in G$. Najmanji prirodan broj n, ako postoji, takav da je $a = g^n$ zovemo **diskretni logaritam** od a u osnovi g i označavamo sa $\log_g a$. Ne postoji formula za izračunavanje diskretnog logaritma, već redom računamo stepene i čekamo da se pojavi tražena vrednost. U najgorem slučaju g je generator i tada moramo proći sve stepene što zahteva $O(q)$ testiranja, a vremenska složenost stepenovanja je $O(\log^4 q)$. Kriptosistemi koji se zasnivaju na problemu diskretnog logaritma su Difi-Helman, Mesi-Omura i ElGamal.

9. Algoritam Geljfond-Šenksa (Baby-step-giant-step algoritam)

Algoritam Geljfond-Šenksa smanjuje vremensku i prostornu složenost pretraživanja diskretnog logaritma sa $O(q)$ na $O(\sqrt{q}\log^2 q)$. Dakle, cilj je izračunati $n = \log_g a$ u F_q^*, gde su a i g poznati. Prvo zapišemo n kao $mi + j$, gde je $m = [\sqrt{q}]$ i $0 \leq i \leq m$, $0 \leq j \leq m-1$. Tada važi $\log_g a = n \Leftrightarrow g^n = a \Leftrightarrow g^{mi+j} = a \Leftrightarrow g^j = a(g^{-m})^i$. Računamo parove $(j,\ g^j)$ i $(i,\ a(g^{-m})^i)$ za sve $i,\ j$ i čuvamo ih sortirane po drugoj koordinati. Kada nađemo i i j za koje se poklapa druga koordinata lako dolazimo do n.

10. Polig-Helmanov algoritam

Neka su $q,\ B \in N$. Za broj $q = p_1^{\alpha_1} \dots p_k^{\alpha_k}$ kažemo da je B**-gladak** ako za sve $1 \leq i \leq k$ važi $p_i^{\alpha_i} \leq B$. **Polig-Helmanov metod** omogućava brzo izračunavanje diskretnog logaritma u F_q^*, sa pretpostavkama da je $B \ll n$ (npr. $B \sim \log n$) i da je $q-1$ B-gladak. Dakle, cilj je izračunati $n = \log_g a$ u F_q^*, gde su a i g poznati. Zbog Male Fermaove teoreme n tražimo kao ostatak po modulu $q-1$. Ako je $q-1 = p_1^{\alpha_1} \dots p_k^{\alpha_k}$, po Kineskoj teoremi o ostacima dovoljno je odrediti n kao ostatak po modulu $p_i^{\alpha_i}$ za svako i. Računanje n po modulu p^α:

- Računamo $\zeta_p = g^{\frac{q-1}{p}}$, a zatim i ζ_p^0, ζ_p^1, ..., ζ_p^{p-1} i sve parove $(j,\ \zeta_p^j)$ čuvamo u nekoj tabeli. Vrednosti ζ_p^j zovemo p-ti koren iz 1, jer su rešenja jednačine $x^p \equiv 1\ (mod\ q-1)$.
- Zapišemo $n \equiv n_0 + n_1 p + \dots + n_{\alpha-1}p^{\alpha-1}\ (mod\ p^\alpha)$. Potrebno je odrediti vrednosti n_i.
- Računamo $a^{\frac{q-1}{p}}$, međutim važi $a^{\frac{q-1}{p}} = g^{\frac{n(q-1)}{p}} = \zeta_p^n = \zeta_p^{n_0}$. U tabeli imamo sve vrednosti ζ_p^j pa lako pronalazimo vrednost n_0.
- Računamo $(\frac{a}{g^{n_0}})^{\frac{q-1}{p^2}} = g^{\frac{(n-n_0)(q-1)}{p^2}} = \zeta_p^{\frac{n-n_0}{p}} = \zeta_p^{n_1}$ pa iz tabele dobijamo n_1. Slično, $(\frac{a}{g^{n_0+n_1 p}})^{\frac{q-1}{p^3}}$ daje n_2, $(\frac{a}{g^{n_0+n_1 p+n_2 p^2}})^{\frac{q-1}{p^4}}$ daje n_3 i tako dalje.

Ponovo nemamo računanje diskretnog logaritma, već traženje vrednosti u tabeli. Dodatno, sada i čuvamo tabelu, odnosno trošimo memoriju, međutim dužina tabele je manja od B. Za svako p imamo α puta ponavljanje računa i na kraju Kinesku teoremu. Vremenska složenost je polinom od B, što je prihvatljivo za malo B. Dakle, ako Alisa i Boban izaberu javni ključ q tako da je $q-1$ B-gladak, Cica može da dekriptuje poruku za dovoljno malo B. Alisa i Boban mogu proveriti B-glatkost za neko malo B, čime pokrivaju i sve vrednosti manje od B, ali

uvek postoji opasnost od $(B+1)$-glatkosti gde je vremenska složenost praktično ista kao za B.

11. Mesi-Omura kriptosistem

Mesi-Omura kriptosistem koristi se za razmenu ključeva ili poruka. Ako je poruka duža, deli se na blokove. Algoritam:

- Fiksira se konačno polje F_q i to je svima poznato, odnosno q je javni ključ.
- I Alisa i Boban biraju svoje tajne ključeve e_A i e_B tako da važi $NZD(e_A,\ q-1)$ = $NZD(e_B,\ q-1)$ = 1.
- Svako od njih računa svoj tajni ključ $d_i \equiv e_i^{-1}\ (mod\ q-1)$, $i \in \{A,\ B\}$.
- Ako je M blok poruke kodiran elementom polja F_q koju treba poslati Alisa, ona računa M^{e_A} i to šalje Bobanu.
- Boban ne može da pročita M^{e_A}, ali može da izračuna $(M^{e_A})^{e_B}$ = $M^{e_Ae_B}$ i to šalje nazad Alisi.
- Alisa računa $(M^{e_Ae_B})^{d_A} = M^{e_B}$ i šalje opet Bobanu. Ovde se koristi $e_Ad_A \equiv 1\ (mod\ q-1)$ što povlači $M^{e_Ad_A} = M$ u F_q.
- Boban računa $(M^{e_B})^{d_B} = M$ i dolazi do početne poruke.

12. Alisa šalje Bobanu poruku pomoću Mesi-Omura kriptosistema i pretpostavljamo da je Cica videla celokupnu komunikaciju. Objasniti zašto Cica ipak ne može da dekriptuje poruku

Ako Cica presretne komunikaciju najviše što može da zna je M^{e_A}, M^{e_B}, $M^{e_Ae_B}$ i javni ključ q. Da bi došla do $M = (M^{e_A})^{d_A}$ mora da izračuna $e_A = \log_{M^{e_B}}(M^{e_Ae_B})$ i $d_A \equiv e_A^{-1}\ (mod\ q-1)$, što ne može lako (brzo) da uradi. Kao dodatna zaštita ključevi e_A, e_B, d_A i d_B mogu se menjati kod svakog bloka. Ponekad se kaže da Mesi-Omura nije ni simetričan ni asimetričan kriptosistem jer se smatra da ima samo javni ključ q, a ostalo su parametri koji se jednokratno generišu.

13. ElGamalov kriptosistem

U **ElGamalovom kriptosistemu** javni ključ q je stepen nekog prostor broja i $g \in F_q^*$ je generator. Algoritam:

- Boban bira svoj tajni ključ e_B i pomoću njega pravi javni ključ g^{e_B} koji šalje Alisi, tj. objavljuje kao kod Difi-Helmana. Ovaj ključ se šalje samo jednom na početku.
- Neka je $M \in F_q$ kodirani blok koji Alisa želi da pošalje. Ona generiše slučajan prirodan broj $k < q$ koji će koristiti samo jednom za blok M. Za naredni blok bira novo k. Alisa šalje Bobanu par informacija g^k i Mg^{e_Bk} = $M(g^{e_B})^k$.
- Boban računa $g^{e_Bk} = (g^k)^{e_B}$, a zatim i njegov inverz i dobija $M = Mg^{e_Bk}(g^{e_Bk})^{-1}$.

Da bi pročitala poruku, Cica mora da reši problem diskretnog logaritma, što ne može lako (brzo) da uradi.

14. Kako se generiše slučajan veliki prost broj

Znamo kako radi generator pseudoslučajnih prirodnih brojeva, ali nasumično izabran broj verovatno nije prost, a ne postoji ni formula za n-ti prost broj p_n. Prvo biramo neparan (veliki) pseudoslučajan broj n. Zatim prost broj tražimo u nizu n, $n+2$, $n+4$, ... Očekujemo da prethodni korak ne traje dugo, jer je razmak između uzastopnih prostih brojeva reda: $p_{m+1} - p_m = (m+1)\log(m+1) - m\log m \sim \log m \sim \log n$. Potreban nam je efikasan način da proverimo da li je neki broj prost. Elementarno rešeto je sporo - ima vremensku složenost $O(\sqrt{n})$.

15. Testovi primalnosti. Šta su ulazni i izlazni podaci kod testa primalnosti

Testovi primalnosti su napravljeni tako da ako broj n padne na testu onda je on složen. Ako broj n prođe test on može, ali ne mora biti prost. Verovatnoća da broj n koji je prošao test bude prost je:

$$v = \frac{card\{n \in [1,\ N] \mid n\ je\ prost\}}{card\{n \in [1,\ N] \mid n\ je\ prošao\ test\}}$$

Test primalnosti je bolji ako je v veće. Obično zavisi od nekih parametara. Ponavljanje testa za razne (nezavisne) parametre povećava verovatnoću da je broj koji je preživeo sva testiranja zaista prost. Test primalnosti mora da radi brzo. Dakle, ulaz za test predstavlja broj n, a izlaz broj v koji predstavlja verovatnoću da je n prost.

16. Definisati pseudoproste i Karmajklove brojeve. Šta je glavni nedostatak Karmajklovog testa primalnosti

Mala Fermaova teorema kaže da za prost broj n i a za koje je $NZD(a,\ n) = 1$ važi $a^{n-1} \equiv 1\ (mod\ n)$ $(\star)$. Međutim, nije nemoguće da MFT važi i ako n nije prost broj. Ako za prirodan broj a i složen broj n tako da $NZD(a,\ n) = 1$ važi MFT, kažemo da je n **pseudoprost broj** u bazi a. **Ojlerova teorema** kaže da važi $a^{\phi(n)} \equiv 1\ (mod\ n)$, ali $\phi(n)$ obično ne znamo. Ako je n pseudoprost u bazi a, onda mora da važi i $a^{n-\phi(n)-1} \equiv 1\ (mod\ n)$. Ovde možemo tražiti sve baze a za koje je n pseudoprost broj jer je vrednost $n - \phi(n) - 1$ obično mnogo manja od $n-1$. Teorema:

1. Ako je n pseudoprost i u bazi a i u bazi b onda je pseudoprost i u bazi ab
   △:

   $(ab)^{n-1} = a^{n-1}b^{n-1} \equiv 1 \cdot 1\ (mod\ n)$ ■
2. Ako je n pseudoprost u bazi a, a nije pseudoprost u bazi b, onda nije pseudoprost ni u bazi ab
   △:

   $(ab)^{n-1} = a^{n-1}b^{n-1} \equiv b^{n-1} \not\equiv 1\ (mod\ n)$ ■
3. Ako n nije pseudoprost u bazi a, onda nije pseudoprost ni u bazi b, bar za pola b-ova iz Z_n^* = $\{b \in Z_n \mid NZD(b,\ n) = 1\}$
   △:

Za svaku bazu c u kojoj je n pseudoprost postoji baza $b = ca$ u kojoj n nije pseudoprost na osnovu 2. ■

Karmajklov broj n je složen broj koji je pseudoprost u svakoj bazi $a \in Z_n^*$. Po prethodnoj teoremi verovatnoća da broj n koji nije ni prost ni Karmajklov prođe test $(\star)$ u jednom testiranju sa slučajno izabranom bazom je najviše $\frac{1}{2}$, a u k testiranja sa slučajno i nezavisno izabranim bazama je najviše $\frac{1}{2^k}$. Test je vrlo efikasan, ali ne odvaja proste od Karmajklovih brojeva. Svaki Karmajklov broj je oblika $p_1 p_2 \dots p_k$, gde je $k \geq 3$ i p_i-ovi su međusobno različiti prosti brojevi. Beskvadratan broj n je Karmajklov akko za svaki prost broj p_i važi $p_i | n \ \Rightarrow \ p_i - 1 | n - 1$. Dakle, Karmajklovi brojevi zaista postoje i moramo da imamo test koji ih odvaja od prostih, što Karmajklov test ne može da uradi i to predstavlja njegov glavni nedostatak.

17. Miler-Rabinov test primalnosti

Posledica MFT: Ako je p neparan prost broj i $a \in Z$ tako da $NZD(a,\ p) = 1$, onda je $a^{\frac{p-1}{2}} \equiv \pm 1 \ (mod\, p)$. Broj na desnoj strani zovemo **Ležandrov simbol** i označavamo sa $(\frac{a}{p})$.
△:
$a^{p-1} \equiv 1 \ (mod\, p) \ \Rightarrow \ p | a^{p-1} - 1$ = $(a^{\frac{p-1}{2}} - 1)(a^{\frac{p-1}{2}} + 1)$, pa p deli bar jednu zagradu jer je prost. ■

Ako za prirodan broj a i neparan složen broj p tako da $NZD(a,\ p) = 1$ važi MFT kažemo da je p **Ojlerov pseudoprost broj** u bazi a. Ako je p Ojlerov pseudoprost u bazi a, onda je on i pseudoprost u bazi a.
△:
$a^{p-1} = (a^{\frac{p-1}{2}})^2 \equiv_p (\pm 1)^2 = 1$ ■

Pokazali smo $a^{p-1} \equiv 1 \ (mod\, p) \ \Rightarrow \ a^{\frac{p-1}{2}} \equiv \pm 1 \ (mod\, p)$, ali ako je $a^{\frac{p-1}{2}} \equiv 1 \ (mod\, p)$ i $\frac{p-1}{2}$ parno, ovo možemo ponoviti i dobiti $a^{\frac{p-1}{4}} \equiv \pm 1 \ (mod\, p)$. Postupak ponavljamo sve dok je $a^{\frac{p-1}{2^i}} \equiv \pm 1 \ (mod\, p)$ i $\frac{p-1}{2^i}$ parno i dobijamo $a^{\frac{p-1}{2^{i+1}}} \equiv \pm 1 \ (mod\, p)$. Dobijamo kongruentan niz vrednosti $1,\ 1,\ \dots,\ 1,\ -1,\ \dots$, gde su uzastopne jedinice na početku $k \geq 0$ puta, a posle -1 dolazi bilo šta.

Miler-Rabinov test primalnosti: Neka je n neparan i $a \in Z$ tako da $NZD(a,\ n)$ = 1. Zapišemo $n - 1$ = $2^r d$, gde je d neparan i za svako $0 \leq j \leq r - 1$ računamo $a_j = a^{2^j d} \ (mod\, n)$. Broj n prolazi Miler-Rabinov test u bazi a ako je ispunjen jedan od uslova:

1. $a_0 = 1$
2. postoji $0 \leq s \leq r - 1$ tako da $a_s = -1$

Primetimo da je $a_{j+1} \equiv a_j^2 \ (mod\, n)$, pa na osnovu 1. sledi da su svi $a_j = 1$, a na osnovu 2. da su svi $a_j = 1$ za $j > s$. Za složen broj n koji prolazi Miler-Rabinov test u bazi a kažemo da je **jako pseudoprost broj** u bazi a.

18. Veza između pseudoprostih i jako pseudoprostih brojeva. Efikasnost Karmajklovog i Miler-Rabinovog testa

Ako je broj n jako pseudoprost u bazi a, onda je on i pseudoprost u bazi a.
△:
$a^{n-1} \equiv_n a_{r-1}^2 = 1$ ■

Kao posledica, efikasnost Miler-Rabinovog testa je veća nego efikasnost Karmajklovog testa. Teorema:

1. Ne postoji složen broj n koji je jako pseudoprost u svakoj bazi $a \in Z$ tako da $NZD(a,\ n)$ = 1.
2. Ako je n neparan složen broj, onda on može biti jako pseudoprost za najviše četvrtinu baza $a \in Z_n^*$.

Dakle, efikasnost Miler-Rabinovog testa je najmanje $1 - \frac{1}{4^k}$, gde je k broj testiranja.

19. Rivest-Šamir-Ejdelman kriptosistem

Rivest-Šamir-Ejdelman (RSA) algoritam pomoću kojeg Alisa šalje Bobanu poruku ili ključ:

- Boban tajno bira dva velika prosta broja p i q, računa $n = pq$ i računa $\phi(n)$ = $(p-1)(q-1)$. Zatim, bira broj $1 \leq e \leq \phi(n)$ tako da važi $NZD(e,\ \phi(n))$ = 1 i računa $d = e^{-1}\ (mod\ \phi(n))$. Alisi šalje javni ključ $(n,\ e)$, a d čuva kao svoj tajni ključ. U ovom trenutku Boban može da zaboravi p i q, ali nikako ne sme da ih objavljuje.
- Neka je poruka koju Alisa želi da pošalje $M < n$. Ona računa $N = M^e\ (mod\ n)$ i to šalje Bobanu.
- Boban pomoću tajnog ključa računa $N^d \equiv M^{ed} \equiv M\ (mod\ n)$. Ovde se koristi $ed \equiv 1\ (mod\ \phi(n))$ i Ojlerova teorema.

Cica vidi n, e i N, ali ne može da dođe do poruke M sve dok ne odredi d.

20. Prividno jednosmerne funckije. Navesti primer prividno jednosmerne funkcije

Funkcija $f(M) = M^e\ (mod\ n)$ je primer **prividno jednosmerne funkcije**. To znači:

- f je jednosmerna za Alisu i Cicu, tj. one ne mogu da odrede f^{-1} u realnom vremenu.
- f nije jednosmerna za Bobana, tj. on bi mogao da odredi f^{-1} jer ima podatak o faktorizaciji $n = pq$.

21. Fermaov metod faktorizacije

Fermaov metod faktorizacije pretpostavlja da je $n = pq$, gde su p i q slične veličine. Brojevi p i q ne moraju biti prosti, ali jesu ako govorimo o RSA. Zapišemo n = pq = $s^2 - t^2$, gde su $s = \frac{p+q}{2}$ i $t = \frac{p-q}{2}$ prirodni brojevi. Problem se svodi na nalaženje s i t, pri čemu je $s > \sqrt{n}$ i t

malo. U nizu $s_1 = [\sqrt{n}] + 1$, $s_2 = [\sqrt{n}] + 2$, ..., $s_i = [\sqrt{n}] + i$, ... tražimo najmanje s_i tako da je $s_i^2 - n$ potpun kvadrat, tj. $t_i = \sqrt{s_i^2 - n}$ ceo broj. Tada je $p = s_i + t_i$ i $q = s_i - t_i$.

22. Kriptoanaliza RSA Fermaovim metodom

U RSA važi: odrediti tajni ključ $d \Leftrightarrow$ naći inverz za množenje $\cdot_{\phi(n)} \Leftrightarrow$ naći $\phi(n) \Leftrightarrow$ rastaviti n $\Leftrightarrow$ naći pravi delilac od n. Osnovna pretpostavka RSA je da ne postoji efikasan način da se reši jedan od ovih problema, čime bi se rešio i svaki od njih. Elementarno rešeto je presporo jer je vreme kriptovanja $O(\log^3 n)$. Cica bi Fermaovim metodom faktorizacije mogla da dobije p i q jer je n deo javnog ključa. Sada veoma lako računa $\phi(n) = (p-1)(q-1)$ na osnovu čega nalazi tajni ključ $d = e^{-1}\ (mod\ \phi(n))$ jer je i e deo javnog ključa. Na kraju, čita poruku M na isti način kao i Boban: $N^d \equiv M^{ed} \equiv M\ (mod\ n)$.

23. Polardov $(p-1)$-metod

Polardov metod omogućava da brzo faktorišemo prirodan broj n pod pretpostavkama da n ima prost činilac p tako da je $p-1$ B-gladak i da je $B \ll n$ (npr. $B \sim \log n$). Pre primene Polardovog algoritma treba izračunati vrednost m = $NZS(1,\ 2,\ \ldots,\ B)$. U kanonskoj faktorizaciji m učestvuju samo prosti brojevi p koji dele neki od brojeva $1,\ 2,\ \ldots,\ B$, pa je $p \leq B$. Ako je α najveći stepen tako da $p^\alpha | m$ onda p^α deli neki od brojeva $1,\ 2,\ \ldots,\ B$ pa je $p^\alpha \leq B$, odnosno $\alpha \leq \log_p B$. Dakle:

$$m = \prod_{p\ prost,\ p \leq B} p^{[\log_p B]}$$

Svi B-glatki brojevi dele m. Na osnovu $p-1 | m$ i MFT ($a^{p-1} \equiv 1\ (mod\ p)$) važi $a^m \equiv 1\ (mod\ p)$. Metod:

1. Biramo $2 \leq a \leq n-1$ tako da $NZD(a,\ n) = 1$
2. Računamo $x = a^m - 1\ (mod\ n)$. Ako je $x = 0$ vraćamo se na prethodni korak i biramo novo a, a inače prelazimo na sledeći korak.
3. Računamo g = $NZD(n,\ x)$. Ako je $g = 1$ pretpostavka o B-glatkosti nije ispunjena i eventualno se može pokušati sa većim B. Ako je $g = n$ probati sa drugim a, tj. vratiti se na prvi korak. Ako je $g \neq \{1,\ n\}$ onda će g biti traženi pravi delilac od n. Vrednost g ne zavisi od p, već samo od n i a koje biramo proizvoljno.

Ako je pretpostavka o B-glatkosti ispunjena, algoritam će dati $g \in \{2,\ 3,\ \ldots,\ n-1\}$.

24. Zašto se javni ključ $n = pq$ u RSA ne može izabrati tako da ne bude osetljiv na napad Polardovim metodom

RSA je ranjiv ako je $n = pq$ tako da je jedan od brojeva $p-1$ ili $q-1$ B-gladak. Možemo da proverimo B-glatkost $p-1$ za neko fiksirano B, a samim tim i za sve vrednosti manje od B, ali i dalje postoji opasnost od $(B+1)$-glatkosti. Ne možemo tražiti najmanje B koje ispunjava prethodni uslov jer bismo se opet vratili na faktorizaciju.

25. Intergritet poruke i heš algoritam

Integritet poruke podrazumeva da Boban treba da bude siguran da Alisina poruka nije usput promenjena, slučajno ili namerno. Promenjena poruka može da bude nečitljiva, ali čak i ako je čitljiva Boban neće primetiti da je promenjena. Iz tog razloga se u kriptosistem obično uključuje i **heš algoritam** koji Alisa primenjuje na poruku pre kriptovanja. Boban primenjuje heš algoritam na dešifrovanu poruku i upoređuje dobijenu vrednost sa Alisinim rezultatom. Cica može da promeni šifrat, ali ne zna kako da promeni Alisinu vrednost heš algoritma. Heš algoritam se sastoji od više uzastopnih primena **heš funkcije**. Heš funkcija h je jednosmerna i otporna na koliziju. Slaba otpornost podrazumeva da je za dato x teško odrediti $y \neq x$ tako da $h(x) = h(y)$. Jaka otpornost podrazumeva da je teško odrediti različite x i y tako da $h(x) = h(y)$. Ulaz heš funkcije $h(x,\ y)$ su argumenti fiksirane dužine k i m, a izlaz je dužine m. Ulaz heš algoritma je poruka promenljive dužine, a izlaz ima fiksiranu dužinu koja je obično mnogo manja od dužine ulaza. Heš algoritmi se najčešće dobijaju **MD (Merkle-Damgor) konstrukcijom**:

- Poruka se deli na blokove $M_1,\ \ldots,\ M_n$ dužine m.
- Izabere se neka inicijalna vrednost K_0 (npr. 0...0) i računa se $K_1 = h(K_0,\ M_1)$. MAC (Message Authentication Code) je heš algoritam koji umesto podrazumevane inicijalne vrednosti koristi tajni ključ.
- Redom se računaju vrednosti $K_i = h(K_{i-1},\ M_i)$, a K_n će biti izlaz algoritma.

26. Autentikacija, digitalni potpis i sertifikat

Autentikacija je proces kojim se dokazuje da poruka dolazi od pravog pošiljaoca. Obuhvata:

- **digitalni potpis** koji povezuje poruku sa javnim ključem.
- **sertifikat** koji povezuje javni ključ sa konkretnom osobom. U pitanju je dokument koji se izdaje od ovlašćenog lica u kome piše "Osoba X koristi ključ Y".

27. Digitalni potpis pomoću RSA kriptosistema

Alisa treba da pošalje poruku Bobanu, a Boban treba da bude siguran u njen identitet. Koraci:

- Alisa generiše javni ključ $(n_A,\ e_A)$ i tajni ključ d_A kao u RSA.
- Boban generiše javni ključ $(n_B,\ e_B)$ i tajni ključ d_B kao u RSA.
- Alisa želi da pošalje kodiranu poruku $M < n_A,\ n_B$.
- Alisa računa $M_1 = M^{d_A}\ (mod\, n_A)$ i $M_2 = M_1^{e_B}\ (mod\, n_B)$ i šalje Bobanu M_2.
- Boban računa $M_3 = M_2^{d_B}\ (mod\, n_B)$ i $M_4 = M_3^{e_A}\ (mod\, n_A)$.

Važi $M_3 \equiv_{n_B} M_2^{d_B} \equiv_{n_B} M_1^{e_B d_B} \equiv_{n_B} M_1$, odnosno $M_3 = M_1$ pa važi $M_4 \equiv_{n_A} M_3^{e_A} \equiv_{n_A} M_1^{e_A} \equiv_{n_A} M^{e_A d_A} \equiv_{n_A} M$, odnosno $M_4 = M$.

28. Čemu služi heširanje prilikom digitalnog potpisa

Potpisivanje cele poruke M je dobro rešenje, jer Cica ne može da izdvoji Alisin potpis, međutim ovo rešenje je sporo. U praksi se obično ključ i digitalni potpis razmenjuju asimetričnim kriptosistemom, a poruka simetričnim, pa je ovaj način neizvodljiv. Zbog toga se najčešće potpisuje $H(M)$, gde je H heš algoritam. Sada poruka može ići simetričnim, a potpis asimetričnim kriptosistemom, ali je potpis vezan za heširanu poruku.

29. Kakvo poboljšanje donose eliptičke krive u sigurnosti kriptosistema, a kakve u kriptoanalizi

Mnogi kriptosistemi sa javnim ključem, kao što su Difi-Helman, Mesi-Omura i ElGamal, imaju unapređenu verziju koja se zasniva na **eliptičkim krivama**. One omogućavaju da se postigne ista zaštita sa manjim ključem koji koristi EK. Na primer, 256-bitni ključ zasnovan na EK menja 3072-bitni ključ. EK se koriste i u kriptoanalizi, posebno za napad na RSA. Čak i ako neki kriptosistem ne koristi EK, on može biti napadnut algoritmom koji koristi EK. Na primer, Polardov $(p-1)$-metod faktorizacije je spor ako n nema prost činilac p tako da je $p-1$ B-gladak. Sa EK će biti dovoljno da za malo s neki od $p+s$ bude B-gladak.

30. Definisati i nacrtati eliptičku krivu nad poljem realnih brojeva

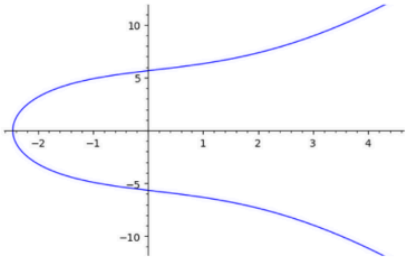
Eliptička kriva nad R je kriva definisana jednačinom

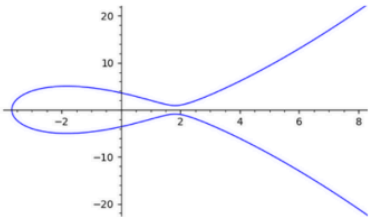
$$E: y^2 = x^3 + ax + b, \;\; a,\, b \in R, \;\; \Delta = -16(4a^3 + 27b^2) \neq 0$$

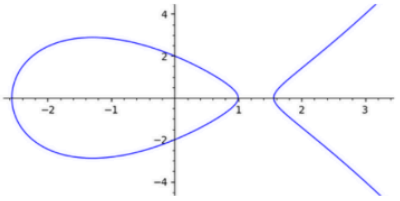
Na skup rešenja dodaje se i jedna "beskonačno daleka" tačka $\mathcal{O}$, pa je puna definicija:

$$E(R) = \{(x,\, y) \in R \times R \mid y^2 = x^3 + ax + b\} \cup \{\mathcal{O}\}$$

Primeri eliptičkih krivih:







31. Definisati eliptičku krivu nad konačnim poljem

Eliptička kriva nad F_q, pri čemu je q stepen prostog broja $p \neq 2,\, 3$, definisana je jednačinom

$$E(F_q) = \{(x,\, y) \in F_q \times F_q \mid y^2 = x^3 + ax + b\} \cup \{\mathcal{O}\}, \;\; a,\, b \in F_q, \;\; \Delta = -16(4a^3 + 27b^2) \neq 0$$

Uobičajeno je da se piše $E(F_q)$, ali je ispravnije $E(F_q;\, a,\, b)$ jer zavisi od sva tri parametra. Sada je operacije teže videti geometrijski, ali se $\oplus$ i $\ominus$ mogu računati algebarski.

32. Definisati operacije na eliptičkoj krivoj. Grupni zakon na eliptičkoj krivoj

Za $P = (x,\ y) \in E(R)$ definišemo **inverz** $\ominus P = (x,\ -y)$ i $\ominus \mathcal{O}$ = $\mathcal{O}$

Za $P,\ Q \in E(R)$ definišemo **sabiranje** $P \oplus Q$:

1. ako je $P = \mathcal{O}$: $\mathcal{O} \oplus Q = Q$
2. ako je $Q = \mathcal{O}$: $P \oplus \mathcal{O} = P$
3. ako je $Q = \ominus P \neq \mathcal{O}$: $P \oplus Q = \mathcal{O}$
4. ako je $P,\ Q \neq \mathcal{O},\ Q \neq P,\ \ominus P$: povučemo pravu l kroz P i Q i tada je $P \oplus Q = \ominus R$, pri čemu važi jedno od sledeća dva:
   - l seče EK u još tačno jednoj tački $R \neq P,\ Q$
   - l je tangentna na EK u jednoj od tačaka P i Q pri čemu je R upravo ta tačka
5. ako je $P = Q \neq \mathcal{O},\ \ominus P$: povučemo tangentu l na EK u tački P. Ona će preseći EK u još tačno jednoj tački $R \neq P$. Tada je $P \oplus Q$ = $P \oplus P$ = $2P$ = $\ominus R$

Grupni zakon na eliptičkoj krivoj: $(E(R),\ \oplus,\ \ominus,\ \mathcal{O})$ je Abelova grupa.

33. Haseova teorema za broj tačaka na eliptičkoj krivoj. Zašto rad sa grupom $(E(F_q),\ \oplus)$ nudi više mogućnosti od grupe $(F_q \backslash \{0\},\ \cdot)$

Haseova teorema: Kardinalnost grupe $E(F_q)$ je $q + 1 + s$, gde je $s \leq 2\sqrt{q}$. Dodatno, za svaku celobrojnu vrednost $s \in [-2\sqrt{q},\ 2\sqrt{q}]$ postoji EK $E(F_q)$ tako da je $|E(F_q)|$ = $q + 1 + s$.

Ideja je da se umesto grupe $(F_q \backslash \{0\},\ \cdot)$ koristi grupa $(E(F_q),\ \oplus)$ jer umesto fiksirane kardinalnosti $q - 1$ imamo slobodu $[q + 1 - 2\sqrt{q},\ q + 1 + 2\sqrt{q}]$. Takođe, stepenovanje g^n se menja sa nP = $P \oplus \ldots \oplus P$. Vrednost nP računamo ponovljenim dupliranjem tačke pomoću $O(\log n)$ operacija $\oplus$, a svaka operacija $\oplus$ se realizuje sa nekoliko sabiranja, oduzimanja i množenja.

34. Problem diskretnog logaritma nad eliptičkim krivama

Problem diskretnog logaritma nad EK: Ako je poznato P i nP odrediti n. U praksi, ovaj problem se rešava još sporije od diskretnog logaritma u F_q^*.

35. Kodiranje i dekodiranje podataka pomoću eliptičke krive

Ako je $q = p$ prost broj, ovaj metod će raditi uspešno sa verovatnoćom $1 - \frac{1}{2^k}$, pri čemu k sami biramo. U praksi je obično $k \in [30,\ 50]$. Poruka koja treba da se kodira se po potrebi deli na blokove m koji se prevode u numerički ekvivalent M. Maksimalna veličina bloka N je takva da važi $Nk < q$. M treba kodirati tačkom $(x_0,\ y_0)$ sa krive $E : y^2 = x^3 + ax + b$ nad Z_p. Pokušamo sa $x_0 = Mk$, pa ako $y^2 = x_0^3 + ax_0 + b$ ima rešenja, izaberemo jedno takvo y_0. Ukoliko nema rešenja, pokušavamo dalje sa $Mk + 1$, $Mk + 2$, ..., $Mk + k - 1$ sve dok ne pronađemo $(x_0,\ y_0)$. Kvadratna kongruencija ima rešenje u $\frac{1}{2}$ slučajeva, pa je verovatnoća da ćemo u k pokušaja bar jednom biti uspešni $1 - \frac{1}{2^k}$. Kodirana poruka je tačka $(x_0,\ y_0)$,

mada se u nekim implementacijama koristi samo x_0. U opštem slučaju kada $q = p^\alpha$ i $F_q \cong \{a_0 + \ldots + a_{\alpha-1}t^{\alpha-1} \mid 0 \leq a_0, \ \ldots, \ a_{\alpha-1} \leq p-1\}$ postupak je isti ali se $Mk + j$ zapiše u osnovi p kao $Mk + j$ = $a_0 + \ldots + a_r p^r$ ($r \leq \alpha - 1$ jer $M < q$) i pokuša se da se za polinom $x_0 = x_0(t)$ = $a_0 + \ldots + a_{r-1}t^{r-1}$ nađe polinom $y_0 = y_0(t)$ tako da $(x_0, \ y_0)$ pripada EK.

Dekodiranje: Od tačke $(x_0, \ y_0)$ treba rekonstruisati poruku m. U slučaju $q = p$ važi M = $[\frac{x_0}{k}]$ jer $[\frac{x_0}{k}]$ = $[M + \frac{j}{k}]$ pri čemu ne znamo šta je j, ali znamo da je iz $[0, \ k-1]$. U opštem slučaju $q = p^\alpha$, polinom $x_0(t)$ = $a_0 + \ldots + a_{r-1}t^{r-1}$ prevodimo u broj $a_0 + \ldots + a_{r-1}p^{r-1}$ i dalje radimo kao u prethodnom slučaju.

36. Difi-Helmanovo usaglašavanje ključa nad eliptičkim krivama

Javni ključ čine konačno polje F_q, eliptička kriva $E : y^2 = x^3 + ax + b$ nad F_q i tačka P = $(x_0, \ y_0) \in E(F_q)$, odnosno parametri $(q, \ a, \ b, \ x_0)$. Poželjno je da P bude generator grupe $(E(F_q), \ \oplus)$. Alisa i Boban biraju svoje tajne ključeve $a_A, \ a_B < |E(F_q)|$, a zatim računaju javne ključeve $a_AP, \ a_BP \in E(F_q)$ i razmenjuju ih. Usaglašeni ključ će biti K = $(a_Aa_B)P \in E(F_q)$. Alisa može da izračuna K = $a_A(a_BP)$, a Boban K = $a_B(a_AP)$. Cica vidi samo a_AP i a_BP, ali ne i K.

37. ElGamalov kriptosistem nad eliptičkim krivama

Javni ključ čine konačno polje F_q, eliptička kriva $E : y^2 = x^3 + ax + b$ nad F_q i tačka P = $(x_0, \ y_0) \in E(F_q)$, odnosno parametri $(q, \ a, \ b, \ x_0)$. Poželjno je da P bude generator grupe $(E(F_q), \ \oplus)$. Boban bira svoj tajni ključ $e < |E(F_q)|$ i pomoću njega računa javni ključ $eP \in E(F_q)$. Za svaki kodirani blok poruke $M \in E(F_q)$ Alisa generiše slučajan broj $k < |E(F_q)|$ i šalje Bobanu tačke kP i $M \oplus keP$, gde keP dobija množeći tačku eP sa k. Boban tačku keP može dobiti tako što kP pomnoži sa e. On sabira tačke $M \oplus kep$ i $\ominus keP$ i dolazi do M. Cica vidi samo eP, kP i $M \oplus kep$ i mora da reši problem diskretnog logaritma nad EK da bi došla do poruke M.

38. Lenstrin metod faktorizacije

Ako hoćemo da faktorišemo broj n za koji verujemo da je složen, možemo pretpostaviti suprotno - n je prost i onda važi da je Z_n polje. Izaberemo neku eliptičku krivu $E(Z_n)$ i neku tačku P sa te krive. Krenemo da računamo $2P, \ 3P, \ 4P, \ \ldots$ ili $2P, \ 4P, \ 8P, \ \ldots$ i negde će se pojaviti problem sa deljenjem. Kada se u imeniocu pojavi broj g koji nije invertibilan po modulu n, onda će $NZD(g, \ n) > 1$ biti pravi delilac n.

Lenstrin metod: Pretpostavka je da n ima prost činilac p tako da je kardinalnost $|E(Z_p)|$ B-gladak broj za neko malo B. Ideja je da ne treba pogađati koje m radi, već pokušati sa m = $NZS(1, \ \ldots, \ B)$ ili m = $B!$. Znamo da $|E(Z_p)|$ deli ove m, pa će biti mP = $\mathcal{O}$ na $E(Z_p)$, tj. pojaviće se imenilac g koji je deljiv sa p. Nećemo računati $mod\, p$, već $mod\, n$. Dakle, računamo mP na $E(Z_n)$. Može da se desi:

- izračunali smo mP bez problema: pretpostavka nije ispunjena, pokušati sa većim B ili drugom EK.

- kao imenilac pojavi se g deljivo sa n: promeniti P.
- kao imenilac pojavi se g koje nije deljivo sa n, ali nije ni invertibilno po modulu n: $NZD(g,\ n)$ je pravi delilac n.

Zero Knowledge proofs

1. Zero Knowledge proofs i ilustrativni primeri

Zero Knowledge proofs - ZKP (dokazi sa nula znanja) je kriptografska tehnika pomoću koje **dokazivač** dokazuje **verifikatoru** da zna neku informaciju bez otkrivanja same informacije. U pitanju su probabilistički dokazi, tj. postoji minimalna verovatnoća da dokazivač ne zna informaciju, a da će dokaz biti prihvaćen od strane verifikatora. Dokazi sa nula znanja mogu biti **interaktivni** i **neinteraktivni**. Neinteraktivni ZKP mogu biti **SNARKs** ili **STARKs**. Primeri:

- **Gde je Valdo?** - dokazivanje poznavanja lokacije nečega ili nekoga unutar složenog skupa podataka bez otkrivanja stvarne pozicije.
- **Ali Babina pećina** - dokazivač treba da ubedi verifikatora da zna tajnu reč, bez otkrivanja iste. Dokazivač ulazi u pećinu i bira jedan put, a verifikator čeka napolju i nasumično traži od dokazivača da izađe kroz jedan od puteva. Dokazivač može otvoriti vrata koristeći tajnu reč, čime dokazuje poznavanje reči bez njenog otkrivanja.
- **Prijatelj daltonista** - kako dokazati daltonisti da su sve lopte različitih boja bez otkrivanja samih boja? Jedan pristup je korišćenje serija zamena između lopti koje kontroliše daltonista, gde on može testirati da li dokazivač i dalje može identifikovati lopte kao različite.

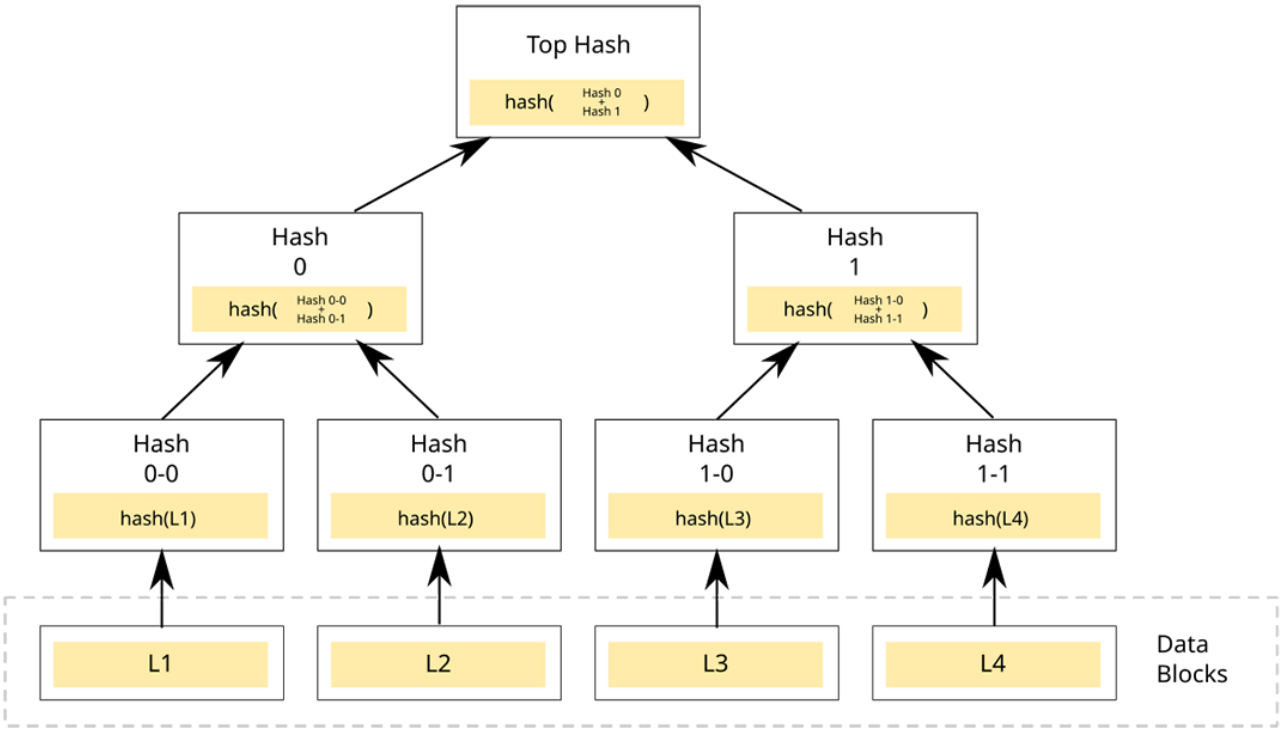
2. Primene ZKP-a

- **Blokčejn** - povećanje privatnosti transakcija.
- **Finansije** - bezbedna razmena podataka i verifikacija bez otkrivanja podataka. Na primer, aplikant za kredit može dokazati da mu je plata unutar određenog opsega bez otkrivanja tačnog iznosa. Slično, može se dokazati da je iznos plaćanja unutar neke granice, ali se ne prikazuje tačan iznos.
- **Online glasanje** - poboljšanje integriteta i privatnosti. Na primer, MACI (Minimum Anti-Collusion Infrastructure).
- **Decentralizovani identifikatori (DIDs)** - bolja verifikacija identiteta bez otkrivanja više informacija nego što je potrebno. Pruža pojedincu mogućnost da kontroliše pristup ličnim identifikatorima. Na primer, dokazivanje državljanstva bez otkrivanja poreskog ID-a ili detalja pasoša.
- **Mašinsko učenje** - verifikacija rezultata ML modela bez otkrivanja podataka ili samog modela. Omogućava korišćenje ML nad osetljivim podacima gde bi korisnik znao

rezultate koje daje model nad njegovim podacima, a da pritom ne otkriva te podatke trećoj strani.
- **Autentifikacija** - dokazivanje da znamo neke informacije bez otkrivanja istih. Jednom kada je ZK dokaz generisan korišćenjem javnih i privatnih ulaza, korisnik ga jednostavno može prezentovati radi autentifikacije svog identiteta kada mu je potrebno da pristupi nekoj usluzi.

3. Merkle Tree i ZK dokaz pripadnosti skupu

Merkle Tree je binarno stablo kod koga listovi sadrže heš transakcija, a unutrašnji čvorovi sadrže heš kombinaciju dece. Omogućava efikasnu pretragu sadržaja velike strukture podataka. Ispitivanje pripadnosti list-čvora stablu odvija se u logaritamskom vremenu. Blok se sastoji od zaglavlja i tela. Zaglavlje, između ostalog, sadrži koren Merkle stabla, a telo sadrži sve potvrđene informacije o transakcijama u bloku. Dokaz pripadnosti skupu se odvija preko protokola dokazivača i verifikatora, gde dokazivač ne sme otkriti informacije o samom dokazu. Ovo osigurava da se članstvo u skupu može dokazati bez otkrivanja detalja o sadržaju skupa ili informacija koje nisu namenjene javnosti.



4. Completeness, Soundness i Zero Knowledge

ZK dokaz mora zadovoljiti tri svojstva:

- **Completeness** - verifikator mora prihvatiti ispravan dokaz.
- **Soundness** - verifikator ne bi trebalo da prihvati netačan dokaz.
- **Zero Knowledge** - verifikator kroz javne parametre neće ništa naučiti o dokazu. Ovo osigurava da čak i nakon što je dokaz potvrđen kao ispravan, verifikator neće dobiti

nikakve dodatne informacije o sadržaju ili detaljima samog dokaza, osim onih koji su već javno poznati.

5. Ciklična grupa $(Z_p^*, \cdot)$

Ciklična grupa $(Z_p^*, \cdot)$ je Z_p^* = $\{1, \ldots, p-1\}$, gde je p prost broj, sa operacijom $\cdot$ definisanom kao $a \cdot b$ = $a * b \; mod \; p$, koja pritom sadrži makar jedan generator. Neki element a je **generator** ciklične grupe Z_p^* ako se svi ostali elementi te grupe mogu dobiti preko njega, tako da je $a_i = a^i \; mod \; p$. Svaka grupa Z_p^* gde je p prost broj je ciklična. Svojstva grupe:

- **zatvorenost** - ako su elementi a i b iz grupe, onda je i $a \cdot b$ u grupi.
- **asocijativnosti** - ako su elementi a, b i c iz grupe, onda je i $a \cdot (b \cdot c)$ = $(a \cdot b) \cdot c$ u grupi.
- **neutral** - ako je element a iz grupe, onda je i $a \cdot 1$ = a u grupi.
- **inverz** - ako je element a iz grupe, onda postoji element b iz grupe tako da je $a \cdot b = 1$.

Generatore ciklične grupe možemo brzo naći koristeći sledeću teoremu: Element g ciklične grupe Z_p^* će biti generator te grupe akko $g^{\frac{p-1}{q}} \neq 1 \; mod \; p$, za svaki prost broj q tako da $q \mid p-1$.

6. Problem diskretnog logaritma

Neka je g generator ciklične grupe Z_p^* = $\{g, \ldots, g^{p-1}\}$, gde je p prost broj. Ako su g i p uzajamno prosti brojevi, na osnovu MFT važi $g^{p-1} = 1 \; mod \; p$ i Z_p^* = $\{1, \; g, \; \ldots, \; g^{p-2}\}$. Broj x će biti **diskretni logaritam** od b u bazi g ako važi $\log_g b = x \Leftrightarrow g^x = b$ u grupi Z_p^*, gde je b element te grupe.

7. Eliptičke krive nad konačnim poljem

Eliptičke krive nad konačnim poljem F_q^* definisane su sa:

$$E(F_q^*) : y^2 = x^3 + ax + b, \;\; a, \, b \in F_q^*$$

Tačke eliptičke krive nad konačnim poljem su samo tačke na grafiku, odnosno nema krive u pravom smislu. Nad tačkama EK definisane su operacije $-$, $+$ i nP. Dokazano je da skup tačaka u ECC (Elliptic Curve Cryptography) uvek formira Abelovu grupu sa sledećim svojstvima:

- **zatvorenost** - ako tačke P i Q pripadaju EK, onda i $P + Q$ pripada EK.
- **asocijatinost** - ako tačke P, Q i R pripadaju EK, onda važi $(P + Q) + R$ = $P + (Q + R)$.
- **identitet** - postoji identični element 0 takav da je $P + 0 = P$.
- **inverz** - svaka tačka P koja pripada EK ima inverz Q takav da je $P + Q = 0$.
- **komutativnost** - ako tačke P i Q pripadaju EK, onda važi $P + Q$ = $Q + P$.

Pretpostavljamo da su EK nad konačnim poljem ciklične.

8. Add and Double algoritam

Računanje nP je jako sporo za veliko n jer sabiramo P sa samim sobom n puta. Umesto toga, efikasniji je **Add and Double algoritam**: nP računamo tako što n zapišemo binarno i time svodimo račun na logaritamski broj sabiranja. Npr. $79P = 2^6P + 2^3P + 2^2P + 2^1P + 2^0P$

9. Multi-Scalar-Multiplication (bucket metod)

Usko grlo u algoritmima za dokazivanje kod većine EK zasnovanih na SNARK sistemima je **Multi-Scalar-Multiplication algoritam**. Naivni algoritam koristi Add and Double algoritam, ali najbrži pristup predstavlja varijanta Pipengereovog algoritma koji nazivamo **bucket metod**:

- **pozicioniranje skalara** - svaki skalar se particioniše na m delova, tako da svaki deo sadrži w bitova.
- **akumulacija** - paralelno obrađivanje skalara i tačaka i akumulacija rezultata u bakete.
- **optimizacija** - tabele rezultata, paralelizacija, izbor EK.

10. Problem diskretnog logaritma nad eliptičkim krivama

Problem diskretnog logaritma nad EK: Ako je poznato P i nP odrediti n. U praksi, ovaj problem se rešava još sporije od diskretnog algoritma u F_q^*.

11. Uparivanje na eliptičkim krivama

Neka je data EK $E(F_q)$, neka su G_1 i G_2 podgrupe od $E(F_q)$ reda p, gde je p prost broj, i neka su g_1 i g_2 generatori grupa G_1 i G_2, redom. Funkcija $e : G_1 \times G_2 \to G_T$, gde je G_T multiplikativna podgrupa od F_q reda p, je **uparivanje nad EK** ako važi:

1. $e(g_1,\ g_2) \neq 1$
2. $e(P + Q,\ R) = e(P,\ R) \cdot e(Q,\ R)$
3. $e(P,\ Q + R) = e(P,\ Q) \cdot e(P,\ R)$

Tipovi uparivanja:

- $G_1 = G_2$ (**simetrično uparivanje**)
- $G_1 \neq G_2$ i postoji efikasan homomorfizam $\phi : G_2 \to G_1$, ali ne postoji efikasan homomorfizam u suprotnom smeru.
- $G_1 \neq G_2$ i ne postoji efikasan homomorfizam između G_1 i G_2.

12. STARKs i SNARKs

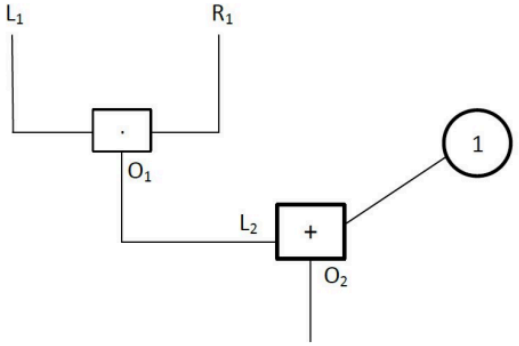
ZN SNARKs koriste sažeti dokaz i eliptičke krive, odnosno **povereno postavljanje (trusted setup)** što znači da se oslanjaju na početne parametre kojima se veruje. Veličina dokaza i vreme za verifikaciju zavise od aritmetičkog kola. Karakteriše ih brza verifikacija. **ZN STARKs** ne koriste eliptičke krive i ne zahtevaju povereno postavljanje, već koriste heš

funkcije. Zbog toga su transparentniji i manje podložni potencijalnim sigurnosnim rizicima vezanim za povereno postavljanje.

13. Aritmetizacija i sistem ograničenja (System Constraints) kod ZKP-a

Aritmetizacija je prevođenje problema u aritmetičko kolo, koje se prevodi u ograničenja, koja se prevode u polinome. Koristi operacije $+$ i $\cdot$ gde ulazni podaci i izlaz moraju biti iz konačnog polja. Verifikator proverava da li se izlaz aritmetičkog kola podudara sa javnom heširanom vrednošću dokazivača. Poenta je da zapišemo deo kola preko polinoma koji pokriva celo kolo tako što podešavamo koeficijente uz delove koje želimo prikazati na 1, a ostale koeficijente na 0. Ulaz za polinom $f(x)$ su kola. U primeru ispod, ulazi su 1 i 2 (imamo dva kola), a koeficijenti su: q_L i q_R za levu i desnu ulaznu vrednost sabiranja, q_M za množenje, q_C za konstantu i q_O za izlazni signal kapije. Na primer, za prikaz ograničenja za množenje uzećemo $q_M = 1$ i $q_O = -1$, a ostale stavljamo na 0 i dobijamo $f(x) = L(x)R(x) - O(x)$.

Example: I know an a such that $a*a+1=b$, for given b.



System constraints:

Gate constraints:
(1) $L_1 \cdot R_1 - O_1 = 0$
(2) $L_2 + 1 - O_2 = 0$

Copy constraints:
$L_1 = R_1$
$O_1 = L_2$

$$L_i \cdot q_{L_i} + R_i \cdot q_{R_i} + O_i \cdot q_{O_i} + q_{C_i} + L_i \cdot R_i \cdot q_{M_i} = 0.$$

Define $L(x), R(x), O(x)$:

$$L(1) = L_1,\ L(2) = L_2,\ R(1) = R_1,\ R(2) = R_2,\ O(1) = O_1,\ O(2) = O_2.$$

$$f(x) = L(x)\cdot q_L(x) + R(x)\cdot q_R(x) + O(x)\cdot q_O(x) + q_C(x) + L(x)\cdot R(x)\cdot q_M(x).$$

14. Komitmenti pomoću polinoma (Polynomial Commitments) kod SNARK-ova

Komitovanje predstavlja opredeljenje za neku vrednost pri čemu je ne otkrivamo ostalima, uz mogućnost da to uradimo kasnije. Komitovanje preko polinoma igra ključnu ulogu pri izgradnji efikasnih ZKP. Omogućava dokazivanje ispravnosti polinoma bez otkrivanja samog polinoma. Najčešći tip polinomskog komitovanja je **KZG**, a koriste se i Dory20, Dark20 i FRI.

15. Trusted setups kod Groth16 i PLONK-a

Trusted setup (povereno postavljanje) je procedura koja se obavlja jedanput radi generisanja podataka koji se koriste svaki put kada se neki kriptografski protokol pokrene. **Groth16** zahteva specifičan trust setup za svako aritmetičko kolo, gde se koristi nasumično odabrana pomoćna tačka sa EK kako bi se sprečilo lažiranje dokaza. **PLONK** nudi univerzalan i ažurirajući trusted setup, gde se koristi nasumično odabrana pomoćna tačka sa EK koja je nezavisna od kola. Međutim, PLONK ima veće veličine dokaza što utiče na troškove gasa u Eterijum mreži. **Transparentni setup** ne koristi tajne podatke, tj. pomoćne tačke sa EK.

16. Non-Interactive Preprocessing Argument sistem

Non-Interactive Preprocessing Argument sistem podrazumeva da se pre kreiranja dokaza generišu informacije koje dokazivač koristi za konstrukciju dokaza. Nakon toga, dokazivač šalje dokaz verifikatoru koji vrši verifikaciju. Za razliku od interaktivnih ZKP, ovde se sve završava u jednom koraku i ne postoji dodatna komunikacija između dokazivača i verifikatora.

17. KZG

KZG je kriptografska šema koja se koristi pri komitovanju preko polinoma. Omogućava dokazivanje ispravnosti polinoma bez otkrivanja samog polinoma. Generiše se komit za polinom i šalje verifikatoru koji proverava da li je vrednost polinoma u određenoj tački zaista 0.

Faze KZG:

- **setup**: biranje nasumične tačke i parametara $H_0 = G$, $H_1 = Gs$, $H_s = Gs^2$, ...
- **commit**: $com(f) = f(s) \cdot G$, gde s pripada konačnom polju F_p, a G je generator.
- **evaluate**: $f(x_0) = y$, gde je x_0 nula $f(x) - y$ i važi $x - x_0 \mid f(x) - y$ i $f(s) - f(z) = (s - z)h(s)$.

18. PLONK

PLONK je ZKP sistem koji pripada SNARK grupi. U pitanju je univerzalni sistem, što znači da je potrebno inicirati trusted setup samo jednom i on će važiti za svako aritmetičko kolo. Trusted setup se sastoji od inicijalnih parametara koji se koriste tokom verifikacije. PLONK se oslanja na komitovanje preko polinoma bez njegovog otkrivanja. PLONK koristi permutacije bazirane na Lagranžovim osnovama za definisanje ograničenja u kolu. Ograničenja se sastoje od ulaznih podataka svake kapije i od vektora koeficijenata za svaku kapiju. PLONK se može koristiti u Non-Interactive Preprocessing Argument sistemima.

19. Protokol Semafor

Semafor je ZKP protokol koji omogućava slanje signala u svojstvu člana grupe, bez otkrivanja sopstvenog identiteta. Takođe, onemogućava ponovno slanje signala. Ima primenu u tajnom glasanju, sistemima zaštite uzbunjivača, anonimnim decentralizovanim organizacijama i slično. Dakle, korisnik može da kreira identitet, doda ga u neku grupu i pošalje proverljiv, anoniman signal. ZKP mogu da osiguraju da je korisnik član date grupe i da već nije poslao signal. **Kolo semafora** predstavlja jezgro protokola. Njegovi delovi su:

- **dokaz pripadnosti** - svaki korisnik ima **identitetsku obavezu** koja se dodaje u Merkle drvo. Ona se formira koristeći identitetski poništavač i tajni ključ tog korisnika. Koren Merkle drveta je javan i svaki član može da dokaže pripadnost grupi, bez otkrivanja ko je.
- **anulirajući heš** - sprečava ponovno slanje signala.

- **signal**

Blockchain

1. Osobine blokčejna

Zbog jedinstvenih karakteristika blokčejna, on se koristi u različitim aplikacijama širom različitih industrija. Neke od važnih osobina su:

- Sigurnost - postiže se decentralizacijom svih informacija smeštenih u blokčejn mreži. Informacije ne mogu biti manipulisane, a heš jednog čvora je povezan sa prethodnim čvorom, pa promene u hešu jednog čvora dovode do promena u hešovima svih čvorova.
- Decentralizovanost - nema upravljačkog ili centralnog autoriteta u blokčejn mreži, već grupa čvorova kontroliše celu mrežu.
- Nepromenljivost - podaci u blokčejn mreži se smatraju trajnim zapisima transakcija. Jednom kada se blok doda u mrežu, ne može se izmeniti ili izbrisati. Svaki čvor blokčejn mreže ima repliku digitalne knjige i proverava validnost transakcija kada se doda u mrežu. Ako većina čvorova potvrdi transakciju, ona se dodaje u javnu knjigu. Ovo osigurava poverenje u blokčejn mrežu.
- Transparentnost - blokovi podataka su dostupni svakom čvoru i čvorovi mogu koristiti podatke prema svojoj potrebi.

2. Prednosti i mane blokčejn tehnologije

Prednosti:

- Integritet podataka - detalji transakcija koji su dodati u mrežu ne mogu se menjati, tj. informacije su nepromenljive.
- Verifikacija - podaci se skladište na decentralizovan način, pa svi korisnici blokčejn mreže mogu lako proveriti ispravnost podataka koristeći ZKP.
- Decentralizacija - budući da se hiljade uređaja koristi za skladištenje podataka u blokčejn mreži, ceo sistem, kao i podaci, visoko su otporni na bilo kakav tehnički kvar i zlonamerne napade.
- Pratljivost - blokčejn je napravljen tako da čuva trajni zapis svih događaja, pa je uvek moguće pratiti poreklo i istoriju bilo koje informacije.
- Sigurnost - blokčejn mreža je osigurana jer svaki entitet te mreže dobija jedinstveni heš koji je povezan sa prethodnim čvorom. Takođe, enkripcijska tehnika blokčejna otežava hakovanje cele blokčejn mreže.
- Brže procesiranje - tradicionalni bankarski sistemi zahtevaju mnogo vremena za procesiranje i završetak transakcija. Nakon uvođenja tehnologije blokčejna, brzina transakcija se značajno povećava.

- Bez mešanja treće strane - nijedna finansijska institucija ili vlada nema kontrolu nad bilo kojom kriptovalutom.
- Automatizacija - pametni ugovori i dostupnost podataka u svakom čvoru smanjuju složenost procesa validacije, što zauzvrat pomaže u automatizaciji procesa i poboljšava brzinu transakcija.

Mane:

- Potrošnja energije - potrošnja energije tokom procesa rudarenja je relativno visoka. Razlog je to što dodavanje novih blokova zahteva rešavanje složenih kriptografskih zadataka i sinhronizaciju sa mrežom.
- Nedozrelost - tehnologija blokčejna je u ranom stadijumu, pa obični ljudi, zbog nedostatka znanja, nemaju mnogo poverenja i nisu spremni da ulažu u nju. Međutim, aplikacije bazirane na blokčejnu efikasno funkcionišu u različitim industrijama.
- Vremenski zahtevno - da bi se novi blok dodao u blokčejn mrežu, rudari moraju izračunati vrednost nonce-a. U mnogim slučajevima, ovaj proces računanja je vremenski zahtevan i troši mnogo resursa. Teži se ubrzanju procesa korišćenja blokčejna u različitim industrijskim sektorima.
- Pravne formalnosti i standardi - mnoge zemlje su počele da rade na tehnologiji blokčejna, a još uvek ne postoje specifični standardi, pa dolazi do različitih pravila u različitim zemljama.
- Napadi od 51% - mogu se pojaviti neki sigurnosni napadi u blokčejn mreži, a napad od 51% je jedan od najčešćih napada. Ovaj napad se dešava kada entitet može upravljati sa više od 50% čvorova blokčejn mreže, što mu omogućava da poremeti blokčejn mrežu namerno modifikujući ili isključujući redosled transakcija.
- Robusnost mreže - sve aplikacije bazirane na blokčejnu imaju osnovnu poslovnu logiku. Ova logika opisuje kako aplikacije funkcionišu i to je fiksna logika koja se ne može preoblikovati nakon razvoja aplikacije.
- Težina razvoja - brza implementacija procesa nije moguća i potrebno je solidno poznavanje različitih programskih jezika za razvoj aplikacije bazirane na blokčejnu. Takođe, potrebno je implementirati kompleksne protokole za postizanje konsenzusa.
- Skladištenje - svaki blok ili transakcija se dodaju u mrežu, što povećava veličinu baze podataka, a veličina knjiga se vremenom povećava. Trenutno, Bitcoin zahteva oko 700GB prostora.
- Razmera - veličina blokova blokčejn mreže je fiksna pa oni mogu sadržati ogreničeni broj transakcija. U slučaju velikog broja korisnika i transakcija, može doći do zagušenja mreže i veće cene transakcija.

3. Primene blokčejn tehnologije

Bankarstvo i finansije:

- Međunarodna plaćanja - stvara neizmenjiv zapis o osetljivim informacijama ili podacima, što podržava upotrebu ove tehnologije u međunarodnim novčanim i platnim transferima.

- Trgovinsko finansiranje - omogućava preduzećima da lako vrše transakcije jedni s drugima izvan geografskih granica.
- Regulatorna usaglašenost i revizija - značajno smanjuje ljudsku grešku i osigurava integritet podataka. Pored toga, korisnici ili vlasnici podataka mogu menjati bilo koje podatke ili zapise tako što će uneti nove podatke.
- Zaštita od pranja novca - povećana bezbednost.
- Osiguranje - smanjenje prevara u osiguranju, pojednostavljenje zahteva, poboljšanje sigurnosti i ubrzanje autentifikacije.

Biznis:

- Upravljanje lancem snabdevanja - praćenje robe i promene vlasništva u realnom vremenu, tokom celog procesa lanca snabdevanja. Podržava sigurno, transparentno, bezbedno deljenje podataka i vidljivost ponude i potražnje.
- Zdravstvo - podaci o zdravlju pacijenata, kao što su godine, pol, izveštaji laboratorijskih testova i medicinska istorija, uvek moraju biti sigurno sačuvani. S obzirom na veliki broj pacijenata, ovi podaci ne bi trebalo da se mešaju sa drugim nesenzitivnim podacima i trebalo bi ih čuvati odvojeno.
- Nekretnine - proces prodaje može biti ubrzan proverom svih relevantnih detalja. Najvažnije, može smanjiti šanse za prevaru i održavati transparentnost u celom procesu kupovine i prodaje.
- Mediji - održavanje usaglašenosti sa revizijom, upravljanje digitalnim pravima, ugovorima, digitalnim identitetom i praćenje autorskih prava. Vlasništvo nad video zapisima, muzikom i drugim povezanim sadržajem može biti lako upravljano.
- Energija - izvršavanje transakcija snabdevanja energijom.
- Poljoprivreda - praćenje proizvoda i transfer vlasništva poljoprivrednih proizvoda, poput kafe, palminog ulja, soje i mnogih drugih, kao i praćenje zemljišta poljoprivrednika.

Vlada:

- Upravljanje evidencijama - podaci pojedinaca mogu se beležiti u blokčejnu i sve vlade i privatne institucije mogu biti čvorovi koji se odnose na te podatke. Ovo može obezbediti transparentan i siguran sistem.
- Glasanje - transparentan i siguran proces glasanja. Ako napadač ili zlonamerni korisnik želi pristupiti glavnom sistemu ili serveru, ne može manipulisati drugim čvorovima zbog svojstva blokčejna koji onemogućava promene.
- Porezi - efikasnije podnošenje poreske prijave, gde se informacije ili podaci čuvaju na privatnoj blokčejn mreži.
- Nevladine organizacije (NGO) - broj nevladinih organizacija rapidno raste i ljudi širom sveta doniraju dobrotvornim organizacijama. Blokčejn omogućava efikasno upravljanje transparentnim sistemom, tako da svi donatori mogu videti kako se njihov novac koristi.

Ostale primene:

- Veliki podaci - blokčejn podržava podatke koji se ne mogu menjati i svaki čvor mreže proverava podatke koji su na njemu sačuvani, pa je dobro rešenje za skladištenje velikih podataka.

4. Izazovi za usvajanje blokčejn tehnologije

- Napad na sajber bezbednost - sajber-napadi su i dalje mogući, ali većinom se dešavaju na javnim (permissionless) blokčejn mrežama koje su otvorene svima. Dozvoljeni (permissioned) blokčejn je sigurniji, jer samo odobreni korisnici ili čvorovi mogu da pristupe mreži.
- Dvostruko trošenje - situacija kada korisnik pokuša da iskoristi istu kriptovalutu u dve različite transakcije. Problem nastaje jer postoji vremenski razmak između slanja transakcije i njenog potvrđivanja, pa napadač može da pokuša da "prevari mrežu" tako što šalje dve transakcije istim sredstvima.
- Izazov modifikacije podataka - nemoguće je izvršiti promene što dovodi do račvanja (forking). Postoje dva tipa račvanja, softversko i tvrdo. Softversko račvanje znači da postoje promene u kodu ili protokolu i podaci prethodnih blokova su kompatibilni s njima. Tvrdo račvanje je promena u protokolu koja nije kompatibilna s prethodnim verzijama, što dovodi do mnogih izazova. Češće se dešava nego softversko račvanje.
- Regulacija GDPR (General Data Protection Regulation) - GDPR se zasniva na principu da je lični podatak osobe dat kompaniji ili organizaciji koja je kontrolor podataka. Kontrolor podataka je pravno odgovoran za zaštitu podataka i podleže zakonu o zaštiti podataka Evropske unije. Ovi kontrolori podataka moraju da ispune obaveze GDPR-a. GDPR, koji je usvojila EU, ima određene smernice, koje su teške za pratiti ako se koristi blokčejn. Blokčejn je distribuirana i decentralizovana mreža bez centralne vlasti sistema, što je u sukobu sa GDPR-om, kao i glavni nedostatak korišćenja blokčejn tehnologije u EU.
- Visoka potrošnja energije - u dozvoljenim blokčejn mrežama, rudari rešavaju zagonetku kako bi dodali novi blok u blokčejn mrežu. U većini mreža, postoji veliki broj rudara koji su uključeni u računanje i samo jedan rudar pobeđuje. Napor drugih rudara postaje beskorisan. Ovo je veoma neefikasno jer se mnogo energije troši tokom računanja. Iz tog razloga, mnogi korisnici ili organizacije ne pokazuju interes za pridruživanje blokčejn mreži.
- Kompresija podataka - čvorovi blokčejn mreže moraju imati mehanizam za kompresiju podataka pre njihovog skladištenja na blokčejn mreži. To je uglavnom zbog brzog povećanja veličine podataka.
- Stopa transakcija - transakcije po sekundi (TPS) je brzina kojom se podaci u blokovima upisuju u blokčejn mrežu. Bitcoin mreža ima 3-7 TPS, a Ethereum ima 10-20 TPS. Ovi brojevi nisu dovoljni jer se milioni transakcija izvršavaju svakodnevno, što se eksponencijalno povećava. Iz perspektive korisnika ili kompanija, TPS bi uvek trebao da bude visok, kako bi bilo koja kompanija ili organizacija mogla da podrži ogroman broj transakcija svakodnevno.

- Troškovi - potrošnja energije, tehnološka složenost i nedostatak blokčejn programera su neki od glavnih faktora koji čine implementiranje blokčejn tehnologije skupljim.
- Usaglašenost i pravna jasnoća - u većini slučajeva, blokčejn tehnologija nije direktno podržana bilo kojim zakonima i naredbama vlade. Moraju postojati specifična pravila i standardi koje definiše vlada za različite entitete poslovanja, tako da vlada može pratiti sve aktivnosti poslovanja koje koriste blokčejn tehnologiju. To sprečava usvajanje od strane kompanija, gde su transakcije vezane za pravne procese i pokrivene putem pametnih ugovora. Slično tome, podacima blokčejn mreže nedostaju regulacije, posebno ako su povezani sa bilo kakvim ličnim i finansijskim podacima korisnika ili kupaca.

5. Osnovni elementi blokčejna

- Decentralizovana mreža - blokčejn se oslanja na decentralizovanu mrežu računara (čvorova), koji doprinose računarskim resursima za skladištenje i obradu transakcija. Ovi računari funkcionišu autonomno i komuniciraju međusobno peer-to-peer (P2P).
- Kriptografija - kriptografske metode koje se koriste u blokčejnu pružaju matematičke dokaze koji garantuju da blokčejn funkcioniše kako je predviđeno. Kriptografski hešovi povezuju blokove podataka u lancu kako bi se sprečila promena podataka nakon upisa. Svaka transakcija je enkriptovana korišćenjem javnog ključa kako bi se verifikovao pošiljalac putem digitalnog potpisa i osiguralo da samo namenjeni primalac može primiti transakciju. Poverljivost transakcije se održava pomoću ZKP-a.
- Knjiga transakcija (ledger) - digitalna knjiga koja hronološki skladišti transakcije u blokovima koji se dodaju samo na kraj mreže.
- Distribuirani konsenzus - odluke, poput validnosti transakcije, donose se na osnovu konsenzusa među učestvujućim čvorovima u mreži, jer nema centralne vlasti. Blokčejn mreže koriste protokole konsenzusa kako bi osigurale saglasnost o svakoj transakciji ili bloku koji se dodaje. Proof-of-Work i Proof-of-Stake su popularni mehanizmi konsenzusa, pri čemu prvi daje moć odlučivanja čvorovima sa više računarske snage, a drugi čvorovima sa većim finansijskim udelom.
- Pametni ugovori - aplikacije na blokčejnu se implementiraju kao pametni ugovori. To su računarski programi koji funkcionišu autonomno, bez ljudske intervencije. Pametni ugovori izvršavaju uslove poput onih u pravnim ugovorima, osiguravajući automatsko i tačno izvršenje.

6. Kriptografski elementi blokčejn tehnologije

Neizmenjivost podataka u blokčejnu postiže se zahvaljujući informacijama o prethodnom hešu koje povezuju uzastopne blokove u mreži. Međutim, u teoriji, funkcija heša može imati različite ulazne vrednosti koje rezultiraju istim heš izlazom, što znači da se blok b_{i-1} može promeniti tako da njegova originalna heš vrednost $H(b_{i-1})$ ostane ista kao i pre, tj. da se poklapa sa hešom $b_i.prev$, koja je sačuvana u bloku b_i. U ovom slučaju, postupak provere bloka ne može otkriti promenu. Izbor funkcije heša je stoga ključan. Trebalo bi odabrati takvu funkciju koja obezbeđuje da je u praksi ovo nemoguće ostvariti. Funkcija heša H koja se koristi u blokčejnu mora biti kriptografska funkcija heša, a ne bilo koja proizvoljna funkcija

heša. Funkcija heša je jednosmerna funkcija koja za ulazni podatak proizvoljne dužine generiše izlazni podatak konstantne dužine, pri čemu se vrednosti predstavljaju kao binarne niske. Na primer, SHA256 je funkcija heša koja generiše binarnu nisku od 256 bitova. Kriptografska funkcija heša H je funkcija heša sa tri osobine:

- Otpornost na sudar - nemoguće je naći različite ulazne poruke x i y tako da $H(x) = H(y)$.
- Sakrivanje - na osnovu izlaza $c = H(x)$, nemoguće je pronaći ulaz x.
- Prijateljstvo sa slagalicom - ako znamo heš vrednost c ulazne poruke napravljene spajanjem r i x, i čak i ako znamo deo ulaza x, ne možemo rekonstruisati preostali ulaz r u vremenskoj složenosti brže od $2n$ gde je n binarna dužina izlaza c.

Zbog ovih svojstava, znajući $b_j.prev = H(b_{j-1})$, nemoguće je naći $b'_{j-1} \neq b_{j-1}$ tako da $H(b'_{j-1}) = H(b_{j-1})$. Sa H kao kriptografskom funkcijom heša, niko ne može promeniti postojeći blok bez detektovanja, pa su podaci u blokčejnu nepromenljivi. Kriptografske metode takođe imaju mnoge druge primene u radu blokčejna. Podsetimo se opklade između Alise i Boba u kojoj se postavlja pitanje šta ako Bob vara. Kriptografska funkcija heša H može rešiti ovaj problem varanja na sledeći način:

1. Alisa bira svoj predikat x ("glava" ili "rep")
   - Generiše tajni slučajni broj r (velike binarne dužine n).
   - Izračuna komit $c = H(r||kx)$.
   - Pošalje c Bobu, umesto slanja njenog predikata kao sirovih podataka.
2. Bob po prijemu c, šalje Alisi iskreni ishod bacanja novčića $x*$. Budući da je funkcija heša H kriptografska, on ne zna pravo predviđanje Alise i nema razloga za prevaru.
3. Alisa po prijemu $x*$, ako je njena pretpostavka tačna ($x = x*$), govori Bobanu da je pobedila tako što će mu poslati tajni broj r.
4. Bob po prijemu broja r proverava da li se angažman c koji je ranije dobio od Alise podudara sa $H(r||kx*)$.

Ovo rešenje naziva se šema komitovanja u kriptografiji. Kritično je da tajni r generisan od strane Alise mora biti veliki. Ako je binarna dužina n mala, Bobanu će trebati kratko vreme da iscrpno pokuša sve moguće vrednosti r i kombinuje ih sa x = "glava" ili x = "rep" da bi video koja kombinacija zadovoljava $c = H(r||kx)$. Kada se ta kombinacija pronađe, može prevariti Alisu govoreći joj da je ishod suprotna vrednost x pronađena u toj kombinaciji. Kada je n velik, iako x može imati samo dve moguće vrednosti, Boban ne može rekonstruisati tajni r zahvaljujući svojstvu "prijateljstvo sa slagalicom" funkcije H.

7. Problem distribuiranog konsenzusa

Problemi distribuiranog konsenzusa mogu se uočiti u različitim vrstama algoritama koji se koriste za postizanje konsenzusa u blokčejn mrežama. Neki problemi koji su prisutni u nekim od popularnih algoritama:

- Proof of Work (PoW) - ovaj algoritam zahteva veliku količinu računarske snage i vremena kako bi se rešio kriptografski problem. Napadi poput sebičnog rudarenja mogu

ometati mrežu, ali ne utiču direktno na neizmenjivost blokčejna.

- Proof of Stake (PoS) - zahteva manje računarske snage od PoW, ali postoji problem jer čvorovi s najvećim ulogom uvek dobijaju šansu da dodaju novi blok, što može dovesti do centralizacije. Takođe, dolazi i do "nothing at stake" problema, gde validatori mogu potvrđivati više grana istovremeno bez ikakvog troška.
- Delegated Proof of Stake (DPoS) - omogućava korisnicima da glasaju za delegate, ali može dovesti do centralizacije ako samo mali broj čvorova dobija većinu glasova.
- Practical Byzantine Fault Tolerance (PBFT) - dobar za privatne blokčejn mreže, ali ima ograničenu skalabilnost i nije pogodan za javne blokčejn mreže jer je bezbedan samo ako je manje od trećine čvorova zlonamerno.
- Stellar Consensus Protocol (SCP) - decentralizovan i ima nisku latenciju, ali zahteva kompleksan proces glasanja i konsenzusa, gde svaki čvor bira čvorove koje smatra "proverenim" i konsenzus se postiže kada dovoljna većina unutar skupa proverenih čvorova potvrdi transakciju.
- Raft - jednostavan za razumevanje i implementaciju, ali je zavisan od lidera čvora, pa ako lider postane zlonameran može ugroziti celokupni sistem.

8. Konsenzus zasnovan na dokazu rada (Proof of Work)

U PoW-u, svi čvorovi u mreži pokušavaju da reše kriptografsku heš funkciju. Ovde se koristi SHA256, koja generiše heš vrednost fiksne veličine od 256 bita. Da bi dodali čvor ili blok korišćenjem PoW-a, rudari (miners) pronalaze određeni broj kako bi rešili kriptografski problem. Ovaj proces je vremenski zahtevan i matematički težak, jer rudari koriste metodu grube sile kako bi pronašli broj koji rešava problem. Ciljni broj zavisi od težine mreže. U Bitcoinu, ciljni broj je obeležen na takav način da proces rudarenja može da se odvija svakih 10 minuta. Napadači ili zlonamerni korisnici mogu da utiču na PoW baziranu blokčejn mrežu ako preuzmu kontrolu nad 25% računarske snage pomoću sebičnih rudarskih napada. Međutim, ovaj napad ne utiče na neizmenjivost blokčejn mreže. PoW je bio veoma popularan u svetu kriptovaluta tokom mnogo godina. Proof-of-work mining:

$$\text{ID} = \text{SHA256}(\text{SHA256}(\text{block\_header})) < \frac{65535 << 208}{\text{difficulty}} = \text{TARGET},$$

gde su "block_header" podaci bloka koje treba da heširamo kako bismo stvorili validan heš. Što je veća težina ("difficulty"), to je teže pronaći validan heš. Nalazak zadovoljavajućeg identifikatora bloka ("ID") nije lak zbog dvostrukog SHA256 heširanja. Nijedan algoritam nije bolji od brute force-a, koji zahteva $O(2^{32})$. Ako povećamo težinu, vrednost TARGET će se smanjiti, čineći teže zadovoljenje nejednačine. Ako se blokovi kreiraju suviše lako, pošto se svaki novokreirani blok emituje u mrežu, trošak komunikacije bi bio veoma visok. Štaviše, kako rudari autonomno dodaju blokove, mnogi blokovi istovremeno kreirani od strane različitih čvorova će biti dodati na isti poslednji blok postojećeg blokčejna-a (njihova lokalna kopija). To ne samo što izaziva ozbiljnu nesaglasnost i ranjivost na dvostruko trošenje, već i gubi napore većine rudara zbog činjenice da samo jedan blok može biti dodat globalnom blokčejnu. Izazov pronalaženja dobrog ID-a koji zadovoljava nejednačinu od gore naziva se PoW problemom. PoW protokol takođe pomaže da se Bitcoin, kao valuta, zaštiti od inflacije.

Njegovo sporo kovanje i ograničena ponuda stvara ograničenu cirkulaciju, čime se cena čini vrednom.

9. Konsenzus zasnovan na dokazu ulaganja (Proof of Stake)

Ovaj algoritam konsenzusa je generalizovana forma PoW-a. U PoS-u, čvorovi se nazivaju validatorima. Validatori validiraju izvršenje kako bi mogli zaraditi naknadu za transakciju. U ovom algoritmu konsenzusa, nema konkurencije između validatora za rešavanje računarskog problema. Čvor se bira korišćenjem lutrije za rudarenje novog bloka na osnovu količine učesnika. Izabrani čvor koristi tehniku digitalnog potpisa za dokazivanje vlasništva nad ulogom. Stoga, nije potrebna velika računarska snaga u PoS-u. Međutim, postoji novi problem jer čvor sa najvećom količinom uloga uvek dobija priliku da doda novi blok. Na taj način, blokčejn indirektno ponovo postaje centralizovana mreža. Takođe, dolazi i do "nothing at stake" problema, gde validatori mogu potvrđivati više grana istovremeno bez ikakvog troška.

10. Stanje blokčejna: model zasnovan na transakcijama i na računima

Zavisno od toga kako je blokčejn dizajniran, struktura podataka stanja može biti različita. Može biti zasnovana na transakcijama (informacije o stanju sastoje se od liste transakcija) ili na računima (informacije o stanju sastoje se od stanja računa). U modelu zasnovanom na transakcijama, poznatom i kao UTXO (Unspent Transaction Output) model, svaka transakcija šalje sredstva jednom ili više primalaca. Stanje transakcije se sastoji od:

- Izlaza (outputs) - lista primaoca i koliko sredstava svaki dobija. Svaki izlaz postaje UTXO, tj. "novac" koji primalac može kasnije da potroši.
- Ulaza (inputs) - lista prethodnih UTXO koje pošiljalac koristi da finansira ovu transakciju. Ti UTXO su prethodno poslati pošiljaocu i još nisu potrošeni.

Jednostavno rečeno transakcija troši stare UTXO (ulazi) i kreira nove UTXO (izlazi). Model zasnovan na računima je intuitivniji. Sličan je modelu računa u banci. Stanje se sastoji od informacija o stanju za svaku adresu. Kada se desi transakcija, stanje se odmah ažurira i čuva informacije o saldu na računima pošiljaoca i primaoca. Stoga, saldo na računu adrese može se odmah dobiti bez ikakvog računanja. Transakcija u modelu zasnovanom na računima je mnogo jednostavnija od UTXO transakcije, jer uključuje samo adrese primalaca i iznos sredstava koji se šalje. Mnogo je brže proveriti da li pošiljalac ima dovoljno sredstava, što se postiže jednostavnim upoređivanjem dva broja: da li saldo pošiljaoca premašuje iznos koji se šalje. Model zasnovan na računima nudi jasnu prednost kada je reč o omogućavanju "pametnih ugovora". Za pametne ugovore, transakcija može biti ne samo transfer vrednosti, već i poziv funkciji proizvoljne logike, pa sadrži kod za izvršenje te funkcije. Procesiranje transakcije stoga uključuje izvršenje koda u transakciji. Kako su pametni ugovori računarski skupi, važnost jednostavnosti računanja je bitna. UTXO stvara računarski overhead jer sve transakcije moraju biti eksplicitno zabeležene, ali omogućava veću privatnost i bolju paralelizaciju transakcija.

11. Struktura blokčejn lanca

Knjiga transakcija blokčejna prati strukturu lanca. Podaci su organizovani u lanac podataka: b_1, b_2, b_3, … Kada je potrebno sačuvati nove transakcije, one se stavljaju u novi blok koji će biti dodat na poslednji blok postojećeg lanca. U blokčejnu zasnovanom na računima (npr. Ethereum), blok takođe sadrži informacije o stanju blokčejna (saldo svih računa u datom trenutku). Osim čuvanja podataka o transakcijama, stanju blokčejna ako je primenljivo, i potrebnih informacija o zaglavlju, blok ima dva važna atributa:

- ID bloka $b_i.id$ - postavljen je na heš vrednost sadržaja bloka koristeći kriptografsku heš funkciju H, tj. $b_i.id = H(b_i)$. Ova heš funkcija je unapred definisana i javno poznata. ID bloka ne mora nužno biti smešten u bloku jer se može izračunati iz sadržaja bloka.
- Prethodni heš $b_i.prev$ - postavljen je na ID prethodnog bloka b_{i-1} na koji se dodaje blok b_i, tj. $b_i.prev = b_{i-1}.id$. Ova informacija je ključna za održavanje integriteta podataka lanca.

Ako se bilo koji deo bilo kog bloka promeni nakon što je zabeležen u blokčejnu, to će biti otkriveno. Da bi se novi blok dodao u blokčejn, mora proći postupak koji se zove validacija bloka. Novi blok b_{i+1} je validan samo ako:

1. Prethodni heš je konzistentan: $b_{i+1}.prev = H(b_i)$
2. Sve transakcije u b_{i+1} su validne
3. Prethodni blok b_i je validan

Kao rezultat toga, postupak validacije za blok b_{i+1} zahteva proveru da li vrednost prethodnog heša smeštenog u bloku b_j odgovara heš vrednosti njenog prethodnog bloka b_{j-1} za sve $j \leq i+1$. Ako je raniji blok, recimo b_{j-1}, promenjen u odnosu na svoju originalnu vrednost, kada izračunamo njegovu heš vrednost $H(b_{j-1})$, naći ćemo da nije identičan sa hešom $b_j.prev$ smeštenim u bloku b_j. To je kršenje pravila i kao rezultat novi blok b_{i+1} se zaključuje kao nevažeći i ne dodaje se blokčejnu. Kao posledica, svi naredni blokovi posle b_j postaju nevažeći jer se lanac više ne može validirati dok se prethodni blok ne vrati u originalno stanje.

12. Procesiranje transakcije

Transakcije se obično pokreću putem korisnički prijateljske aplikacije koja može da interaguje sa mrežom blokčejna putem API poziva. Ova transakcija treba da bude poslata na čvor blokčejna (u praksi, na više čvorova u slučaju da jedan čvor može da zakaže ili da se ponaša neispravno) i biće obrađena na sledeći način:

- Svaki čvor X prilikom prvog prijema transakcije T_x radi:
  - Prosleđivanje transakcije T_x svojim susednim čvorovima.
  - Verifikaciju transakcije, tj. proverava da li pošiljalac transakcije T_x ima dovoljno sredstava za slanje. Ako ima, T_x se stavlja u mempool - red čekanja validnih transakcija koje treba da budu stavljene u novi blok.

    - Kreiranje blokčejna, tj. iz mempoola izvlači transakcije koje čekaju da se uključe u novi blok b i dodaje ovaj blok u postojeću knjigu transakcija na čvoru X. Blok b mora uključivati informacije o prethodnom hešu.
    - Ažuriranje bloka, tj. šalje novi blok b svojim susednim čvorovima.
- Svaki čvor Y prilikom prvog prijema bloka b radi:
    - Prosleđivanje bloka b svojim susednim čvorovima.
    - Validaciju bloka, tj. proverava validnost bloka b u postojećoj knjizi transakcija čvora Y. Ova validacija zahteva proveru konzistentnosti informacija o prethodnom hešu i validnosti svake transakcije u bloku b.
    - Dodavanje bloka b u knjigu transakcija na čvoru Y ako je validan. U suprotnom, ignoriše b.

Postupak obrade transakcije u blokčejnu je jednostavan i omogućava autonomnu obradu na čvorovima blokčejna. Međutim, ova jednostavnost dovodi do nekoliko problema sa konzistentnošću. Prvo, svaka transakcija je emitovana svim čvorovima i ista transakcija može biti dodata u različite blokove kreirane na različitim čvorovima. Moramo da osiguramo da svaka transakcija može biti dodata u blokčejn samo jednom. Drugo, različiti čvorovi paralelno kreiraju različite nove blokove da bi pokušali da ih dodaju na postojeći blokčejn. Moramo da osiguramo da samo jedan od njih bude dodat kao sledeći blok. Treće, različiti čvorovi mogu imati različite kopije blokčejna. Moramo da osiguramo da se slože oko jedne kopije kao globalno ispravne verzije. Da bismo rešili ove nekonzistentnosti, čvorovi se moraju redovno složiti oko trenutnog stanja blokčejna i to je ono što nazivamo postizanje konsenzusa. Dakle, potreban nam je protokol konsenzusa.

13. Sadržaj transakcije i bloka

Blok se sastoji iz dva dela:

1. blok zaglavlja. Svaki blok je povezan sa prethodnim blokom ili roditeljskim blokom i formira lanac. Prvi blok nema prethodni blok. Heš svakog bloka beleži se u zaglavlju odgovarajućeg bloka. Blok zaglavlja sadrži sve informacije o bloku. Postoji šest atributa blok zaglavlja:
    - verzija bloka - sastoji se od nekoliko pravila autentifikacije mreže blokčejna koja moraju biti poštovana od strane svih čvorova.
    - heš korena Merkle stabla - MT je struktura podataka koja se koristi za čuvanje jedne heš vrednosti za sve transakcije umesto čuvanja različitih heš vrednosti za različite transakcije. Takođe je poznato kao binarno heš stablo. MT kombinuje heš vrednosti transakcija u parovima, a heš vrednosti se kombinuju za sve transakcije i dobija se koren koji se zove heš korena MT.
    - nonce - slučajan broj od 32 bita. Može se koristiti samo jednom. Rudari pokušavaju da pogode validan nonce kako bi izračunali heš bloka, koji mora zadovoljiti određene zahteve. Prvi rudar koji pronađe validan nonce koji rezultira heš bloka ima pravo ili moć da doda novi blok u mrežu blokčejna. Rudar takođe biva nagrađen za dodavanje bloka.

- nBits - označava trenutnu težinu koja se koristi za kreiranje bloka.
- heš prethodnog bloka - odnosi se na heš vrednost od 256 bita roditeljskog ili prethodnog bloka.
- vremenska oznaka - odnosi se na Unix vreme koje se koristi za beleženje vremena kada rudar započinje proces rudarenja.

2. bloka tela. Telo bloka se uglavnom sastoji iz dva dela:
   - brojač transakcija - čuva broj transakcija u bloku.
   - transakcije - komunikacija između pošiljaoca i primaoca radi razmene imovine. U mreži blokčejna može biti više od jedne transakcije u bloku. Veličina bloka i transakcije određuju broj transakcija koje mogu biti smeštene u određenom bloku.

Transakcija se sastoji iz:

- Iznos - digitalna vrednost koju pošiljalac treba da prenese.
- Ulaz - detalji digitalne imovine koja želi da se prenese. Digitalna imovina treba da bude apsolutno prepoznatljiva i da uključuje vrednosti koje se razlikuju od drugih imovina.
- Izlaz - detalji računa primaoca. Uključuje vrednost digitalne imovine i identitet (ID) primaoca. Pored toga, izlaz sadrži neka pravila koja primaoc ne sme da prekrši kako bi primio povezanu vrednost.
- Heš (ID) transakcije - svaka transakcija ima poseban heš ili ID koji podržava digitalni potpis na osnovu kriptografije sa javnim ključem.

14. Merkle drvo

Transakcije su organizovane u bloku kao Merkle stablo - binarno stablo gde svaki unutrašnji čvor čuva heš vrednost vrednosti dece. U stablu za Bitcoin postoji osam transakcija $\{D_1, \ldots, D_8\}$, svaka smeštena u list stabla, $H_i = H(D_i)$. Unutrašnji čvorovi su npr. $H_{1-4} = H(H_{1-2}||H_{3-4})$ i $H_{1-2} = H(H_1||H_2)$. Bitcoin koristi SHA2 za heš funkciju H. Vrednost u korenu stabla je heš korena Merkle stabla koji je smešten u zaglavlju Bitcoin bloka. Postoje dva ključna svojstva. Prvo, svaka promena u podacima transakcije uzrokuje promenu heša korena Merkle stabla. Ako se blok promeni, bilo da je reč o podacima transakcije ili netransakcionom delu, heš bloka će se promeniti i biti otkriven. Drugo, brzo se proverava postojanje transakcije u bloku. Na primer, da bi dokazao da je transakcija D_7 u bloku, proverilac treba da pruži četiri vrednosti kao dokaz: H_7, H_{1-4}, H_{5-6} i H_8. Verifikator će izračunati:

$$x = H(H_7||H_8)$$

$$y = H(H_{5-6}||x) = H(H_{5-6}||H(H_7||H_8))$$

$$z = H(H_{1-4}||y) = H(H_{1-4}||H(H(H_7||H_8)||H_{5-6}))$$

i uporediti z sa H_{1-8}. Njihova jednakost znači da je transakcija D_7 u bloku. Za Merkle stablo sa n transakcija, za ovu verifikaciju potrebno je vreme $O(\log n)$. Trebalo bi $O(n)$ da smo naivno čuvali transakcije u strukturi sličnoj listi. Čvor u Bitcoinu može biti nerudarski čvor. Tu je samo da vrši transakcije sa mrežom i nezainteresovan je za stvaranje blokova kako bi

primio nagradu bloka. Budući da blok zaglavlje pruža dovoljno informacija za verifikaciju i vršenje transakcija, čvoru je potrebno samo blok zaglavlje, ne i pun sadržaj bloka. Pošto se stvarne transakcije ne čuvaju, zahtev za skladištenjem za takav čvor je skroman (samo 80 bajtova, koji se lako mogu smestiti u memoriju). Takođe je nizak i trošak komunikacije za povlačenje kopija lanca blokova od suseda (samo povlačenje blok zaglavlja).

15. Pametni ugovori

Mnoge aplikacije u stvarnom svetu mogu imati koristi od tehnologije blokčejna, a imati posebnu blokčejn mrežu za svaku pojedinačnu aplikaciju nije realno. Ovo je motivacija za Ethereum, odnosno postojanje univerzalnog blokčejn računara koji može pokretati aplikacije proizvoljnih namena. Da bi razvili takve aplikacije, programeri pišu računarske programe nazvane "pametni ugovori". Ethereum važi za blokčejn pametnih ugovora. Druge javne blokčejn mreže pametnih ugovora su Algorand, Tezos i Solana. Pametni ugovori se pišu koristeći programski jezik visokog nivoa (npr. Solidity, Viper, Flint, Bamboo). Solidity je najpopularniji jezik za mreže pametnih ugovora. On je Turing-kompletan, što znači da može simulirati bilo koju računsku operaciju. Za razliku od toga, Skript (programski jezik Bitcoina) nije Turing-kompletan, ali je vrlo lagan i pogodan za Bitcoin. Bitcoinu nije potreban univerzalni jezik jer je digitalna valuta jedini cilj Bitcoina. Izvorni kodovi razvijenih pametnih ugovora su vidljivi javnosti. Stoga nema ničega što treba sakriti u radu pametnih ugovora i ljudi mogu biti sigurni da će raditi tačno onako kako je programirano. Sa druge strane, u određenim slučajevima, složen pametan ugovor može sadržati greške i druge sigurnosne propuste koji nisu lako uočljivi. Blokčejn je nepromenljiv i popravka je naporna nakon što je aplikacija sa mnogo korisnika već razvijen. Stoga, profesionalni projekti treba da pažljivo razmotre bezbednosne aspekte pre puštanja pametnog ugovora u rad.

16. Skalabilnost blokčejna, rolapovi, L1/L2

Blokčejn ima tri glavna cilja:

- Decentralizacija - blokčejn pruža programiranje bez poverenja i ne oslanja se na centralnu tačku kontrole. Potrebno je da bude decentralizovan tako da čvorovi autonomno i jednako učestvuju.
- Sigurnost - blokčejn mora da funkcioniše kako se očekuje i da bude otporan na kvarove i napade. To znači da čak i ako deo čvorova pokuša da vara ili prestane da radi (tzv. bizantske greške), mreža kao celina i dalje treba da funkcioniše ispravno. Što više takvih problema mreža može da izdrži a da i dalje ostane bezbedna, to je sigurnost veća.
- Skalabilnost - blokčejn je namerno dizajniran kao "svetski" računar za sve ljude, kako bi pokretali aplikacije. On bi trebao da se skalira sa sve većim brojem transakcija, kako u smislu skladišta, tako i u smislu zahteva za računanje.

Međutim, prema osnivaču Ethereuma, ovi ciljevi ne mogu biti savršeno ostvareni, što je nazvao "trilemom skalabilnosti" za blokčejn. Ako želimo decentralizaciju, mreža mora da ima veliki broj čvorova koji učestvuju u validaciji blokova, ali što je više čvorova uključeno, to je teže i sporije postići konsenzus, što može ugroziti sigurnost i brzinu sistema. Kako se

decentralizacija i sigurnost postižu samo sa malim blokčejn mrežama (veličina mreže ili obim transakcija), cilj skalabilnosti nije ispunjen. Blokčejnovi često moraju da naprave kompromise u ovom trilemu. Na primer, Bitcoin nudi odličnu decentralizaciju i sigurnost, ali ne i skalabilnost. Solana žrtvuje decentralizaciju za skalabilnost, dok Ethereum nudi odličnu decentralizaciju, brzinu i sigurnost, ali njegova skalabilnost ima ograničenja.

Sloj 1 (Layer 1, L1) je osnovni blokčejn, kao što su Bitcoin ili Ethereum. On definiše pravila mreže, konsenzus, sigurnost i osnovnu obradu transakcija. **Sloj 2 (Layer 2, L2)** je dodatni sloj izgrađen povrh L1 koji omogućava bržu i jeftiniju obradu transakcija, dok i dalje koristi sigurnost L1 mreže. Promena osnovnog dizajna blokčejn mreže na sloju 1 (bilo da se radi o mehanizmu konsenzusa, strukturi blokova ili kriptografiji) uvek dolazi sa kompromisima zbog trileme skalabilnosti. Zbog toga je razvijen sloj 2, koji povećava kapacitet i brzinu transakcija bez menjanja sloja 1. Prvi predlog za ovakvo rešenje bio je Plasma (2016), a kasnije su se razvile naprednije tehnike, kao što su rolapovi - tehnika gde se više transakcija sa sloja 2 "spakuje" (roll up) u jedan paket i taj paket se periodično šalje na sloj 1. Ideja je sledeća:

- Obrada transakcija na sloju 2 - korisnici šalju transakcije na L2, gde se one obrađuju mnogo brže i uz niže troškove.
- Sigurnost i konačnost na sloju 1 - periodično se stanje obrađenih transakcija sa L2 upisuje u L1, čime se obezbeđuje sigurnost i sprečava prevara.

Na taj način, L2 nasleđuje sigurnost L1, dok u isto vreme omogućava veći protok transakcija. Na primer, Polygon je mreža sloja 2 koja radi povrh Ethereuma kao sloja 1, dok su popularni rolapovi Optimism, Arbitrum i zkSync.

17. ERC20 tokeni, ERC721 tokeni

Na svakoj blokčejn mreži postoji **osnovni (native) token** koji je vezan isključivo za tu mrežu. Na primer, Bitcoin mreža koristi BTC, dok je na Ethereum mreži osnovni token ETH. U mrežama koje podržavaju pametne ugovore, kao što je Ethereum, osnovni token ima ključnu ulogu jer omogućava korisnicima da implementiraju i koriste pametne ugovore. Svaka interakcija sa pametnim ugovorom zahteva plaćanje naknade u osnovnom tokenu, što se naziva gas fee, a visina te naknade zavisi od računarske složenosti operacije koja se izvršava. Pored osnovnog tokena, na blokčejn mrežama se mogu kreirati i **sekundarni tokeni**. Oni se implementiraju kroz pametne ugovore i imaju širok spektar primena - od programa lojalnosti, preko izdavanja digitalnih valuta centralnih banaka (CBDC), pa sve do različitih aplikacija u igrama, stablecoina ili digitalne imovine. Takvi tokeni funkcionišu tako što pametni ugovor vodi evidenciju o stanjima tokena na adresama korisnika, omogućava njihov prenos između naloga, može da podrži transakcije koje se obavljaju u ime vlasnika (uz njegovu dozvolu), a u nekim slučajevima i da izdaje nove ili uništava postojeće tokene, što se može koristiti za regulaciju inflacije. Da bi kreiranje ovih tokena bilo jednostavno i da bi različite aplikacije i servisi mogli da ih koriste na ujednačen način, uvedeni su standardi pametnih ugovora. Najpoznatiji standardi su razvijeni upravo na Ethereum mreži. **ERC20** definiše pravila za zamenjive tokene, gde su svi tokeni iste vrste jednaki i međusobno zamenljivi, pa se najčešće koristi za kriptovalute i stablecoine. **ERC721** je standard za

nezamenjive tokene, odnosno NFT, gde je svaki token jedinstven i ne može se zameniti drugim, što ga čini pogodnim za digitalno predstavljanje umetničkih dela, kolekcionarskih predmeta ili druge jedinstvene imovine. **ERC1155** predstavlja napredniji standard koji omogućava kombinaciju zamenjivih i nezamenjivih tokena unutar istog pametnog ugovora. On se često koristi u svetu igara, gde isti ugovor može upravljati i zamenjivim tokenima poput virtuelnog novca i nezamenjivim tokenima poput oružja ili specijalnih predmeta.

Na ovaj način, blokčejn omogućava postojanje osnovnog tokena koji osigurava funkcionisanje mreže, ali i čitav sistem dodatnih tokena koji proširuju mogućnosti aplikacija i olakšavaju razvoj različitih digitalnih rešenja.