METODE GENERISANJA VELIKIH PROSTIH BROJEVA

Bogdan Brkić, *Ministarstvo finansija Republike Srpske*
Miroslav Ćajić, *MikroByte d.o.o.*

Sažetak - Tajnost kriptosistema sa javnim ključem zasniva se na teško rješivim ili još uvijek nerješivim matematičkim problemima. Rezultat toga je da brojevi i njihova složenost imaju veoma veliku ulogu u održavanju sigurnosti sistema. Za asimetrične šifarske algoritme, kao što je RSA, po značaju se posebno izdvaja jedna klasa brojeva - prosti brojevi. Ako bi postojao efikasan algoritam za faktorizaciju prostih brojeva sigurnost asimetričnih algoritama bila bi dovedena u pitanje. U većini asimetričnih algoritama za generisanje ključeva se koriste veliki prosti brojevi sa stotinu i više decimalnih cifara. Predmet ovog rada su metode generisanja prostih brojeva i testovi za ispitivanje da li je broj prost.

Ključne riječi - Prosti brojevi, generisanje, testovi, kriptografija, asimetrični algoritmi, RSA, PKI

1. Uvod

Kriptosistemi sa javnim ključem zasnivaju se na upotrebi dva ključa - jedan za šifrovanje, a drugi za dešifrovanje. Zbog toga su i dobili naziv asimetrični. Asimetrični algoritmi svoju tajnost ne zasnivaju na nepoznavanju algoritma, već na upotrebi jednosmjernih funkcija tj. funkcija čije je pronalaženje inverznog postupka težak matematički problem, teško rješiv računarskim resursima u realnom vremenu. Uloga prostih brojeva u kriptosistemima sa javnim ključem je dvostruka. Prva je da se koriste za generisanje ključeva, a druga je da se neke njihove osobine koriste kao jednosmjerne funkcije. Za generisanje ključeva za osrednji kriptografski sistem sa javnim ključevima potrebno je mnogo prostih brojeva. Imajući ovo u vidu iznijećemo neke činjenice o prostim brojevima:

- prostih brojeva čija je dužina manja od n bita iznosi n/(ln n), pa tako prostih brojeva čija je dužina manja ili jednaka 512 bita ima oko 10^{151}
- vjerovatnoća da se izaberu dva ista prosta broja je veoma mala
- kada bi mogla da se formira baza svih prostih brojeva ili samo prostih brojeva čija je dužina npr. manja ili jednaka od 1024 bita, dobila bi se tolika količina podataka koju bi bilo nemoguće pretražiti s ciljem da se provali asimetrični algoritam

Predmet istraživanja ovog rada biće efikasno generisanje prostih brojeva velikih bitskih dužina. Pokazaćemo da je pogrešan način pronalaženja velikih prostih brojeva onaj kojim se provjera da li su slučajni brojevi-kandidati prosti brojevi, vrši faktorizacijom. Primjenom nekog od testova, koje ćemo analizirati, mnogo je lakše utvrditi da li je kandidat prost ili složen broj. Treba imati u vidu da se pod terminom "velike bitske dužine"

misli na brojeve čija je dužina hiljadu i više bita. Osnovne aritmetičke ali i složene matematičke operacije sa ovolikim brojevima se u bilo kom programskom jeziku ne mogu izvršavati standardnim naredbama, već je potrebno pisati posebne funkcije i procedure. S obzirom na tu činjenicu u uvodnim poglavljima daćemo pregled važnijih osobina prostih brojeva i osnovnih aritmetičkih operacija sa velikim binarnim brojevima. Zatim ćemo analizirati probabilističke i stvarne testove za ispitivanje da li je broj prost. Na osnovu poznatih algoritama za četiri probabilistička testa, upotrebom softverskog paketa "Wolfram MATHEMATICA 6", potkrijepićemo teorijske tvrdnje o međusobnim odnosima pouzdanosti probabilističkih testova i pokazaćemo da je Miler-Rabinov test najpouzdaniji i istovremeno među najefikasnijim testovima u pogledu zahtjeva za računskim resursima. Na kraju ćemo analizirati metode za generisanje velikih prostih brojeva i daćemo pregled trenutno najvećih poznatih prostih brojeva.

2. Prosti brojevi

Prema mogućnosti faktorizacije brojeve možemo podijeliti na proste i složene. Prosti brojevi su pozitivni cijeli brojevi koji su djeljivi samo sa samim sobom i 1. Ostali brojevi, koji pored sebe i jedinice imaju i druge djelioce, su složeni. Iako se 1 smatra prostim brojem, mnoge definicije podrazumijevaju da su prosti brojevi veći ili jednaki od 2. Primijetimo da je 2 jedini paran prost broj. Nekada se definiše i skup neparnih prostih brojeva odnosno skup prostih brojeva bez 2.

2.1. Važnije osobine prostih brojeva

Za potrebe daljnjeg razmatranja navešćemo neke osobine prostih brojeva:

- Funkcija koja daje broj svih prostih brojeva koji su manji od broja n označava se sa $\Phi(n)$ i naziva "Ojlerova fi funkcija" ili "Ojlerova totijent funkcija". Prema teoremi o prostim brojevima za prost broj n vrijedi $\Phi(n)=n-1$.
- Prema fundamentalnom teoremu aritmetike svaki pozitivan cijeli broj se na jedinstven način može predstaviti kao proizvod prostih brojeva. Svaki cijeli broj $n\geq 2$ ima faktorizaciju koja se sastoji od proizvoda potencija prostih brojeva u obliku: $n = p1^{e1} p2^{e2} \ldots pk^{ek}$ gdje su pi uzajamno prosti $(ei\geq 1)$
- Prema drugom Euklidovom teoremu broj prostih brojeva je beskonačan.
- Vjerovatnoća da je najveći prost faktor slučajnog broja x manja od $\sqrt{x}$ je ln2.
- Marsenovi prosti brojevi imaju oblik $2p-1$. Značajni su jer je u odnosu na brojeve koji nisu ovog oblika na računaru lako provjeriti da li su prosti. Najveći poznati prosti brojevi često imaju upravo ovaj oblik.
- Prosti brojevi blizanci su prosti brojevi p i $p+2$ tj. susjedni prosti brojevi čija je razlika 2.
- Prost broj p je jak prost broj ako: $p-1$ ima velik prost faktor r, $p+1$ ima velik prost faktor i ako $r-1$ ima velik prost faktor.

2.2. Područje upotrebe prostih brojeva u kriptografiji

Prosti brojevi se koriste u gotovo svim kriptosistemima za generisanje privatnih i javnih ključeva. Efikasno generisanje parametara kojima se generišu ključevi je osnovni preduslov za kriptosisteme sa javnim ključem. Jedan od primjera upotrebe prostih brojeva je generisanje para ključeva za RSA algoritam pomoću prostih brojeva p i q. Oni moraju biti dovoljno veliki i slučajni, u smislu da je verovatnoća da bilo koji od njih bude pogođen dovoljno mala, tako da napadač ne može steći prednosti kroz optimizaciju strategije

pretrage na osnovu tih verovatnoća. Prosti brojevi mogu se upotrijebiti kako bi se neke njihove dodatne osobine, iskoristile protiv nekih specijalizovanih napada.

2.3. Značaj jakih prostih brojeva za asimetričnu kriptografiju

U literaturi koja se odnosi na RSA algoritme, često se predlaže da se u izboru para ključeva koriste jaki prostih brojevi p i q za generisanje n. Jaki prosti brojevi imaju određene osobine koje čine proizvod n teško faktorizovanim. Razlog za ovakav izbor je to što su neke od metoda faktorizacije, kao što su *Polardove* p-1 i p+1 metode, naročito pogodne za proste brojeve p za koje p-1 ili p+1 imaju samo male proste faktore. Jaki prosti brojevi su otporni na ovakve napade. Napredak u faktorizaciji tokom poslednjih deset godina pokazuje opravdanost upotrebe jakih prostih brojeva. Za generisanje ključeva u RSA algoritmu preporučuje se upotreba velikih prostih brojeva reda veličine 100 decimalnih cifara i više. Poznato nam je da se generisanje ključeva za RSA algoritam odvija na način da se izaberu dva velika broja p i q iz kojih računamo n=p*q, a zatim odaberemo e i d uz poštivanje određenih uslova. Postoji dobar razlog zbog čega n ima tačno dva prosta faktora. Prvi razlog je jednostavnost formule za računanje $\Phi(n)=(p-1)(q-1)$, a drugi je teža faktorizacija n, jer je poznato da je faktorizaciju teže izvršiti ukoliko broj ima manje prostih faktora.

3. Modularna aritmetika

Kako se kongruencije dosta koriste u algoritmima kojima se određuje da li je broj prost, navešćemo neke osnovne pojmove i osobine.

3.1. Osnovni pojmovi i osobine kongruencija

Ako pri cjelobrojnom dijeljenju cijelog broja a sa cijelim brojem n dobijamo ostatak cijeli broj b, kažemo da je a je kongruentan sa b po modulu n, tj. a=b+kn za neki cijeli broj k.

$$a \equiv b \pmod{n}.$$

Operacija a mod n označava ostatak pri dijeljenju a sa n i može uzeti vrijednosti iz opsega [0,n-1]. Neke od osobina kongruencije su:

$$(a+b) \bmod n = ((a \bmod n) + (b \bmod n)) \bmod n$$
$$(a-b) \bmod n = ((a \bmod n) - (b \bmod n)) \bmod n$$
$$(a*b) \bmod n = ((a \bmod n) * (b \bmod n)) \bmod n$$
$$(a*(b+c)) \bmod n = (((a*b) \bmod n) + ((a*c) \bmod n)) \bmod n$$

3.2. Brzi algoritmi za modularnu eksponencijaciju

Problem je pronaći ostatak pri dijeljenju cijelog broja a^k sa n tj. pronaći $a^k \bmod n$. U obzir treba uzeti da i a i k mogu biti veoma veliki brojevi. Očigledna ali u većini slučajeva neizvodljiva metoda sastojala bi se u potenciranju a eksponentom k, a zatim dijeljenjem sa n i pronalažnjem ostatka. Kako u pojedinim kriptografskim operacijama broj a može imati stotinu i više decimalnih cifara ovakav metod je neizvodljiv i na najsavremenijim računarima.

3.2.1. Memorijski efikasna metoda

Ovaj metod zasniva se na činjenici da je:

$$a*b \bmod n = ((a \bmod n)*(b \bmod n)) \bmod n$$

Ukoliko ovo pravilo primijenimo na a^2, a^3, a^4, ... uočavamo algoritam koji je lako i vremenski prihvatljivo implementirati na personalnom računaru. Naime,
$a^2 \bmod n = ((a \bmod n)(a \bmod n)) \bmod n$,
$a^3 \bmod n = ((a \bmod n)(a^2 \bmod n)) \bmod n = ((a \bmod n)(((a \bmod n)(a \bmod n)) \bmod n)) \bmod n$,
$a^4 \bmod n = ((a \bmod n)(a^3 \bmod n)) \bmod n = ((a \bmod n)(((a \bmod n)(((a \bmod n)(a \bmod n)) \bmod n)) \bmod n)) \bmod n$, ...

Ovaj metod zahtijeva da se obavi veći broj operacija u odnosu na prethodno pomenutu prostu metodu, međutim, tokom svake operacije se radi sa znatno manjim brojevima što rezultuje brzim izvođenjem modularne eksponencijacije. Iako sa većim brojem operacija, ovaj algoritam je brži u odnosu na prosti.

3.2.2. Metod "s desna-u lijevo"

Ovaj metod smanjuje i broj operacija i utrošak memorije. Bazira se na predstavljanju eksponenta k u binarnom obliku.

$$k = \sum_{i=0}^{j-1} b_i 2^i \quad => \quad a^k = a^{\sum_{i=0}^{j-1} b_i 2^i} = \prod_{i=0}^{j-1} (a^{2^i})^{b_i} \quad => \quad x = \left(\prod_{i=0}^{j-1} (a^{2^i})^{b_i}\right) \bmod n$$

Primjer: Pronaći $11^{13} \bmod 53$!

$11^{13} = 11^1 * 11^4 * 11^8$
$11^{13} \bmod 53 = (11^1 * 11^4 * 11^8) \bmod 53 = ((11^1 \bmod 53) * (11^4 \bmod 53) * (11^8 \bmod 53)) \bmod 53$

$11^1 \bmod 53 = 11$
$11^2 \bmod 53 = (11^1 * 11^1) \bmod 53 = ((11^1 \bmod 53) * (11^1 \bmod 53)) \bmod 53 = (11*11) \bmod 53 = 15$
$11^4 \bmod 53 = (11^2 * 11^2) \bmod 53 = ((11^2 \bmod 53) * (11^2 \bmod 53)) \bmod 53 = (15*15) \bmod 53 = 13$
$11^8 \bmod 53 = (11^4 * 11^4) \bmod 53 = ((11^4 \bmod 53) * (11^4 \bmod 53)) \bmod 53 = (13*13) \bmod 53 = 10$
$11^{16} \bmod 53 = (11^8 * 11^8) \bmod 53 = ((11^8 \bmod 53) * (11^8 \bmod 53)) \bmod 53 = (10*10) \bmod 53 = 47$ ...

$11^{13} \bmod 53 = 11 * 13 * 10 = 1430 \bmod 53 = 1430 - 26*53 = 52$

4. Testovi za ispitivanje da li je broj prost

4.1. Generalni model odabira prostog broja

Za kriptografe je generisanje velikih prostih brojeva uvijek predstavljalo problem. Generalan metod za generisanje prostih brojeva je da se generiše broj n odgovarajuće dužine, a da se zatim provjerava da li je on prost. Najtrivijalnija provjera da li je broj prost bila bi da se provjeri da li je on djeljiv sa bilo kojim prostim brojem manjim od n. Uopšteni algoritam bi bio:

- Generisati neparan broj n koji je kandidat za prost broj
- Provjeriti da li je n prost

- Ako n nije prost vratiti se na prvi korak

Neznatna unapređenja mogu se postići načinom odabira sledećeg broja koji je kandidat za prost broj tokom povratka na prvi korak. Najprostiji način je provjera upotrebom sekvence n+2, n+4, n+6, ...

4.2. Testovi za ispitivanje da li je broj prost

Prethodan metod je neefikasan jer zahtijeva dosta vremena za provjeru i znatne računske resurse. Umjesto provjere da li su svi brojevi manji od posmatranog broja n njegovi djelioci, koriste se testovi za ispitivanje da li je broj prost. Postoje dvije grupe testova za ispitivanje da li je broj prost:

- "stvarni testovi" čiji je rezultat podatak da li je broj-kandidat prost ili nije prost i
- "probabilistički testovi" koji nam govore da je broj-kandidat ili "vjerovatno prost" ili je složen

Probabilistički testovi deklarišu cijeli broj n kao kandidata za prost broj sa nekom vjerovatnoćom, dok stvarni testovi obezbjeđuju matematički dokaz da je broj prost ili složen. Probabilistički testovi generalno zahtijevaju manje računskih resursa i izvršavaju se za kraće vrijeme od stvarnih testova.

4.2.1. Probabilistički testovi

Probabilistički testovi se najčešće koriste za ispitivanje da li je broj prost. Pored testiranog broja n, koriste i slučajno odabrane brojeve a. Ovakvim testovima se za prost broj nikada ne može dobiti rezultat da je složen, ali je moguće da složen broj testom bude prepoznat kao prost. Redukovanje ove greške može se postići ukoliko se test ponavlja sa nekoliko slučajno odabranih vrijednosti a. Za dva najčešće korištena testa, *Fermaov* i Miler-Rabinov, za bilo koji složeni broj n najmanje polovina brojeva koje je moguće birati za a detektuju da je n složen. Ovo znači da k ponavljanja smanjuje grešku vjerovatnoće za najviše 2^{-k} i može se učiniti proizvoljno malom povećavanjem broja pokušaja k.

Osnovni algoritam svakog probabilističkog testa bio bi:

- slučajno biramo broj a
- provjeravamo određene jednakosti (u zavisnosti od odabranog testa) koje uključuju brojeve a i n
- ako jednakosti ne vrijede test se prekida i broj a je "svjedok složenosti" broja n; ako jednakosti vrijede vraćamo se na prvi korak i ponavljamo postupak dok ne postignemo potrebnu sigurnost

Ako nakon određenog broja ciklusa ustanovimo da n nije složen broj, onda ga možemo proglasiti "vjerovatno prostim".

4.2.1.1 *Fermaov* test

Fermaov test bazira se na jednoj od osobina prostih brojeva, iskazanu kroz *Fermaovu* teoremu. Prema *Fermaovoj* teoremi, ako je n prost broj onda za svaki cijeli broj a takav da je $1 \leq a \leq n-1$ i uzajamno prost sa n važi:

$$a^{n-1} \equiv 1 \bmod n \quad \text{ili} \quad a^{n-1} - 1 \equiv 0 \bmod n.$$

Pronalaženje barem jednog broja a iz intervala $1 \leq a \leq n-1$ koji je uzajamno prost sa n, za koje ne vrijede $a^{n-1} - 1 \equiv 0 \bmod n$ garantuje da je n složen broj. Takav broj a se naziva "***Fermaov* svjedok složenosti**" broja n. Čak i ako za svako a vrijedi prethodna jednakost, to nije dovoljno da bi se tvrdilo da je n prost. Ako je n složen broj i postoji broj a, $1 \leq a \leq n-1$, takvo da je $a^{n-1} \equiv 1 \bmod n$, onda je broj a "***Fermaov* lažov**" da je broj n prost, a broj n je pseudoprost za bazu a. *Fermaovi* pseudoprosti brojevi mogu zadovoljiti *Fermaov* test. Primjer za ovu tvrdnju je složen broj 341=11*31 koji je pseudoprost za bazu 2 tj. $2^{340}=1 \bmod 341$. "***Karmajklov* broj**" je cijeli složen broj takav da za sve cijele brojeve a iz opsega [1,n-1] koji su uzajamno prosti sa n tj. NZD(a,n)=1 vrijedi $a^{n-1} \equiv 1 \bmod n$. Najmanji *Karmajklov* broj je 561=3*11*17. Ovih brojeva u opsegu $1\text{-}10^{15}$ ima 105.212, a u opsegu $1\text{-}10^{18}$ oko 1.400.000, odnosno približno jedan u 700 milijardi brojeva. Svaki *Karmajklov* broj ima najmanje tri prosta faktora. Nedostatak *Fermaovog* testa se ogleda u sledećem: ako je n *Karmajklov* broj tada su jedini svjedoci oni brojevi a ($1 \leq a \leq n-1$) koji nisu uzajamno prosti sa n, a ovo znači da ako su svi prosti faktori broja n veliki, čak i *Fermaov* test sa velikim brojem iteracija t može dati rezultat da je n prost broj. Neefikasnost *Fermaovog* testa je otklonjena u *Solovej-Štrasenovom* i *Miler-Rabinovom* testu koji počivaju na strožijim kriterijima.

4.2.1.2 *Solovej-Štrasenov* test

Solovej-Štrasenov test je test koji je postao popularan pojavom asimetričnih šifarskih sistema, a posebno pojavom RSA. On koristi *Jakobijev* simbol J(a,n) za ispitivanje da li je n prost broj, a zasnovan je na *Ojlerovom* kriteriju. Prema *Ojlerovom* kriteriju, ako je n prost broj onda za svaki cijeli broj a takav da je $1 \leq a \leq n-1$, i koji je uzajamno prost sa n važi:

$$a^{(n-1)/2} \equiv J(a,n) \ (mod\ n).$$

Složen broj n koji je uzajamno prost sa a i zadovoljava $a^{(n-1)/2} \equiv J(a,n) \pmod n$ kažemo da je "***Ojlerov* pseudoprost broj za bazu a**", a broj a je "***Ojlerov* lažov**". Ako a nije uzajamno prost sa n ili ako je $a^{(n-1)/2} \neq J(a,n) \pmod n$, kažemo da je a "**svjedok**" složenosti broja n.

Primjer: n=91=7*13 je *Ojlerov* pseudoprost broj za bazu 9 jer je $9^{45} \equiv 1 \bmod 91$, i J(9,91)=1.

Ponavljanjem testa t puta sa različitim vrijednostima a, vjerovatnoća da složeni broj n prođe svih t testova nije veća od $1/2^t$. Broj a koji dokazuje da je n prost broj je svjedok. Vjerovatnoća da slučajan broj a bude svjedok nije manja od 50%.

4.2.1.3 *Lemanov* test

Lemanov test je nešto jednostavniji od *Solovej-Štrasenovog* testa, jer se ne računa *Jakobijan*. Kao i kod *Solovej-Štrasenovog* testa vjerovatnoća da je broj a svjedok složenosti broja n nije manja od 50%. Test treba ponoviti t puta za različite vrijednosti broja a. Ako je u svih t koraka vrijednost $a^{(n-1)/2}$ 1 ili -1, ali nije uvijek 1, p je vjerovatno prost broj sa mogućnošću greške od $1/2^t$.

4.2.1.4 *Miler-Rabinov* test

Miler-Rabinov test je efikasniji, a podjednako je korektan kao i *Solovej-Štrasenov*, pa ga je u potpunosti i zamijenio. Zbog svoje jednostavnosti ovo je najčešće korišten probabilistički test za ispitivanje da li je broj prost. Naziva se još i "jaki pseudoprosti test" jer se zasniva na nekim osobinama jakih pseudoprostih brojeva.

Predstavimo broj n u obliku $n-1 = 2^s * r$ gdje je s najveći stepen broja 2 koji dijeli n-1. Jasno je da je r neparan broj. Jak pseudoprost broj za bazu a je neparan složeni broj n, ako postoji broj j, $0 \leq j \leq s-1$, takav da vrijedi ili $a^r \equiv 1 \bmod n$ ili $a^{2^j * r} \equiv n-1 \bmod n$. Broj a se naziva "jakim lažovom" da je broj n prost. Ako je pak $a^r \neq 1 \bmod n$ i $a^{2^s * r} \neq n-1 \bmod n$ za sve j, $0 \leq j \leq s-1$, onda je a "jak svjedok" da je n složen broj.

Primjer: $n=91=7*13=2^1*45+1$ => $s=1, r=45$

Postoji a=9 takvo da je n pseudoprost i to za bazu 9, jer je $9^{45} \equiv 1 \bmod 91$.
Skup strogih lažova da je 91 prost je {1, 9, 10, 12, 16, 17, 22, 29, 38, 53, 62, 69, 74, 75, 79, 81, 82, 90}.

Vjerovatnoća da neki složen broj prođe test brže opada u ovom testu nego u ostalim i iznosi $1/4^t$, gdje je t broj ciklusa. Tri četvrtine mogućih vrijednosti a sigurno su svjedoci. Za većinu slučajnih brojeva, oko 99,9% mogućih vrijednost a su svjedoci.

4.2.1.5 Komparacija probabilističkih testova

Uporedićemo *Fermaov, Solovej-Štrasenov, Lemanov i Miler-Rabinov* test, iako *Lemanov* test koji je veoma sličan *Solovej-Štrasenovom* možemo posmatrati kao pojednostavljeni *Solovej-Štrasenov*. Kroz jedan primjer ćemo posmatrati odnos "lažova" za pojedine testove:

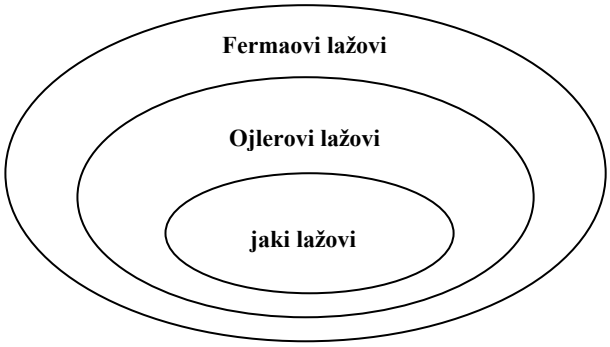
Za složen broj 65=5*13 imamo:

- Fermaovi lažovi su (16): 1, 8, 12, 14, 18, 21, 27, 31, 34, 38, 44, 47, 51, 53, 57, 64
- Ojlerovi lažovi su (8): 1, 8, 14, 18, 47, 51, 57, 64
- strogi lažovi su (6): 1, 8, 18, 47, 57, 64

Posmatranjem veće grupe složenih brojeva može se doći do sledećih zaključaka:

- *Ojlerovi* lažovi su *Fermaovi* lažovi (ali svi *Fermaovi* nisu i *Ojlerovi*)
- strogi lažovi su *Ojlerovi* lažovi (ali svi *Ojlerovi* nisu strogi)

odnosno, skup strogih lažova je podskup skupa *Ojlerovih* lažova, a skup *Ojlerovih* lažova je podskup skupa *Fermaovih* lažova nekog složenog broja n.



Slika 1. Odnos pouzdanosti probabilističkih testova

U pogledu korektnosti rezultata možemo zaključiti da *Miler-Rabinov* test nikada nije lošiji od *Solovej-Štrasenovog*, a *Solovej Štrasenov* nikada nije lošiji od *Fermaovog* za bilo koji broj n. Ovome u prilog idu i vjerovatnoće greške da se složen broj proglasi prostim, a koje iznose $1/2^t$ u *Fermaovom* i *Solovej-Štrasenovom* i $1/4^t$ u *Miler-Rabinovom* testu (t je broj ciklusa).

U pogledu efikasnosti i upotrebe resursa *Miler-Rabinov* test se može učiniti mnogo složeniji nego što jeste. On, zapravo, u pogledu broja računskih operacija zahtijeva isti broj operacija modularnog množenja kao i *Fermaov* test. U najgorem slučaju sekvenca operacija u *Miler-Rabinovom* testu je za s=1 odnosno r=(n-1)/2, kada treba izračunati $a^r \bmod n = a^{(n-1)/2} \bmod n$. Isti slučaj je i za M-R(n,1) i S-Š(n,1) gdje za *Solovej-Štrasenov* test treba izračunati $a^{(n-1)/2} \bmod n$, ali i *Jakobijev* simbol dodatno u odnosu na M-R.

Možemo zaključiti da je među navedenima najbolji *Miler-Rabinov* test, i to iz sledećih razloga:

- zahtijeva najmanji broj operacija
- relativno je jednostavan za implementaciju u odnosu na druge testove (ne zahtijeva npr. računanje *Jakobijevog* simbola)
- vjerovatnoća greške je najmanja $1/4^t$ u odnosu na druge testove $1/2^t$
- bilo koji strogi lažov je ujedno i *Ojlerov* lažov, pa *Miler-Rabinov* nikada nije nekorektniji u odnosu na *Solovej-Štrasenov*

4.2.1.6 Komparacija probabilističkih testova upotrebom programskog paketa MATHEMATICA

Programski paket "Mathematica", za razliku od nekih drugih paketa i kodova već ima riješene funkcije kao što su modularna eksponencijacija sa velikim brojevima (Powermod[a,k,n]) ili broj prostih brojeva koji su manji od nekog prirodnog broja (PrimePi[n]). Ispitivaćemo prostost prvih 10.000 brojeva, a za parametar a uzećemo da je $2 \le a \le 20$. Iako postoji više verzija definisanja *Solovej-Štrasenovog* testa, u zavisnosti od toga da li se uzima jedan od uslova $a^{(n-1)/2} \equiv J(a,n) (mod\ n)$ ili $a^{(n-1)/2} \bmod n = 1$ tj. $a^{(n-1)/2} \bmod n = n-1$, ili oba uslova, testovi i rezultati koji su dati u Tabeli 1. pokazuju da je uslov za $a^{(n-1)/2} \bmod n$ "jači" od uslova $a^{(n-1)/2} \equiv J(a,n) (mod\ n)$. Ovo znači da u *Solovej-Štrasenovom* treba uzeti uslov sa *Jakobijanom*, a u *Lemanovom* uslov sa $a^{(n-1)/2} \bmod n$. Nije potrebno ispitivati oba uslova u jednom testu, jer se ne postižu bolji rezultati, a algoritam se zbog povećanog broja operacija usporava.

```
FermaovKriterij[n_, a_] := If[2 ≤ a ≤ n - 1 && GCD[a, n] == 1 , If[PowerMod[a, n - 1, n] == 1, True, False], False]

LemanovTest[n_, a_] := If[2 ≤ a ≤ n - 1 && OddQ[n] == True, If[GCD[a, n] == 1, r = PowerMod[a, (n - 1) / 2, n];
   If[r ≠ 1 && r ≠ n - 1, False, True], False], False]

SolovejStrasenTest[n_, a_] := If[2 ≤ a ≤ n - 1 && OddQ[n] == True, If[GCD[a, n] == 1, r = PowerMod[a, (n - 1) / 2, n];
   J = JacobiSymbol[a, n]; If[r ≠ Mod[J, n], False, True], False], False]

SolovejStrasenTest2[n_, a_] := If[2 ≤ a ≤ n - 1 && OddQ[n] == True, If[GCD[a, n] == 1, r = PowerMod[a, (n - 1) / 2, n];
   If[r ≠ 1 && r ≠ n - 1, J = JacobiSymbol[a, n]; If[r ≠ Mod[J, n], False, True], True], False], False]

MilerRabinTest[n_, a_] := (If[2 ≤ a ≤ n - 1 && OddQ[n] && GCD[a, n] == 1,
   k = FactorInteger[n - 1];
   s = k[[1, 2]];
   r = (n - 1) / (2 ^s);
   z = PowerMod[a, r, n];
   If[z ≠ 1 && z ≠ n - 1,
    j = 1;
    While[j ≤ s - 1 && z ≠ n - 1, z = Mod[z^2, n]; If[z == 1, Return[False]] j ++;];
    If[z ≠ n - 1, Return[False]]],
   Return[False]];
  Return[True])

Opseg = 10 000;
Prostih = PrimePi[Opseg];
Print["Prostih brojeva <= ", Opseg, " ima ", Prostih];
For[a = 2, a ≤ 20, a++,
 fVjerovatnoProstihBrojeva = PrimePi[a];
 lVjerovatnoProstihBrojeva = PrimePi[a];
 sVjerovatnoProstihBrojeva = PrimePi[a];
 s2VjerovatnoProstihBrojeva = PrimePi[a];
 mVjerovatnoProstihBrojeva = PrimePi[a];
 For[n = 1, n ≤ Opseg, n++,
  If[FermaovKriterij[n, a], fVjerovatnoProstihBrojeva++];
  If[LemanovTest[n, a], lVjerovatnoProstihBrojeva++];
  If[SolovejStrasenTest[n, a], sVjerovatnoProstihBrojeva++];
  If[SolovejStrasenTest2[n, a], s2VjerovatnoProstihBrojeva++];
  If[MilerRabinTest[n, a], mVjerovatnoProstihBrojeva++]
 ];
 Print["a = ", a,
  " FERM=", fVjerovatnoProstihBrojeva, " Karm=", fVjerovatnoProstihBrojeva - Prostih, " (", Round[(fVjerovatnoProstihBrojeva - Prostih) / Prostih * 100.00, 0.01], "%)",
  " LEHM=", lVjerovatnoProstihBrojeva, " pp=", lVjerovatnoProstihBrojeva - Prostih, " (", Round[(lVjerovatnoProstihBrojeva - Prostih) / Prostih * 100.00, 0.01], "%)",
  " S-S=", sVjerovatnoProstihBrojeva, " pp=", sVjerovatnoProstihBrojeva - Prostih, " (", Round[(sVjerovatnoProstihBrojeva - Prostih) / Prostih * 100.00, 0.01], "%)",
  " S-S-2=", s2VjerovatnoProstihBrojeva, " pp=", s2VjerovatnoProstihBrojeva - Prostih, " (", Round[(s2VjerovatnoProstihBrojeva - Prostih) / Prostih * 100.00, 0.01], "%)",
  " M-R= ", mVjerovatnoProstihBrojeva, " pp=", mVjerovatnoProstihBrojeva - Prostih, " (", Round[(mVjerovatnoProstihBrojeva - Prostih) / Prostih * 100.00, 0.01], "%)"
 ]
]
```

Listing 1: "Mathematica" program koji omogućava poređenje probabilističkih testova

a	Udio pseudoprostih brojeva u skupu brojeva koji zadovoljavaju test (%)														
	Fermaov test			Lemanov test			Solovej-Štrasenov test			Solovej-Štrasenov test 2			Miler-Rabinov test		
2	1251	22	**1.79 %**	1243	14	**1.14 %**	1241	12	**0.98 %**	1243	14	**1.14 %**	1234	5	**0.41 %**
3	1252	23	**1.87 %**	1241	12	**0.98 %**	1238	9	**0.73 %**	1241	12	**0.98 %**	1235	6	**0.49 %**
4	1276	47	**3.82 %**	1251	22	**1.79 %**	1251	22	**1.79 %**	1251	22	**1.79 %**	1239	10	**0.81 %**
5	1248	19	**1.55 %**	1238	9	**0.73 %**	1237	8	**0.65 %**	1238	9	**0.73 %**	1234	5	**0.41 %**
6	1256	27	**2.20 %**	1246	17	**1.38 %**	1239	10	**0.81 %**	1246	17	**1.38 %**	1237	8	**0.65 %**
7	1244	15	**1.22 %**	1241	12	**0.98 %**	1237	8	**0.65 %**	1241	12	**0.98 %**	1235	6	**0.49 %**
8	1299	70	**5.70 %**	1268	39	**3.17 %**	1260	31	**2.52 %**	1268	39	**3.17 %**	1242	13	**1.06 %**
9	1278	49	**3.99 %**	1251	22	**1.79 %**	1251	22	**1.79 %**	1251	22	**1.79 %**	1244	15	**1.22 %**
10	1259	30	**2.44 %**	1245	16	**1.30 %**	1237	8	**0.65 %**	1245	16	**1.30 %**	1235	6	**0.49 %**
11	1257	28	**2.28 %**	1243	14	**1.14 %**	1237	8	**0.65 %**	1243	14	**1.14 %**	1234	5	**0.41 %**
12	1262	33	**2.69 %**	1246	17	**1.38 %**	1242	13	**1.06 %**	1246	17	**1.38 %**	1238	9	**0.73 %**
13	1254	25	**2.03 %**	1243	14	**1.14 %**	1239	10	**0.81 %**	1243	14	**1.14 %**	1235	6	**0.49 %**
14	1261	32	**2.60 %**	1251	22	**1.79 %**	1242	13	**1.06 %**	1251	22	**1.79 %**	1237	8	**0.65 %**
15	1248	19	**1.55 %**	1238	9	**0.73 %**	1234	5	**0.41 %**	1238	9	**0.73 %**	1232	3	**0.24 %**
16	1292	63	**5.13 %**	1275	46	**3.74 %**	1275	46	**3.74 %**	1275	46	**3.74 %**	1265	36	**2.93 %**
17	1254	25	**2.03 %**	1243	14	**1.14 %**	1241	12	**0.98 %**	1243	14	**1.14 %**	1240	11	**0.90 %**
18	1269	40	**3.25 %**	1253	24	**1.95 %**	1250	21	**1.71 %**	1253	24	**1.95 %**	1242	13	**1.06 %**
19	1267	38	**3.09 %**	1254	25	**2.03 %**	1244	15	**1.22 %**	1254	25	**2.03 %**	1241	12	**0.98 %**
20	1261	32	**2.60 %**	1249	20	**1.63 %**	1241	12	**0.98 %**	1249	20	**1.63 %**	1237	8	**0.65 %**

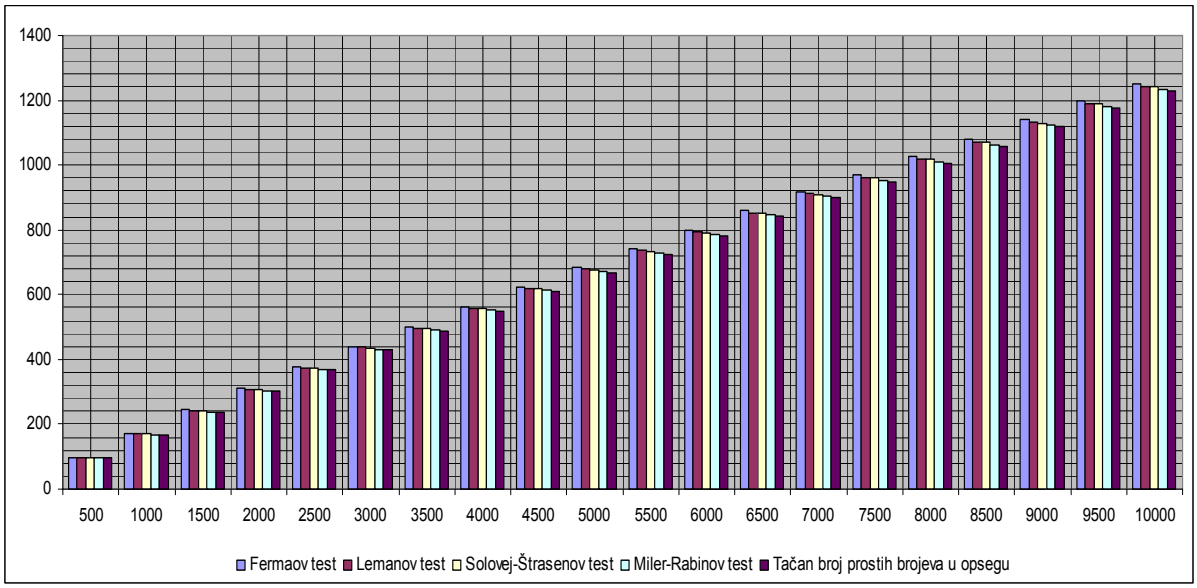
Tabela 1: Uporedni rezultati probabilističkih testova za 1≤n ≤10000 i 1≤a≤n-1 dobijeni paketom "Mathematica" na onovu programa datog listingom 1.

```
FermaovKriterij[n_, a_] := If[2 ≤ a ≤ n - 1 && GCD[a, n] == 1, If[PowerMod[a, n - 1, n] == 1, True, False], False]

LemanovTest[n_, a_] := If[2 ≤ a ≤ n - 1 && OddQ[n] == True, If[GCD[a, n] == 1, r = PowerMod[a, (n - 1) / 2, n];
   If[r ≠ 1 && r ≠ n - 1, False, True], False], False]

SolovejStrasenTest[n_, a_] := If[2 ≤ a ≤ n - 1 && OddQ[n] == True, If[GCD[a, n] == 1, r = PowerMod[a, (n - 1) / 2, n];
   J = JacobiSymbol[a, n]; If[r ≠ Mod[J, n], False, True], False], False]

MilerRabinTest[n_, a_] := (If[2 ≤ a ≤ n - 1 && OddQ[n] && GCD[a, n] == 1,
   k = FactorInteger[n - 1];
   s = k[[1, 2]];
   r = (n - 1) / (2 ^s);
   z = PowerMod[a, r, n];
   If[z ≠ 1 && z ≠ n - 1,
    j = 1;
    While[j ≤ s - 1 && z ≠ n - 1, z = Mod[z^2, n]; If[z == 1, Return[False]] j ++;];
    If[z ≠ n - 1, Return[False]]],
   Return[False]];
  Return[True])

Opseg = 10 000;
a = 2;
fVjerovatnoProstihBrojeva = PrimePi[a];
lVjerovatnoProstihBrojeva = PrimePi[a];
sVjerovatnoProstihBrojeva = PrimePi[a];
mVjerovatnoProstihBrojeva = PrimePi[a];
For[n = 1, n ≤ Opseg, n++,
 If[FermaovKriterij[n, a], fVjerovatnoProstihBrojeva++];
 If[LemanovTest[n, a], lVjerovatnoProstihBrojeva++];
 If[SolovejStrasenTest[n, a], sVjerovatnoProstihBrojeva++];
 If[MilerRabinTest[n, a], mVjerovatnoProstihBrojeva++];

 If[Mod[n, 500] == 0,
  Print[n,
   " ", fVjerovatnoProstihBrojeva,
   " ", lVjerovatnoProstihBrojeva,
   " ", sVjerovatnoProstihBrojeva,
   " ", mVjerovatnoProstihBrojeva,
   " ", PrimePi[n]]]
]
```

Listing 2: Distribucija pseudoprostih brojeva do 10.000, po intervalima od 500

n	Fermaov test	Lemanov test	Solovej-Štrasenov test	Miler-Rabinov test	Tačan broj prostih brojeva u opsegu
500	96	96	95	95	95
1000	171	170	169	168	168
1500	244	242	241	239	239
2000	310	308	307	303	303
2500	376	374	373	368	367
3000	441	437	436	431	430
3500	501	497	496	491	489
4000	562	558	557	552	550
4500	625	619	618	613	610
5000	685	679	678	673	669
5500	742	736	734	729	725
6000	800	794	792	787	783
6500	859	853	851	846	842
7000	918	912	910	904	900
7500	968	962	960	954	950
8000	1026	1019	1017	1011	1007
8500	1080	1073	1071	1064	1059
9000	1139	1131	1129	1122	1117
9500	1199	1191	1189	1182	1177
10000	1251	1243	1241	1234	1229

Tabela 2: Rezultati programa datog u listingu 2



Slika 2. Odnos pouzdanosti probabilističkih testova

Rezultati iz Tabele 1. nam govore da je strogih pseudoslučajnih brojeva manje od *Ojlerovih pseudoslučajnih*, a *Ojlerovih* manje od *Fermaovih* pseudoslučajnih brojeva. Manji postotak u redovima tabele znači da je neki test pouzdaniji od drugog testa. Na osnovu dobijenih rezultata zaključujemo da je *Miler-Rabinov* test najpouzdaniji (ima najmanju vjerovatnoću greške), a slijede ga *Solovej-Štrasenov, Lemanov i Fermaov*. Isti zaključak možemo donijeti i iz drugog testa (Listing 2). Vidimo da je broj verifikovanih prostih brojeva najbliži stvarnom broju prostih brojeva kada se koristi *Miler-Rabinov* test (Tabela 2. i Slika 2.), a najveće odstupanje je kod Fermaovog testa.

4.2.2. Stvarni testovi

Za razliku od probabilističkih testova gdje rezultat testiranja može biti da je broj složen ili prost ali sa određenom vjerovatnoćom, rezultat stvarnih testova je tvrdnja da je testirani broj n prost ili složen. Zbog toga se često nazivaju i "algoritmi dokazivanja da je broj prost". Ovi testovi su računski znatno složeniji od probabilističkih, pa se zato na broj n obično prvo primjenjuju probabilistički testovi, a zatim testovi dokazivanja da je broj prost. Cijeli broj n za koji se utvrdi da je prost nekim od algoritama dokazivanja da je broj prost nazivaju se "dokazano prosti".

Lukas-Lemerov test za testiranje _Mersenovih_ brojeva

Mersenovi brojevi su cijeli brojevi oblika 2^s-1, s≥2. *Mersenovi* brojevi koji su prosti zovu se *Mersenovi* prosti brojevi. *Mersenov* broj 2^s-1, s≥2, je prost ako i samo ako je s prost broj i ako niz cijelih brojeva definisanih sa $u_0=4$, $u_{k+1}=(u_k^2-1) \bmod n$ zadovoljava $u_{s-2}=0$ za za s≥2.

Testiranje da li je broj n prost upotrebom faktorizacije broja n-1

Ovim testom se pokazuje da je broj n prost broj upotrebom potpune ili djelimične faktorizacije broja n-1 za koju pretpostavljamo da je poznata. Izgleda besmisleno bazirati pretpostavku o tome da li je broj n prost oslanjajući se na faktorizaciju broja n-1, jer je logično da ako se može faktorizovati broj n-1 da isto možemo uraditi i sa brojem n i pokazati da li je prost ili nije. Međutim, faktorizacija broja n-1 može biti jednostavna za neke klase brojeva kao što su *Fermaovi* brojevi oblika $2^{2^k}+1$ ili za neke druge brojeve koji su "formirani" specifičnim metodama.

Cijeli broj n≥3 je prost ako i samo ako postoji cijeli broj a takav da je $a^{n-1} \equiv 1 \bmod n$ i $a^{(n-1)/q} \neq 1 \bmod n$ za svaki prosti djelitelj q broja n-1.

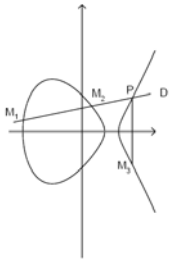
Test _Jakobijevih_ suma

Osnovna ideja je testirati skup kongruencija. Jedna od verzija *Jakobijevog* testa je algoritam sa slučajnostima čija je vjerovatnoća pronalaženja prostog broja $1-1/2^k$, a koji uvijek daje ispravan odgovor. Test je praktičan jer se za velike brojeve koji imaju nekoliko stotina decimalnih cifara može obaviti na personalnom računaru za nekoliko minuta. Ovaj test nije tako lako implementirati kroz softver kao što je to slučaj sa npr. *Miler-Rabinovim* testom.

Testiranje upotrebom eliptičkih krivih

Ovaj test se bazira na eliptičnim krivama i generalan je tj. može se upotrijebiti za testiranje brojeva koji nemaju neki poseban oblik. Od svih stvarnih testova ovaj test je najbrži. Elitptične krive možemo predstaviti jednačinom:

$$E(a,b) : y^2 = x^3 + ax + b$$



gdje su koeficijenti a i b takvi da je $4a^3 + 27b^2 \neq 0$. Najveći prost broj za koji je ovim testom 2006. godine potvrđeno da je prost ima 20.562 decimalnih cifara je:

$((((((2521008887^3+80)^3+12)^3+450)^3+894)^3+3636)^3+70756)^3+97220$

Ovaj distribuirani test je trajao 2219 dana ili blizu 6 godina na računaru sa AMD Opteron procesorom na 2.39 GHz.

AKS (*Agrawal–Kayal–Saxena*) test

AKS test je generalan test i bazira se na jednakosti: $(X-a)^n \equiv (X^n-a) \bmod n$

LLR (*Lucas–Lehmer–Ri*esel) i BLS (*Brillhart–Lehmer–Selfridge*) testovi

LLR je test za brojeve oblika $N = k2^n - 1$, $2^n > k$ i najbrži je deterministički test za brojeve ovog oblika. BLS je najbrži deterministički test za brojeve oblika $N = k2^n + 1$.

***Pepinov* test**

Pepinov test je namijenjen za provjeru da li je *Fermaov* broj $F_n = 2^{2^n} + 1$ prost broj. *Fermaov* broj je prost ako i samo je zadovoljen uslov: $3^{(F_n-1)/2} \equiv -1 \bmod F_n$

5. Generisanje prostih brojeva

5.1. Generisanje slučajnih kandidata za prost broj

Jedna od taktika za generisanje prostih brojeva bila bi:

- generisati slučajan n-bitni prost broj p
- postaviti bit najveće težine na 1 čime se osigurava tražena bitska dužina
- postaviti bit najmanje težine na 1 čime se osigurava da je generisani broj neparan
- provjeriti da li je p djeljiv malim prostim brojevima: 3, 5, 7, 11, ... (najčešće se ispituje djeljivost sa svim prostim brojevima manjim od 256, a najefikasnije je testirati djeljivost sa prostim brojevima manjim od 2000)
- za pet slučajnih vrijednosti a izvršiti *Miler-Rabinov* test
- ako p ne prođe test vratiti se na prvi korak i generisati novo p

Provjera djeljivost sa malim prostim brojevima nije obavezna, ali se sa njom odbacuje velik postotak neparnih brojeva prije nego što se dođe do sledećeg koraka, Miler-Rabinovog testa. Ispitivanje djeljivosti sa 3,5,7 može da odbaci 54% neparnih kandidata, ispitivanje djeljivosti prostim brojevima manjim od 100 odbacuje se 76%, manjim od 256 odbacujemo 80%, ... Odbacivanjem prostih brojeva koji su djelioci p, a manji su od n, odbacuje se 1.12/(ln n) kandidata.

5.2. Inkrementalno pretraživanje

Druga moguća realizacija bila bi da se nakon prvog generisanja kandidata p i njegovog eventualnog odbacivanja ne generiše novi slučajni kandidat, već da se izbor

novih kandidata vrši u okolini broja p. Za svakog novog kandidata se primjenjuje prethodni test (5.1.) a pokazuje se da je ispitivanje djeljivosti sa malim prostim brojevima nešto jednostavnije ako je ispitivanje već rađeno za prethodne kandidate koji su od p_k (p_{k-2}, p_{k-4}, ...).

5.3. Generisanje jakih prostih brojeva

Prost broj p je jak prost broj ako postoje cijeli brojevi r, s i t takvi da vrijedi: p-1 ima veliki prost faktor, označen sa r, p+1 ima veliki prost faktor, označen sa s i r-1 ima veliki prost faktor, označen sa t.

5.4. *Maurerov* algoritam za generisanje "sigurno" prostih brojeva

Maurerov algoritam generiše "sigurno" proste brojeve, odnosno brojeve za koje se nekim od stvarnih testova može pokazati da su prosti. Ovaj podskup prostih brojeva je gotovo uniformno raspoređen po skupu svih prostih brojeva. Očekivano vrijeme za izvršavanje ovakvog algoritma je neznatno veće od vremena potrebnog za algoritam 5.1. sa jednim ciklusom u *Miler-Rabinovom* testu.

6. Trenutno stanje u oblasti velikih prostih brojeva

Da bi smo stekli jasniju predstavu o veličini trenutno najvećih poznatih prostih brojeva daćemo podatke objavljene na Web stranici http://primes.utm.edu/largest.html koja bilježi sve pronađene velike proste brojeve od 1994. godine. Podaci su ažurirani 25.10.2009. godine. *Sophie-Germain* prost broj p je onaj za koji je i 2p+1 takođe prost.

Pozicija	Broj	Cifara	Od
1	$2^{43112609}-1$	12.978.189	2008
2	$2^{42643801}-1$	12.837.064	2009
3	$2^{37156667}-1$	11.185.272	2008
4	$2^{32582657}-1$	9.808.358	2006
5	$2^{30402457}-1$	9.152.052	2005
6	$2^{25964951}-1$	7.816.230	2005
7	$2^{24036583}-1$	7.235.733	2004
8	$2^{20996011}-1$	6..320.430	2003
9	$2^{13466917}-1$	4.053.946	2001
10	$19249 \cdot 2^{13018586}+1$	3.918.990	2007

Tabela 3. Najveći prosti brojevi

Pozicija	Broj	Cifara	Od
1	$65516468355 \cdot 2333333+1$	100.355	2009
2	$65516468355 \cdot 2333333-1$	100.355	2009
3	$2003663613 \cdot 2195000+1$	58.711	2007
4	$2003663613 \cdot 2195000-1$	58.711	2007
5	$194772106074315 \cdot 2171960+1$	51.780	2007
6	$194772106074315 \cdot 2171960-1$	51.780	2007
7	$100314512544015 \cdot 2171960+1$	51.780	2006
8	$100314512544015 \cdot 2171960-1$	51.780	2006
9	$16869987339975 \cdot 2171960+1$	51.779	2005
10	$16869987339975 \cdot 2171960-1$	51.779	2005

Tabela 4. Najveći prosti brojevi "blizanaci"

Pozicija	Broj	Cifara	Od
1	$2^{43112609}-1$	12.978.189	2008
2	$2^{42643801}-1$	12.837.064	2009
3	$2^{37156667}-1$	11.185.272	2008
4	$2^{32582657}-1$	9.808.358	2006
5	$2^{30402457}-1$	9.152.052	2005
6	$2^{25964951}-1$	7.816.230	2005
7	$2^{24036583}-1$	7.235.733	2004
8	$2^{20996011}-1$	6.320.430	2003
9	$2^{13466917}-1$	4.053.946	2001
10	$2^{6972593}-1$	2.098.960	1999

Tabela 5. Najveći Marsenovi prosti brojevi

Pozicija	Broj	Cifara	Od
1	$607095 \cdot 2^{176311}-1$	53.081	2008
2	$48047305725 \cdot 2^{172403}-1$	51.910	2009
3	$137211941292195 \cdot 2^{171960}-1$	51.780	2008
4	$33759183 \cdot 2^{123458}-1$	37.173	2006
5	$7068555 \cdot 2^{121301}-1$	36.523	2005
6	$2540041185 \cdot 2^{114729}-1$	34.547	2005
7	$1124044292325 \cdot 2^{107999}-1$	32.523	2004
8	$112886032245 \cdot 2^{108000}-1$	32.523	2003
9	$18912879 \cdot 2^{98395}-1$	29.628	2001
10	$3364553235 \cdot 2^{88888}-1$	26.768	1999

Tabela 6. Najveći Sophie-Germain prosti brojevi

7. Zaključak

Prosti brojevi imaju veoma važnu ulogu u kriptografiji. U ovom radu navedene su metode generisanja prostih brojeva i testovi za ispitivanje da li je broj prost. Analiza je pokazala da su stvarni testovi tačniji, ali manje efikasni u pogledu vremena ispitivanja i upotrebe računskih resursa. S druge strane, probabilistički testovi su znatno brži i lakši za implementaciju, ali daju rezultat sa određenom vjerovatnoćom odnosno sa manjom tačnošću. Teorija, ali i simulacija u softverskom paketu "MATHEMATICA" pokazuju da je Miler-Rabinov test najpouzdaniji i da je među jednostavnijim za realizaciju. Pri određivanju parametara za generisanje ključeva RSA algoritma, prostih brojeva p i q, treba voditi računa o sledećem:

- p i q moraju biti dovoljno veliki kako bi faktorizacija njihovog proizvoda bila računski teško izvodljiva
- p i q bi trebali biti slučajni prosti brojevi takvi da je napad njihovog otkrivanja "grubom silom" neisplativ
- p i q bi trebalo da imaju predefinisanu bitsku dužinu kako bi zadovoljavali specifikaciju algoritma

Otkriće RSA algoritma dovelo je do razmatranja nekoliko dodatnih ograničenja pri izboru p i q koja bi osigurala da rezultujući RSA sistem bude bezbjedan od napada kriptoanalitičara, te je uvedena definicija jakih prostih brojeva. Međutim, vjeruje se da uvođenje jakih prostih brojeva nudi malo više od onoga što nudi izbor slučajnog prostog broja, imajući u vidu bitske dužine RSA ključeva (a samim tim i dužine p i q) koje se danas koriste i u budućnosti preporučuju. S druge strane, upotreba jakih prostih brojeva nije ništa manje bezbjedna i zahtijeva minimalne dodatne računske zahtjeve, tako da su dodatni "računski troškovi" veoma mali.

8. Literatura

1) A.Menzes, P.van Oorschot, S. Vanstone ***Handbook of Applied Cryptography***, CRC Press, 1996.
2) B. Schneier ***Applied Cryptography***, John Wiley & Sons, 1996.
3) R.L.Rivest, A.Shamir, L.Adleman ***A Method for Obtaining digital Signatures and Public-KeyCryptosystems,*** Communications of ACM, 1983
4) U.M. Maurer ***Fast Generation of Prime Numbers and Secure Public-Key Cryptographic Parameters,*** Journal of Cryptology, vol. 8, pp. 123-155, 1995
5) M. Masud, H. Galzie, K.A. Hossain, M.U. Islam ***Aspect of Prime Numbers in Public Key Cryptosystem,*** International Journal of The Computer, the Internet and Management Vol. 13 No.2 (May-August, 2005)
6) M. Joye1, P. Paillier ***Fast Generation of Prime Numbers on Portable Devices: An Update***, Lecture Notes in Computer Sciences, Springer, 2006.
7) E. Kranakis ***Primality and Cryptography,*** John Willey & Sons, 1991

8) C. Blair ***Some Mathematical Techniques in Cryptography and Related Fields,*** University of Illinois, 1991-1993
9) A.O.L. Atkin, F. Morain ***Elliptic Curves and Primality Proving,*** Math. Comp., 1993
10) W. Bosma, M.P. van der Hulst ***Faster primality testing,*** Springer-Verlag, 1998
11) R.G.E Pinch ***Some primality testing algorithms,*** Notices of the AMS, 1993
12) D.M.Bressoud ***Factorisation and primality testing,*** Springer-Verlag, 1989
13) M.O.Rabin ***Probabilistic algorithms for primality testing,*** IJournal of Number Theory, 1980
14) R. Crandall, C. Pomerance ***Prime Numbers - A Computational Perspective,*** Springer-Verlag, 2001
15) P. Ribenboim ***The Little Book of Big Primes,*** Springer Verlag, 1991
16) K. Kedlaya ***Is This Number Prime,*** Berkley Math Circle, 2002-2003
17) M. Joye, P. Paillier, S. Vaudenay ***Effcient generation of prime numbers In Cryptographic Hardware and Embedded Systems - CHES 2000***, vol. 1965 of Lecture Notes in Computer Science, pp. 340{354, Springer-Verlag, 2000
18) C. Lu, A.L.M. Dos Santos, F.R. Pimentel ***Implementation of fast RSA key generation on smart cards***, 17th ACM Symposium on Applied Computing, ACM Press, 2002
19) R.D. Silverman ***Fast generation of random, strong RSA primes,*** Cryptobytes, 1997
20) M. Agrawal, N. Kayal, N. Saxena ***PRIMES is in P,*** Indian Institute of Technology Kanpur, Department of Computer Science & Engineering, 2002
21) M. Agrawal, S. Biswas ***Primality and identity testing via chinese remaindering,*** Proceedings of Annual IEEE Symposium on Foundations of Computer Science, 1999
22) L.M. Adleman, C. Pomerance, R.S. Rumely ***On distinguishing prime numbers from composite numbers,*** Ann. Math., 117:173-206, 1983
23) S. Goldwasser, J. Kilian ***Almost all primes can be quickly certified,*** Proceedings of Annual IEEE Symposium on Foundations of Computer Science, 1986
24) G.L. Miller. ***Riemann's hypothesis and tests for primality,*** Journal of Computer Systems Science, 13:300-317, 1976
25) M.O. Rabin. ***Probabilistic algorithm for testing primality,*** Journal of Number Theory, 12:128-138, 1980
26) R. Solovay, V. Strassen ***A fast Monte-Carlo test for primality*** SIAM Journal on Computing, 6:84-86, 1977
27) D. Kussin, ***Primality Test, Factorisation And Discrete Logarithm***s

Abstract - Privacy of PKI systems is based on the difficult yet solvable or unsolvable mathematical problems. The result is that numbers and their complexity have important role in the maintenance of systems security. For asymmetric algorithms, such as RSA, the significance of particular stands out one class of numbers - prime numbers. If there would be an efficient algorithm for prime numbers factorisation, security asymmetric algorithms would be called into question. In most asymmetric algorithms for key generation we use a prime numbers whose length is one hundred and more digits. The subject of this paper are prime numbers generating methods and primality tests.

Keywords - Primes, generating, testing, cryptography, asymmetric algorithms, RSA, PKI