SJÄLVSTÄNDIGA ARBETEN I MATEMATIK

MATEMATISKA INSTITUTIONEN, STOCKHOLMS UNIVERSITET

Var är nyckeln?
En introduktion till primtalstester inom kryptografi

av

Fredrik Huss

2026 - No L13

MATEMATISKA INSTITUTIONEN, STOCKHOLMS UNIVERSITET, 106 91 STOCKHOLM

Var är nyckeln?
En introduktion till primtalstester inom kryptografi

Fredrik Huss

Självständigt arbete i matematik 15 högskolepoäng, grundnivå

Handledare: Olof Sisask

2026

Abstract

This thesis aims to provide an introduction to primality tests within cryptography through 5 overarching questions. We review the history of cryptography to see the role of mathematics in cryptography, especially in relation to primality tests. We also analyze the role of primality tests in cryptography and examine what primality tests are through a few examples. The main primality tests discussed in this introduction are Miller Rabin´s primality test and the AKS primality test. In summary, the result of this thesis is as follows. The role of primality tests within cryptography is a tool to create keys needed to encrypt messages in asymmetric cryptography. Primality tests work through the trial of an integer n with an aspect unique to primes. Primality tests can be categorized into two groups. One is a probabilistic test, where, through multiple trials with witnesses, one can determine if an integer is a probable prime or not. Examples given are Fermat's primality test and Miller-Rabin's primality test. A deterministic primality test gives a definitive answer to whether an integer n is prime or not. Examples of deterministic primality tests given in this paper are Wilsons primality test and AKS primality test. Finally, we attempt to examine the practicality of primality tests in a cryptographic context through a comparison between the Miller-Rabin test and the AKS-test through the $O-$notation. We see that the AKS test is more complex than the Miller-Rabin test, and thus less practical within a cryptographic context, where speed and ease of use are the key to practicality.

Detta arbete siktar på att ge en introduktion av primtalstester inom kryptografin genom fem frågeställningar. Vi går igenom historien av kryptografin för att hitta betydelsen av matematik, främst primtalstester inom kryptografisk kontext. Vi går också igenom vad primtalstester är genom några exempel, vilket våra huvudexempel är Miller-Rabins primtalstest och AKS-primtalstest. Kortfattat är resultatet som följande. Primtalstesters roll inom kryptografin är som verktyg för skapandet av nycklar till krypteringen av meddelanden i asymmetrisk kryptografi. Primtalstester fungerar genom att pröva ett heltal n med ett koncept unikt för primtal, och kan delas in i två grupper. Den ena är probabilistiskt test där genom flera prövningar av heltalet n med möjliga vittnen, se om n är ett möjligt primtal eller inte. Exempel på probabilistiska primtalstest är Fermats test, och Miller-Rabins primtalstest. Den andra gruppen är deterministiska primtalstest som ger ett definitivt svar om ett heltal n är ett primtal eller ej. Exempel av deterministiska primtalstest givet i arbetet är Wilsons primtalstest, och AKS-primtalstest. Slutligen försöker vi se praktikaliteten av primtalstester ur en kryptografisk kontext genom jämförelse mellan AKS-primtalstest och Miller-Rabins primtalstest med hjälp av O notationen. Här ser vi att AKS-testet är mer komplext än Miller-Rabin och därmed mer opraktiskt ur en kryptografisk kontext där snabbhet och användarvänlighet är nycklarna till praktikaliteten.

Förord

Jag vill främst tacka två personer. Den första är min handledare Olof Sisask för uppsatsidén, och de handledningar som har getts under uppsatsskrivandets gång. Även om min kommunikationsförmåga var urusel var de handledningar som har getts extremt hjälpsamt för mig. Den andra är min studiekamrat Niklas Hellberg, som har hjälpt till med definitioner rörande kroppar, samt har korrekturläst detta arbete.

Innehåll

1 Introduktion

1.1 Inledning

Vad för bild har folk när de hör ordet kryptografi? En misstanke jag har är att alla har en sorts bild av kryptografi. Från fiktionella verk som behandlar spioner, historiska verk om hur de allierade knäckte den ökända apparaten ”Enigma”, eller som en kul sak för scouter att lära sig, tror jag att många ser kryptografi som något som existerar utanför vardagen. För dem som har denna bild av kryptografi blir det nog en överraskning att få veta att kryptografi nu utgör en del av vardagen i form av nätverkssäkerhet, och till och med har en stor koppling till matematiken. Så stor att det utgör ett fält inom matematiken att studera om. Speciellt med upptäckten av asymmetrisk kryptografi har flera olika områden inom matematiken fått en koppling till konsten att kryptera.

En av dessa aspekter är primtalstester, vilket kort sagt är algoritmer för att testa om ett valt tal är ett primtal eller inte. Vad har primtalstester med kryptering att göra? Hur fungerar primtalstester, och ger dessa tester ett definitivt svar om ett slumpmässigt valt tal är primtal? Hur snabba är dessa tester, och hur praktiska är dessa primtalstester? Sådana frågor är något som detta arbete vill svara på.

1.2 Syfte och frågeställningar

Detta arbete syftar till att introducera läsaren till primtalstester inom kryptografi genom att kort utreda några berömda tester, bland annat Miller-Rabin-testet och AKS-primtalstestet, ur en kryptografisk vinkel. För att kunna uppnå detta syfte kommer denna text att svara på följande frågor:

1. Vad har primtalstester för mening inom kryptografin?
2. Hur funkar primtalstester? Är det deterministiskt?
3. Vad är Miller-Rabins primtaltest? Hur fungerar det?
4. Vad är AKS-primtalstest? Hur fungerar testet?
5. Hur praktiskt är AKS-primtalstest jämfört med Miller-Rabins test?

1.3 Metod

För att besvara dessa frågor har en litteraturstudie utförts, där flera verk har lästs och innehållet sammanfattats. Valet av litterära verk som har använts i denna uppsats härstammar främst från *An Introduction to Mathematical Cryptography* skriven av Jeffrey Hoffstein, Jill Pipher and Joseph H. Silverman [8], och de rekommenderade verk som har nämnts i boken. Med andra ord ligger Hoffstein som grund, med andra verk som dels ersätter, och fördjupar det som står hos Hoffstein. Samtidigt har böcker från tidigare matematiska kurser som *Discrete Mathematics* skriven av Norman L. Biggs [1] använts för att supplementera matematiska förklaringar. De litterära verk som har använts finns beskrivna i litteraturlistan vid slutet av verket.

Slutligen är pseudokoderna för algoritmerna som denna uppsats tar upp (förutom Wilsons) baserade på pseudokoder från Dietzfelbinger [6].

2 Kort om kryptografi

Som en fortsättning av introduktionen och som ett svar på vår första fråga vill vi försöka visa kopplingen mellan kryptografi och primtalstester. Något som vid ett ögonblick ser ut att vara två helt separata områden inom matematik. Mer detaljerat kommer vi kort att gå igenom kryptografins historia, från antika försök att gömma meddelanden till den asymmetriska kryptografins upptäckter med Diffie och Hellmans arbete samt RSA-kryptografins födelse. Därefter går vi kort igenom RSA-kryptografin, den kryptografiska metoden som en majoritet av matematikstudenter kommer att känna igen, och den kryptografiska metod som primtalstester får sin mening av. Här genomgås mekanismerna bakom RSA-kryptografin som repetition.

2.1 Vad är kryptografi?

Kryptografi, kort sagt, är konsten att gömma ett meddelandes mening. Med andra ord är det inte att gömma själva meddelandet som är målet, utan att göra meddelandet oläsligt om man inte vet nyckeln till meddelandet. Detta skiljer sig från steganografi,som fokuserar på att gömma meddelandet, medan meddelandet i sig kan läsas av vem som helst som har kunnat hitta ([17] s.9-10, [9] s. xiii).

Utifrån kryptografi, som är konsten att gömma ett meddelandes mening, finns det också kryptoanalys, vilket fokuserar på att läsa ut det gömda meddelandet. Kryptografi och kryptoanalys tillsammans bildar ett större ämnesområde kallat kryptologi.

I vår nutid delas kryptografi in i två stora riktningar: symmetrisk kryptografi och asymmetrisk kryptografi. Symmetrisk kryptografi är den typ av kryptografi där både sändare och mottagare behöver en gemensam nyckel för att kunna enkryptera och dekryptera meddelandet som skickas mellan dem. På detta vis kan man se symmetrisk kryptografi som en metod där båda parter, sändare och mottagare, har en lika stor roll i kommunikationen mellan varandra.

Asymmetrisk kryptografi däremot bygger på att mottagaren har dekrypteringsnyckeln hemligt för sig själv, och har enkrypteringsnyckeln öppen för alla. Här har sändaren en betydligt mindre roll, då hen endast behöver kryptera meddelandet till mottagaren med den krypteringsnyckel som mottagaren har visat till alla. En bra metafor som Hoffstein ([8] s.46) använder för att beskriva asymmetrisk kryptografi är en brevlåda som mottagaren endast har nyckeln till, och som ställs mitt på en väg för alla att lägga in sitt meddelande.

2.2 Historia av kryptografi

Med att vi kommer att utreda detta ur en kryptografisk vinkel, är det nog bäst att visa betydelsen av primtalstest ur ett kryptografiskt perspektiv. Som ett första steg att visa denna röda tråd vill jag kort gå igenom kryptografins 'metodologiska' historia. Med andra ord kommer jag att kort sammanfatta utvecklingen av de kryptografiska metoder som dyker upp under historien, och där den matematiska kopplingen inte direkt syns tills väldigt sent i historien. Med att metoderna är i fokus kommer därmed användningen av kryptografin, vare sig politisk, militär eller religiös, inte att vara fokuset. Det har också väldig lite matematisk koppling, som först kommer att dyka upp med framkomsten av datorer och asymmetrisk kryptografi. Därmed kan denna del hoppas över av de som är intresserad av det matematiska innehållet.

2.2.1 Början av hemlig skrift och kryptografin under antiken

Kryptografin ser ut att sakna en tydlig början ([8] s.34). Dock har människor döljt meddelanden på olika sätt. I en av de grekiska epikerna av Hete nämns det hur ett meddelande om Persiens planer att angripa grekerna gömdes bakom en vaxtavla, samtidigt som i antika Kina rullades meddelanden skrivna på ett silkband ihop till en boll som svaldes ([9] s.73, [17] s.9-10). Dessa exempel är dock inte det vi kallar för kryptografi, utan är exempel på steganografi.

Primitiva metoder av kryptografi har dykt upp i olika civilisationer, men inte överlevt den civilisation de föd-

des ur ([9] s.72). Av de exempel som har överlevt har vi exempelvis ”skytale” från Sparta. Metoden fungerar genom att man binder ett band runt en stav av en viss diameter och skriver texten tvärs över bandet. När bandet tas från staven kommer det att se ut som att bokstäver står slumpmässigt över bandet, och saknar mening så länge bandet inte binds till staven av samma diameter som staven texten skrevs på. Här bygger chiffret på att skapa en kombination som passar vid en viss stavdiameter, är ett exempel av transpositionschiffer där bokstäverna i en mening rearrangeras ([17] s.11).

En annan metod är substitutionschiffer, som fungerar genom att ersätta bokstäver med andra tecken. En av de mest kända exemplaren av substitutionschiffer i antiken är Caesars chiffer som nämns i Gallerkrigen. Denna metod bygger på att bokstäver ersätts av en annan bokstav genom att flytta på bokstäverna i alfabetet några steg. Andra exempel av substitutionschiffer kan hittas i boken *Kãma sutra* där som nummer 45 av de 64 'konst' som en kvinna anses ha nytta av är en metod för chiffer ([9] s.74-75, [17] s.13).

Från dessa exempel går det att se att även om det går att förklara dessa kryptografiska metoder med matematik, bygger dessa metoder mest på primitiva metoder.

2.2.2 Kryptografin från medeltiden till tidigmodern tid, och kryptoanalysens födelse

Kryptografin under medeltida Europa såg ut att befinna sig på en liknande nivå, och för vissa europeiska riken hade den fallit till enklare metoder än tidigare. Medan kryptografin inte ändrades stort i Europa, utvecklades ett nytt fält kallat kryptoanalys på Arabiska halvön ([9]s.93). Kryptoanalys är konsten att kunna dekryptera meddelanden utan kännedomen om nyckeln, vilket var något som inte var utvecklat till detta stadium tidigare. Kryptoanalysen av detta stadium använde sig av frekvensanalys av bokstäverna inom ett språk för att dekryptera meddelanden. Upptäckten av kryptoanalys attribuеras till utvecklingen av den arabiska kulturen i samband med muslimska guldåldern, där möjligheten till vetenskap blev stor, dels genom översättning av antika texter till arabiska ([9] s.96-99 & [17] s.20-24). Som en utsträckning av denna utveckling existerade en rad olika chiffer inom den arabiska världen ([9] s.94-96).

Med kryptoanalysens upptäckt och dess spridning till Europa under senmedeltiden, dök också behovet av en mer komplicerad kryptografi upp. Den monoalfabetiska kryptografin kompletterades i det senmedeltida Europa med ”null” (symboler som saknar mening), kodord som substituerar hela ord med symboler eller andra ord, och homofoner där en bokstav kunde ha flera substitutioner ([17] s. 33-34 & [9] s.106-107). För att kunna organisera dessa koder och chiffer började så kallade nomenklatorer, en lista av chiffer och koder, att användas ([17] s. 35). Denna metod kom att dominera i den europeiska kryptografin i flera århundraden.

Parallellt med användandet av homofoner, kodord och null dök ett annat system upp. Runt 1466-1467 skrev Leon Battista Alberti en essä där han föreslog kryptering med hjälp av två skivor ovanpå varandra. Den ena skivan, som är stationär, skulle beskriva det vanliga alfabetet, och den andra skivan, som kan roteras, skulle beskriva chiffret. Detta i sig skiljer sig inte från enkel monoalfabetisk chiffer sett i exempelvis Caesars chiffer. Den revolutionära idén Alberti föreslog var att rotera på chifferskivan efter ett tag, vilket leder till ett annat chifferalfabet som används för kryptering. Genom detta ändras meningen av det krypterade meddelandet mittvägs, vilket försvårar kryptanalys med frekvensanalys ([9] s.127-130). Detta är början av något som nu kallas för polyalfabetisk chiffer ([8] s.35). Metoden bygger på att man alternerar mellan flera chifferalfabet när man krypterar, vilket försvårar dekryptering med frekvensanalys, då samma bokstav i det krypterade meddelandet kan stå för flera olika bokstäver.

Runt 50 år efter Albertis essä kom en annan aspekt av den polyalfabetiska kryptografin att dyka upp genom verket *Polygraphia* skrivet av Johannes Trithemius. I verket dyker exempelvis en så kallad ”tableau” (eller tablå på svenska) upp, en kvadratisk matris av alfabeter radade efter varandra, där varje rad beskriver ett unikt chifferalfabet. I samma verk förekom även idén av att ändra chifferalfabet efter varje bokstav, jämfört med Albertis metod att ändra alfabet efter tre till fyra ord ([9] s. 135-136). Detta ser ut att komma att bli standarden vid kryptering med polyalfabetisk kryptering.

En annan förbättring kom i form av ”countersign” (eller kontrasignering om vi översätter det till svenska) föreslagen av Giovan Batista Belaso, där en mening bestående av slumpmässigt valda ord skulle fungera som en nyckel till krypteringen. Texten man ville kryptera sattes under meningen så att varje bokstav av texten var bunden till en av varje bokstav i meningen, som nu pekade på vilket chifferalfabet man skulle använda ([9] s.137).

Dessa tre angivna aspekter ser ut att kombineras ihop med Vigenère-chiffret, som kom att ses som omöjligt att dekryptera under sin tid. Detta chiffer beskrivs som ett polyalfabetiskt chiffer som använder sig av en tableau med 26 tecken (så en 26×26 matris) där varje rad är förflyttad med en bokstav, och som använder sig av ett ord eller en fras som nyckel (countersign) ([17] s.54-56). Dock ser detta chiffer vara en förenklad version. Vigenère hade nämligen föreslagit en sorts automatisk nyckel, som utvecklades efter den krypterade texten. Med andra ord behövde mottagaren endast ha en bokstav given som nyckel för att dekryptera första bokstaven, och därefter kunde mottagaren använda den dekrypterade bokstaven som nyckel för att dekryptera nästa bokstav.([9] s.147-148)

Under denna period ser vi hur matematik i form av frekvensanalys användes för att dekryptera krypterade meddelanden, vilket gav kravet på att komplicera kryptografin för att behålla hemligheten. Detta gjordes i form av att lägga till olika system som null och koder, eller mer opopulärt, genom att använda flera bokstavsordningar med ord som nyckel.

2.2.3 Symmetrisk kryptografi under 1800-talet och dess mekanisering

Under resterande tidigmodern tid ser det ut att olika varianter av nämnda metoder av kryptografi dyker upp, speciellt i form av nomenklatorer som var normen. En intressant metod som dyker upp just innan modern tid är en så kallad hjulchiffer skapad av Thomas Jefferson, där man krypterar meddelanden med hjälp av flera hjul på en axel. Varje hjul är delat i 26 segment, där man har bokstäverna i alfabetet skrivna på ett unikt sätt, så att inget hjul har samma permutation. Man krypterar ett meddelande genom att rotera hjulen på axeln så att meddelandet syns på en rad. Vi får förutom raden med meddelandet 25 krypterade rader, med andra ord 25 olika krypterade meddelanden . Genom att skicka en av de krypterade raderna till mottagaren som har en liknande apparat med samma konfiguration kan mottagaren dekryptera meddelandet genom att återskapa den krypterade raden och söka fram det dekrypterade meddelandet från resterande 25 rader. Nyckeln här blir ordningen av de olika hjulen, vilket skapar olika $n!$ olika ordningar för n antal hjul. Exempelvis för Thomas Jeffersons förslag med 36 hjul så skulle det finnas 36! olika nycklar att använda ([9] s.192-195).

Med uppfinningen av långdistanskommunikation, först med telegraf under 1800-talet och senare med radio vid slutet av 1800-talet, fick kryptografin större betydelse än tidigare. Detta var dels på grund av farhågan att vem som helst kunde lyssna på meddelandet, vilket blev verklighet med radion, men främst på grund av kostnaden för att skicka meddelanden, eftersom det kostade för varje bokstav som skickades. Koder som förkortade ord och nomenklatorer såg ut att bli normen ([9] s.189-190).

Dock från ett militärt håll, där slagfältet blev mer komplicerat och kommunikation med telegrafi blev normalt, började kravet på ett system som snabbt kunde distribueras öka. Nomenklatorer som var normen sedan tidigare krävde tid att skapas och distribueras, samt kunde orsaka stora skador för en sida när nomenklatorn fångades av fienden. Därmed började polyalfabetiska chiffer som var opopulära inom officiella kretsar tills dess, att bli mer populär ([9] s.190-191).

Under mitten av 1800-talet dök en lösning till polyalfabetiska kryptografiska metoder som tidigare nämnda Vigenére-chifferet. Metoden för dekrypteringen kom att kallas för Kasinski-testet från en före detta preussisk officer som publicerade metoden. Med detta blev polyalfabetisk chiffer inte en säker krypteringsmetod då lösningen blev öppen för alla, och nya metoder började eftersökas ([17] s.73-74).

Under 1890-talet kom August Kerckhoffs ut med boken *La Cryptographie Militarie*, där de principer som kom att bestämma kryptografi hade angetts. Bland dessa principer är det som nu kallas för Kerckhoffs princip, som innebär att man ska anta att fienden vet metoden som användes för kryptering, därmed är hemligheten av nyckeln det som är viktigt, och inte metoden i sig. Här anges också två metoder för kryptanalys som kom att bli standard ([9] 233-238).

En annan kryptografisk fynd som kan vara matematiskt intressant skedde vid liknande tid. Vid 1888 hade en man med namnet de Viaris kommit på med en maskin som kunde kryptera meddelanden på vis av Vigenére-chiffer, och noterade också hur samma chiffer kunde fungera matematiskt med ekvationen "$c + \gamma = \chi$", där c beskrev okrypterade bokstaven, γ beskrev nyckeln och χ beskrev den krypterade bokstaven. Om summan i ekvationen blev över 26 kunde man subtrahera det med 26 för att få bokstaven, vilket är en sorts moduloräkning ([9] s.240-242). Med andra ord kunde man ersätta en tablå i en Vigenére-chiffer med denna ekvation där en chifferalfabet beskrivs med antal bokstäver som förflyttas istället för att skriva ut hela alfabeten. Här kan man se en början på kryptografins koppling till matematik.

Parallellt med olika metoder för kryptering med chiffer som uppfanns, dök också konceptet av superkryptering ("superencipherment" på engelska) upp hos diplomatiska håll, där koder såg ut att vara normen. Superkryptering innebär krypteringen av kodord, eller kodnummer, och kunde göras på flera sätt, som substitution med mono- eller polyalfabetisk substitution. Det var dock två metoder som användes frekvent. Den ena är transposition, där kodord eller kodnummer transpositionerades. Den andra var substitution som främst för kodnummer, där nyckeln var ett tal som adderades till koden som kallades för "placode", och summan blev den krypterade koden, kallad för "encicode" ([9] s.251-252). Exempelvis kan vi superkryptera en "placode" 1984 genom att addera nyckeltalet 2875, och få vårt "encicode" som är 4859.

Vi har tidigare sett mekaniska apparater för att kryptera meddelanden, från Albertis två skivor sedan medeltiden, Jeffersons hjulchiffer till den apparat som de Viaris skapade, som kunde kryptera med Vigenére-chiffer. Efter första världskriget började komplicerade maskiner att dyka upp för att kryptera meddelanden. Ett berömt exempel på en sådan maskin är den ökända Enigma som kom att bli den främsta metoden för kryptering för tyskarna under andra världskriget. Apparaten i sin enklaste form bestod av tre delar: ett tangentbord för att mata in meddelandet, en "scrambler" bestående av tre avtagbara hjul som utför kryptering i polyalfabetisk chiffer samt en plugtabell, där genom att koppla mellan olika bokstäver utgör en enkel monoalfabetisk substitution, och slutligen en lamptavla som visar det krypterade meddelandet ([17] s.106-117). Nyckeln för Enigma blir därmed startpositionen, en specifik kombination av tre faktorer: ordningen av de tre hjulen, positionen av varje hjul och kabelkombinationer.

Användningen av Enigma och liknande maskiner, såsom den japanska "PURPLE"(en förbättrad kopia av Enigma), och Lorenz-chiffret (en mer komplicerad variant av Enigma)ledde till en utveckling inom kryptografin. Den första programmerbara datorn utvecklades just för att dekryptera meddelanden krypterade med Lorenz-chiffret. Programmerbara datorer efter kriget användes i samma syfte, men började också användas för att kryptera meddelanden. Det finns tre skillnader som kryptering med dessa datorer har jämfört med mekaniska apparater som enigma. Det första är möjligheten att programmera komplexa mekanismer som är fysiskt omöjligt att skapa med mekaniska chiffer. Det andra är hastigheten av kryptering, där datorer är betydligt snabbare än mekaniska chiffer. Den tredje är datorernas språk som används, där datorer har binära tal som dess språk. Därmed behöver bokstäver konverteras till binära koder, där det mest berömda är ASCII-koder. Med att datorer blev tillgängliga för allmänheten under 50-talet blev detta kryptografiska användning mer frekvent ([17] s.180-185).

Det som vi har sett här är hur kryptografin blev mer diversifierad på grund av det krav som uppstod till följd av metoder som underlättar långdistanskommunikation, såsom telegrafer eller radion. Flera metoder för symmetrisk kryptografi dök upp i hopp om att försvåra dekryptering. En sorts teoretisering har

skett inom kryptografin, som med Kerckhoffs' krav på säker kryptering, eller de Viaris påstående om hur Vigenérechiffret kunde kopplas till matematik med ekvationer (och modulär aritmetik). Vi kunde även se mekaniseringen av kryptografi genom apparater som Enigma, till slutligen användningen av datorer för främst dekryptering, men också för kryptering. Dock byggde alla metoder hittills på att båda parter kände till en nyckel för kryptering, vilket blev ett problem när man behövde förmedla nyckeln till mottagaren utan att någon tjuvlyssnare snodde nyckeln mittvägs ([17] s.185-186). Detta kom att ändras under 1970-talet.

2.2.4 Upptäckten av asymmetrisk kryptografi

Asymmetrisk kryptografi sägs ha upptäckts av James Ellie år 1969, men hölls hemligt från allmänheten då han arbetade under GCHQ ([8] s.61). Dock kom upptäckten för allmänheten i form av artikeln *New Directions in Cryptography* av Whitfield Diffie och Martin Hellman år 1973. I denna artikel presenterades två system för att lösa problemet som symmetrisk kryptografi hade: att behöva dela ut nyckeln till mottagaren säkert i en osäker miljö. Den ena idén var kallad för "Public key cryptosystem" av författarna, och den andra "Public key distribution system".

"Public key cryptosystem" bygger på att man har tre mängder, K för nycklar, M för meddelanden och C för krypterade meddelanden. Nyckeln $k \in K$ här kommer att ge upphov till två algoritmer. En algoritm $E_k : M \to C$ som krypterar meddelandet och en algoritm $D_k : C \to M$ som dekrypterar meddelandet ([8] s.46-47). Diffie och Hellman ([7] s.648) anger följande egenskaper om dessa algorimter:

1. För varje $k \in K$, är D_k inversen till E_k.
2. För varje $k \in K$, och $m \in M$, så är E_k enkel att beräkna. För varje $k \in K$, och $c \in C$ så är D_k enkel att beräkna.
3. För nästan alla $k \in K$ så är varje ekvivalent algoritm till D_k svår att beräknas från E_k.
4. För varje $k \in K$, så är det lätt att beräkna E_k och D_k från k.

Processen för detta system är som följande:

1. Bob skapar k och får fram E_k och D_k.
2. Bob publicerar E medan D hålls gömt.
3. Den som vill skicka ett meddelande till Bob krypterar sitt meddelande genom att använda algoritmen E för att kryptera meddelandet.
4. Bob som tar emot det krypterade meddelandet, dekrypterar genom att använda D.

"Public key distribution system", som Diffie & Hellman anger ha tänkts ut av matematikern Ralph Merkle, skiljer sig från "Public key cryptosystem" i sitt mål. Medan "Public key cryptosystem" bygger på att man har ett par nyckelalgoritmer och publicerar krypteringsalgoritmen, syftar "Public key distribution system" på att två personer byter nyckeln till ett symmetrisk kryptografiskt system ute i det osäkra. Det är nu känt som Diffie-Hellman key exchange ([7] s.648-649).

Kort efter publikationen av artikeln upptäcktes flera asymmetriska kryptografiska system som uppfyllde det koncept Diffie och Hellman hade nämnt i sin artikel. Den första är RSA-kryptografi, som presenterades 1978 av tre matematiker ([8] s.64).

Vi har genom denna genomgång av kryptografins historia fått se hur kryptografi har utvecklats. Från enkla substitutioner där kunskap inom linguistik såg ut att vara mer centralt än matematik, till asymmetrisk kryptografi där matematik har blivit det centrala medlet för att kryptera meddelanden. Med detta kan vi nu se en tydlig koppling mellan kryptografi och matematik, vilket leder till vårt huvudämne. Av de sist presenterade kryptografiska metoderna kunde vi se hur olika matematiska teorier har använts för att kryptera, samt dekryptera meddelanden. Av dessa är det kryptografiska metoder som använder sig av primtal, bland annat RSA-kryptografi, som vårt huvudämne, primtalstester, har sin koppling till kryptografi.

2.3 Kort om RSA-kryptografi

Med att vi har täckt kryptografins historia går vi kort igenom RSA-kryptering som metod med hjälp av Hoffstein et al. [8] för att tydligare visa var primtalstester kommer att spela sin roll.

RSA-kryptografins mekanism bygger på modulo-beräkning, och tyder som följande:

1. Ett par primtal p och q väljs.
2. Utifrån dessa primtal genereras $n = pq$ och $(p-1)(q-1)$.
3. Krypteringsexponenten e väljs sådant att följande egenskap uppfylls:

$$\gcd(e, (p-1)(q-1)) = 1.$$

   Notera att valet för e inte är begränsat, förutom att egenskapen ovan ska uppfyllas. Detta innebär att $e = 1$ också är ett möjligt val för e, även om meddelandet inte krypteras då (ty $e^1 = e$). När e är valt, bestäms inversen d enligt följande:

$$d \cdot e \equiv 1 \pmod{(p-1)(q-1)}.$$

4. Med detta är alla komponenter till nycklarna färdiga. Talparet (e, n) blir den offentliga nyckeln som är öppen för allmänheten, medan talparet (d, n) blir den privata nyckeln.
5. För kryptering av ett meddelande konverteras meddelandet till ett heltal m där $1 < m < n$, och därefter utförs följande beräkning:

$$c = m^e \pmod{n}.$$

6. Slutligen, för dekryptering av det krypterade meddelandet c utför vi följande beräkning:

$$m = c^d \pmod{n} = (m^e)^d \pmod{n}.$$

Kort sagt är RSA-kryptografin en kryptografisk metod som bygger på svårigheten att faktorisera ett par primtal. Som man kan se, bygger alla komponenter till nycklarna på primtalen p och q. Med att (e, n) är tillgängligt som offentligt nyckel kan den som vill tjuvkika på meddelandet (oftast kallad för Eve) som skickas från Bob försöka lista ut p eller q genom att faktorisera n med primtalen som Eve kommer på. Det viktigaste här är att det ska vara svårt att hitta p och q från pq. Om p eller q är låga primtal finns risken att det blir lätt att faktorisera p och q, och tiden för faktoriseringen blir därmed kort. Därmed vill Alice som genererar nycklarna helst ha stora primtal som p och q. Sedan kan de stora primtalet ge tillgång för att kryptera stora meddelanden på en gång. Exempelvis för meddelanden som använder sig av ASCII-koder använder ett bokstav åtta bitar (siffror i bas 2) åt gången, så ett meddelande skriven med ASCII-kod på tio bokstäver kommer att kräva ett tal med 80 siffror. Dock kan det vara svårt att se om ett valt tal är ett primtal om man inte har en lista av primtal i förväg. Det är här som primtalstester får sin betydelse.

3 Primtalstesters grund och komplexitetsberäkning

Vi har genom historien av kryptografi, och en kort överblick av RSA-kryptografi, fått se meningen med primtalstester inom kryptografi: att verifiera att talet man väljer är lämpligt att använda som nyckel. Vi vill därmed fokusera på att svara på våra resterande frågor. För att kunna besvara dessa frågor går vi kort igenom primtalstesters grunder, samt hur algoritmers komplexitet beräknas med O-notationen.

3.1 Primtalstesters mekanism

Ett primtalstest är som namnet säger ett test i form av en algoritm för att avgöra om ett valt tal är ett primtal eller ej. Som vi har sett i tidigare sektion, är en essentiell del av RSA-kryptografi att hitta ett par primtal för att vidare utveckla nycklar som ska användas till kryptering samt dekryptering. Samtidigt ska dessa primtal vara stora för att göra det svårt att hitta dessa två primtal.

Principen bakom primtalstest är att se om vi kan hitta något motexempel a som går emot en egenskap A unik för primtal, för något heltal n. Om a visar sig vara ett motexempel, kallas det för ett **vittne**. Om n är ett sammansatt tal bör det finnas ett a som är ett vittne bland lögnarna (icke-vittnen för sammansatta tal). Låt oss se hur ett primtalstest fungerar i praktiken med hjälp av direkträkning.

1) Vi väljer ett heltal n som vi vill se om det är ett primtal eller ej.
2) Vi väljer en egenskap A som är unik endast hos primtal och bygger en algoritm som prövar om A gäller för heltalet n. För att pröva om n har egenskapen A matas ett möjligt motexempel a från ett bestämt intervall som parameter. I detta fall blir egenskapen A att ett primtal endast har sig själv och 1 som delare. Därmed behöver vi visa att $a \nmid n$ för alla primtal a i intervallet $2 \leq a \leq \sqrt{n}$.
3) Vi provar heltalet n genom att sätta in n samt ett valt heltal a från intervallet i algoritmen. I vårt fall är a ett primtal i intervallet $2 \leq a \leq \sqrt{n}$.
4) Om a visar sig vara ett motexempel, det vill säga att n inte uppnår vår valda egenskap med a, är det säkert att n är ett sammansatt tal. I vårt fall blir a ett vittne om a är en delare till n.
5) Om A inte motbevisas med det valda a och n prövar vi igen med ett annat a i samma intervall tills antingen 4). sker, eller vi är nöjda med resultatet. I vårt fall får vi pröva med alla a i intervallet $2 \leq a \leq \sqrt{n}$ för att resultatet ska vara bra nog för att säga att n är ett primtal.

Det skiljer sig mellan olika primtalstester när vi kan nöja oss med resultatet, men i stort sett kan primtalstesterna delas in i två grupper:

1. Deterministiska primtalstester: Primtalstester som prövar alla möjliga vittnen a kallas för deterministiska, och ger ett definit svar om ett valt heltal n är ett primtal eller inte. Vårt exempel med direkträkning kan räknas som ett deterministiskt test, då vi provar alla möjliga a i intervallet $2 \leq a \leq \sqrt{n}$ tills vi är nöjda. Notera att då vi prövar alla möjliga vittnen bevisar deterministiska primtalstester om ett valt heltal n är ett primtal om det för alla möjliga vittnen a visar att bn är ett primtal.
2. Probabilistiska primtalstester: Primtalstester som inte provar alla möjliga vittnen a, utan istället nöjer sig med ett visst antal slumpmässigt valda a, kallas för probabilistiska primtalstester. Anledningen till att det räcker med l omgångar är på grund av att man har matematiskt bevisat att sannolikheten att träffa en lögnare för ett slumpmässigt valt a är liten (mindre än $\frac{1}{2}$). Med det blir sannolikheten att alla k stycken valda a är lögnare är extremt liten (om det är mindre än $\frac{1}{2}$ för en omgång, blir sannolikheten för att detta sker efter l omgångar är 2^{-l}). Enligt Dietzfelbinger [6] sida 8 så räcker det om denna totala risk är justerad till mindre än 10^{-20}.

I detta arbete kommer totalt fyra primtalstester att presenteras, varav två primtalstester utreds i detalj. Det är ett deterministiskt primtalstest i form av AKS-primtalstestet samt ett probabilistiskt test i form av Miller-Rabins primtalstest.

3.2 Hur snabb kan ett primtalstest vara? Kort om O (Big O)-notationen

Med att vi behandlar algoritmer kommer vi att behöva en metod för att mäta praktikaliteten av ett primtalstest. Om vi lånar ord från Biggs [1], så är effektiviteten av en algoritm antalet operationer som behövs av algoritmen för en input av storlek n i sitt värsta fall. För att mäta antalet operationer en algoritm behöver för en input med storlek n används notationer som O-notationen, som möjliggör klassifiering av valda funktioner g i representativa funktioner f utan att gå in i detaljer. Denna del kommer främst användas för att bygga upp vårt svar till vår sista frågeställning, det vill säga om hur praktiskt AKS-primtalstest är jämfört med Miller-Rabins primtalstest.

Låt oss först skriva ner definitionen för O-notationen, vilket är som följer:

Definition 3.2.1 (*Big-O notationen* [1], [6], [18])**.** *Låt g och f vara funktioner $\mathbb{N} \to \mathbb{R}^+$. När vi säger att*

$$g(n) = O(f(n)),$$

eller

$$g(n) \text{ är } O(f(n)),$$

där $O(f(n))$ är mängden av funktioner definierad som

$$O(f(n)) = \{g(n) : \text{Det finns konstanter } c \text{ och } n_0 \text{ sådant att } 0 \leq g(n) \leq c \cdot f(n) \text{ för alla } n > n_0\},$$

betyder det att $g(n)$ tillhör mängden $O(f(n))$, och inte att de är lika med varandra, vilket "=" oftast innebär. Med andra ord betyder det att det finns en positiv konstant k sådan att

$$g(n) \leq k \cdot f(n).$$

Notera att detta ger en övre gräns till funktionen $g(n)$ (d.v.s. en funktion $g(n)$ kan inte vara mer komplicerad än $O(g(n))$).

För lägre gräns används Ω notationen (via $f(n) = \Omega(g(n))$ om och endast om $g(n) = O(f(n))$). När man vill beskriva att både en undre och övre gräns existerar används Θ notationen.

Vi anger följande lemma om kombination av olika mätningar av O-notation:

Lemma 3.2.2 ([6])**.** *Anta två funktioner $g_1(n)$ och $g_2(n)$, där $g_1(n) = O(f_1(n))$ och $g_2(n) = O(f_2(n))$. Följande gäller vid kombination av dessa funktioners mätningar:*

a) Vid addition av funktionerna $g_1(n)$ och $g_2(n)$, blir den totala komplexiteten

$$g_1(n) + g_2(n) = O(\max\{f_1(n), f_2(n)\}).$$

Med andra ord tas den största mätningen av funktionerna när de adderas ihop.

b) Vid multiplikation av funktionerna $g_1(n)$ och $g_2(n)$, blir den totala komplexiteten

$$g_1(n) \cdot g_2(n) = O(f_1(n) \cdot f_2(n)).$$

Med andra ord multipliceras mätningarna om funktionerna multipliceras med varandra.

Vid beräkningar av komplexa algoritmer kan det leda till krångliga notationer med logaritmiska faktorer. För att förenkla används så kallade $\tilde{O}$ notation där dessa logaritmiska faktorer ignoreras. Definitionen för denna notation är som följer:

Definition 3.2.3 ([6],[18])**.** *Anta funktionen $f : \mathbb{N} \to \mathbb{R}^+$, där $\lim_{n\to\infty} f(n) = \infty$. Då är*

$$\tilde{O}(f(n)) = \{g \mid g : \mathbb{N} \to \mathbb{R}^+, \exists\ C > 0\ \exists\ n_o\ \exists\ k\ \forall n \geq n_0 : g(n) \leq C \cdot f(n)\log(f(n))^k\}.$$

Med andra ord finns det en asymptotisk övre gräns för en funktion $g(n)$ där logaritmiska faktorer ignoreras om det finns positiva konstanter C, k och n_0 sådant att $0 \leq g(n) \leq C \cdot f(n)\log(f(n))^k$ för alla $n > n_0$.

Det finns två sätt att mäta komplexiteten hos en algoritm. Den ena är att räkna antalet enkla aritmetiska operationer som en algoritm har. Med aritmetiska operationer menas det här operationer som addition, multiplikation eller division med rester. Genom att anta att varje sådan operation tar en viss tid, och räkna antalet sådana operationer en algoritm har, får vi en uppskattning som kan representeras med en O notation. Beräkningen av aritmetiska operationer kan dock vara tidskrävande med stora input, som de primtal som oftast används för att skapa nyckel i RSA-kryptering. Som alternativ kan man räkna antalet så kallade *bit-operationer* som en algoritm kräver för en input n. Dessa två sätt att mäta är inte proportionella och kan ge olika värden för samma algoritm ([18] s. 927 & [14] s. 9). Från de verk vi har observerat ser det ut att bit-komplexiteten är mer frekvent använd för att mäta en algoritms komplexitet. I sida 9 i [14] noterar de att "running time" används i samband med bit-komplexitet. Vi kommer därmed att använda antalet bit-operationer som en algoritm kräver i vårt korta svar på frågeställning 5. ([18] s. 927 & [14] s. 9)

Innan vi fortsätter vidare till nästa sektion vill vi ta upp några operationer för att kunna räkna ut O notationen, tagen från sidorna 17-20 i Dietzfelbinger [6]. När vi räknar med bit-operationer definieras den binära representationen av ett naturligt tal n som

$$\| n \| = \lceil \log_2(n+1) \rceil,$$

och för 0 gäller

$$\| 0 \| = 1.$$

Med detta kan bit-komplexiteten beräknas på följande sätt:

Lemma 3.2.4. *Låt n och m vara två naturliga tal.*

a) Vid addition eller subtraktion av n och m tar det

$$O(\| n \| + \| m \|) = O(\log_2 n + \log_2 m)$$

bitoperationer.

b) Vid multiplikation av n och m tar det

$$O(\| n \| \cdot \| m \|) = O(\log_2 n \cdot \log_2 m)$$

bitoperationer.

c) Att beräkna resten av n efter division med m, det vill säga $n \mod m$ tar det

$$O((\| n \| - \| m \| + 1) \cdot \| m \|) = O((\log_2 n - \log_2 m + 1) \cdot \log_2 m) = O(\log(\frac{n}{m}) + 1) \cdot \log m)$$

bitoperationer.

Notera att dessa är de fyra matematiska operationerna översatta till binär representation enligt ovan, och därför hoppar vi över beviset för detta lemma (se [6] sida 18). Slutligen vill vi gå igenom följande fakta tagna från [6] sida 20 och 21. Analysen av algoritmen som leder till dessa lemmor (och därmed fungerar som bevis) refererar jag till samma sida. Observera att två olika värden för bit-komplexitet anges beroende på vilken metod man använder för att räkna ut, nämligen "naiv" metod och "avancerad" metod. Naiv metod innebär oftast det som först prövas, där en snabbare metod (avancerad metod) för att beräkna komplexiteten finns. Detta innebär att en naiv metod inte alltid är enkel, och metoder som använder sig av "repeated squaring" kan anses vara naiva om en snabbare metod finns.

Lemma 3.2.5. *Antalet aritmetiska operationer det tar för att räkna ut $a^n \mod m$ tar $O(\log n)$ multiplikationer och divisioner. Detta leder till en bit-komplexitet av $O(\log_2 n \log_2^2 m)$ med naiva metoder, samt $\tilde{O}(\log n \log m)$ med avancerade metoder.*

Lemma 3.2.6. *Antalet multiplikationer det tar för att testa om ett heltal n är en potens i formen $n = a^b$ är inte större än $O(\log_2^2 n \log\log n)$. Detta innebär en bit-komplexitet på $O(\log_2^4 n \log\log n)$ med naiv metod (notera att "repeated squaring" används här), och $\tilde{O}(\log_2^3 n)$ med avancerade metoder (genom fast multiplication"enligt [6] sida 21).*

Slutligen har vi följande lemma som kommer till stor användning.

Lemma 3.2.7. *Följande gäller angående komplexiteten för Euklides algoritm för två heltal n och m:*

a) Aritmetiska komplexiteten är som mest $2\min\{\| n \|, \| m \|\} = O(\min\{\log_2 n, \log_2 m\}$.

b) Bit-komplexiteten är $O(\log(n)\log(m))$

Bevis. Vi bevisar för varje del.

a) Observera att för varje division vi utför i Euklides algoritm minskar talen strikt. Med andra ord, för exempelvis två omgångar av division i Euklides algoritm, kommer talen i den andra divisionen vara strikt mindre än i den första divisionen. Som exempel, anta a_i, b_i där $a_i > b_i$. Vi får att

$$a_i = c_1 \cdot b_i + (a_i \mod b_i)$$

i vår första division. Anta att $b_{i+1} = a_i \mod b_i$ och $a_{i+1} = b_i$. Vår andra division blir då

$$a_{i+1} = c_2 \cdot b_{i+1} + (a_{i+1} \mod b_{i+1}).$$

Observera att $a_i > a_{i+1}$ och $, b_i > b_{i+1}$, och därmed minskas talen strikt för varje omgång av Euklides algoritm. Vi vill nu visa att talen minskar snabbt, genom att observera resten b_{i+2}. Anta talensekvensen b_i, b_{i+1}, b_{i+2}. Vid en omgång av algoritmen ser vi att två fall finns för b_{i+1}:

Fall 1 $b_{i+1} > \frac{1}{2}b_i = \frac{1}{2}a_{i+1}$.

I detta fall kommer $b_{i+2} = a_{i+1} - b_{i+1} < \frac{1}{2}b_i$.

Fall 2 $b_{i+1} \leq \frac{1}{2}b_i = \frac{1}{2}a_{i+1}$

I detta fall kommer $b_{i+2} = a_{i+1} \mod b_{i+1} < b_{i+1} \leq \frac{1}{2}b_i$.

Observera att i båda fall kommer storleken av b_{i+2} att vara strikt mindre än $\frac{1}{2}b_i$. Med andra ord halveras resten efter två omgångar av Euklides algoritm. Vi kan därmed säga att efter $2\min\{\| n \| , \| m \|\}$ divisioner stannar algoritmen då vi får $(\gcd(n,m), 0)$ och vi inte kan utföra någon division mer. Med att vi utför $2\min\{\| n \|, \| m \|\}$ divisioner blir då komplexiteten $2\min\{\| n \|, \| m \|\} = O(\min\{\log(n), \log(m)\})$.

b) Med en naiv metod utför vi divisioner i denna algoritm. Från lemma 3.2.4 vet vi att bit-komplexiteten för division är $O((\| n \| - \| m \| +1)\cdot \| m \|)$. Om vi använder samma exempel som ovan, notera att $b_i = a_{i+1}$ för $0 \leq i < t$ där t är vår sista division i algoritmen. En division i Euklides algoritm får bit-komplexiteten

$$(\| a_i \| - \| b_i \| +1) \| b_i \| = (\| a_i \| - \| a_{i+1} \| +1) \| b_i \| = \| a_i \| \cdot \| b_i \| - \| a_{i+1} \| \cdot \| b_i \| + \| b_i \|$$

$$\leq (\| a_i \| - \| a_{i+1} \|)\cdot \| b_0 \| + \| b_i \| .$$

För det totala antalet divisioner som utförs får vi att

$$\sum_{0\leq i<t} O((\| a_i \| - \| b_i \| +1) \| b_i \|) = O(\sum_{0\leq i<t} ((\| a_i \| - \| a_{i+1} \|)\cdot \| b_0 \| + \| b_i \|) = O(\| a_0 \| \cdot \| b_0 \| + t \| b_0 \|)$$

$$= O((\| a_0 \| \cdot \| b_0 \|).$$

Från exemplet ser vi att för n, m blir bit-komplexiteten $O(\| n \| \cdot \| m \|) = O(\log_2 n \log_2 m)$.

□

Komplexiteten för lemma 3.2.7 gäller för både ”Euclidean algorithm” och ”Extended Euclidean algorithm” ([6] s. 32).

4 Grundläggande primtalstester och teorier

Vi vill nu kort gå igenom enkla, men viktiga primtalstester innan vi går in på de mer praktiska testerna. Dessa tester är kort sagt direkta tillämpningar av matematiska satser och ger därmed en enkel bild av hur de olika typerna av primtalstester fungerar. Samtidigt bygger vissa primtalstester på de teorier som tas upp, som till exempel Fermats lilla sats.

4.1 Några teorier innan primtalstester

I de efterföljande sektionerna kommer vi att använda oss av en rad definitioner och satser från talteori, samt om ändliga kroppar. Denna sektion är till för att samla ihop dessa teorier här för att hålla flödet i efterföljande sektioner.

4.1.1 Grupper

Definition 4.1.1 ([6]). *En grupp är en mängd G bunden med en binär operation $\circ$ på G med följande egenskaper:*

1) (Associativitet): För tre element $a, b, c \in G$ så gäller det att:

$$(a \circ b) \circ c = a \circ (b \circ c).$$

2) (Identitetselement): Ett element $e \in G$ existerar sådant att

$$a \circ e = e \circ a = a \text{ för alla element } a \in G.$$

3) (inverselement): För alla $a \in G$ existerar ett inverselement $b \in G$ sådant att

$$a \circ b = b \circ a = e.$$

Det visar sig att för varje element a är elementet b unikt, och betecknas a^{-1}.

Definition 4.1.2 (Ordningen av en grupp [15]). *Om en grupp G har n element, kallas G som **en grupp av ordningen** n och noteras som $\mid G \mid$.*

Definition 4.1.3 (Ordningen av ett element n i en grupp [1]). *Låt n vara ett element i en ändlig grupp G. Det minsta positiva heltalet x sådant att*

$$n^x = 1 \ i \ G$$

*kallas för **ordningen av** n **i** G.*

Definition 4.1.4 ([6], [15]). *En delmängd $H \in G$ är en delgrupp till G om följande krav uppfylls:*

1) identitetselementet till G existerar i H, det vill säga

$$e \in H,$$

2) för alla två element $h_1, h_2 \in H$ gäller

$$h_1 h_2 \in H,$$

(notera att vi i ibland inte skriver $\circ$ mellan h_1 och h_2)

3) för alla element $h \in H$ existerar en invers $h^{-1} \in H$.

För ändliga grupper G gäller följande lemma:

Lemma 4.1.5 ([15]). *En delmängd $H \in G$ är en delgrupp till en ändlig grupp G om och endast om:*

1) Identitetselementet e ingår i H.

2) H är sluten inom gruppoperationen för G. Med andra ord, för två element $a, b \in H$ så ska

$$ab \in H$$

gälla.

Beviset bygger på det som anges i [6] samt [15].

Bevis. Beviset bygger på att vi kontrollerar om krav 3) i definition 4.1.4 uppfylls. Vi noterar följande för elementet $a \in H$ (vars invers vi vill visa finns):

$$f_a : H \to H, b \mapsto a \circ b$$

baserat på krav 2) från definition 4.1.4. Vi vill nu visa att f_a är en bijektion. Vi kan se att om

$$f_a(b_1) = f_a(b_2)$$

gäller

$$a \circ b_1 = a \circ b_2.$$

Per associativitetskrav för grupp får vi att

$$b_1 = (a^{-1} \circ a) \circ b_1 = (a^{-1} \circ a) \circ b_2 = b_2,$$

och därmed är f_a injektiv. Då $f_a : H \to H$ är injektiv innebär det att f_a utför en injektion från en ändlig mängd H till en mängd av samma storlek, vilket per postfacksprincipen innebär att f_a också är surjektiv. Då f_a är både injektiv och surjektiv så är f_a bijektiv. Då H är en ändlig mängd, kommer f_a per definition att vara en bijektion mot sig själv.

Notera att per sats är $e \in H$, och då f_a är en bijektion, finns det ett b i H sådant att

$$f_a(b) = a \circ b = e.$$

Notera dock att per definition 4.1.1 så existerar det ett inverselement a^{-1} sådant att $a \circ a^{-1} = e = a^{-1} \circ a$, vilket innebär att

$$a^{-1} \circ (a \circ b) = a^{-1} \circ e \Leftrightarrow (a^{-1} \circ a) \circ b = a^{-1} \circ e \Leftrightarrow b = a^{-1}$$

vilket bevisar existensen av inverselement i H. Därmed uppfylls krav 3 i definition 4.1.4, och i den utsträckning har detta lemma bevisats. □

Definition 4.1.6 (([1],[15])**.** *En sidoklass är en mängd i form av en sammansättning av ett element $g \in G$ och elementerna i en delgrupp H via gruppoperation. Sidoklasser delas in i två olika typer beroende på sammansättningen:*

1. *sidoklasser i formen gH kallas för vänster sidoklass där $gH = \{x \in G : x = g \circ a$ för någon $a \in H\}$*
2. *sidoklasser i formen Hg kallas för höger sidoklass där $Hg = \{x \in G : x = a \circ g$ för någon $a \in H\}$.*

Det finns en ekvivalensrelation $\sim$ mellan sidoklasser enligt följande:

för alla $g_1, g_2 \in G$ så är $g_1 \sim g_2$ om och endast om $g_1H = g_2H$.

Notera att alla element i en sidoklass gH är distinkta. Om $gh_1 = gh_2$ så tyder det per association att:

$$h_1 = (g^{-1}g)h_1 = g^{-1}(gh_1) = g^{-1}(gh_2) = (g^{-1}g)h_2 = h_2.$$

Därmed kommer varje sidoklass till en delgrupp H att ha samma antal element och därmed samma kardinalitet. Med andra ord

$$\mid gH \mid = \mid H \mid .$$

Till denna definition vill vi ha följande sats:

Sats 4.1.7 ([1]). *Låt H vara en delgrupp till G, och $g_1, g_2 \in G$. Två (vänstra) sidoklasser g_1H och g_2H är antingen identiska eller så har de ingen gemensam element.*

Denna sats tyder på att alla sidoklasser till en delgrupp H är disjunkta, och därmed kommer gruppen G kunna beskrivas som en union av delgrupp H och dess (antingen vänstra eller högra) sidoklasser. Vi vill nu bevisa denna sats baserat på beviset från [1] och [15].

Bevis. Låt oss anta ett element x som tillhör två vänstra sidoklasser g_1H och g_2H. Vi vill nu bevisa att dessa sidoklasser är identiska, då de delar elementet x. Vi kan skriva x som följande:

$$x = g_1h_1, h_1 \in H,$$

$$x = g_2h_2, h_2 \in H.$$

Vi vill nu visa att $g_1H \subseteq g_2H$ och vice versa. För $g_1H \subseteq g_2H$ vill vi först anta ett element $y \in g_1H$ sådant att $y = g_1h$ för någon $h \in H$. Vi kan med tidigare notationer uttrycka y som:

$$y = g_1h = (g_1h_1)h_1^{-1}h = xh_1^{-1}h =$$

$$= (g_2h_2)h_1^{-1}h = g_2(h_2h_1^{-1}h).$$

Eftersom H är en delgrupp, så kommer $h_2h_1^{-1}h \in H$ per definition 4.1.4. Per definition 4.1.6 så kommer därmed $y \in g_2H$. Då vi har definierat y som ett godtyckligt element i g_1H, och $y \in g_2H$, så kommer $g_1H \subseteq g_2H$.

Observera att ingen särskillnad finns mellan g_1 och g_2 i antaganden och resonemang. Därmed gäller $g_2H \subseteq g_1H$ enligt symmetri. Då vi fick att g_1H och g_2H är delgrupper till varandra, så betyder det att $g_1H = g_2H$. □

Med dessa definitioner och satser kan vi bevisa Lagranges sats, som kan användas till en del saker, där bevis till Fermats lilla sats är en av dem.

Sats 4.1.8 (Lagranges sats [1]). *Om G är en ändlig grupp, och H är en delgrupp till G, så är ordningen av H (låt oss beteckna detta som m) en delare till ordningen av G (låt oss beteckna detta som n), det vill säga*

$$m \mid n.$$

Beviset tar vi från [6] och [15].

Bevis. Antag att vi har en grupp G, med delgrupp H och dess disjunkta sidoklasser $g_1H, g_2H, ..., g_iH$ där någon av sidoklasserna är H (då om $g \in H$ så är $gH = H$). Notera att vi har nämnt att $g_1H, g_2H, ..., g_iH$ har samma kardinalitet, samt med sats 4.1.7 att G kan skrivas som en union av $g_1H, g_2H, ..., g_iH$. Med andra ord är

$$\mid G \mid = \mid g_1H \mid + \mid g_2H \mid + ... + \mid g_iH \mid .$$

Omvandla ovan och vi får att

$$\mid G \mid = i \mid H \mid .$$

Vi ser därmed att ordningen av G är en produkt av C och ordningen av H, och därmed gäller Lagranges sats. □

4.1.2 Ringar och ändliga kroppar

Definition 4.1.9 ([1]). *En ring är en mängd R med två operationer $+$ och $\cdot$, samt ett element $0 \in R$ som uppfyller följande axiomer:*

1. *R med operation $+$ är en kommutativ grupp, det vill säga uppfyller följande:*

$$(a+b)+c = a+(b+c),$$

$$a+0 = 0+a = a,$$

$$a+(-a) = (-a)+a = 0,$$

och

$$a+b = b+a.$$

2. *R med operation $\cdot$ är sluten, associativ och har identitetselement.*
3. *Den distrubitiva lagen gäller, det vill säga för $a, b, c \in R$ så skall följande gälla:*

$$a \times (b+c) = (a \times b) + (a \times c)$$

och

$$(a+b) \times c = (a \times c) + (b \times c).$$

Definition 4.1.10 ([8]). *En kropp är en kommutativ ring vars alla element utom 0 har en multiplikativ invers. En kropp bestående av ändligt antal element kallas för en ändligt kropp.*

En speciell variant av ändliga kroppar är så kallade splittringskroppar (Splitting fields på engelska).

Definition 4.1.11 (Splittringskropp [13]). *Låt K vara en förlängningskropp (extension field) till $\mathbf{F}$, och $f \in \mathbf{F}[x]$ en funktion. K är en **splittringsfält** för f över $\mathbf{F}$ om f kan skrivas om som en produkt av linjära faktorer vars koefficienter existerar i K. Med andra ord så är K en splittringsfält för f över $\mathbf{F}$ om*

$$f = a(x-\alpha_1)(x-\alpha_2)...(x-\alpha_n)$$

där a är den ledande koefficienten av f, och $\alpha_i \in K$ för $i = 1, 2, ..., n$.

Sats 4.1.12 ([1]). *Om K är en kropp, och $f(x)$ är en polynom av grad $n \geq 1$ i $K[x]$ så kommer ekvationen $f(x) = 0$ ha högst n rötter i K.*

Bevis. Låt oss anta att polynomet $f(x)$ med grad n har m stycken distinkta rötter $a_1, a_2, ..., a_m \in K$. Dessa rötter kommer per faktorsats ge irreducibla delare $x-a_1, x-a_2, ..., x-a_m$ till $f(x)$, vilket leder till följande faktorisering av $f(x)$ i $K[x]$:

$$f(x) = (x-a_1)(x-a_2)...(x-a_m)g(x)$$

där $g(x) \in K[x]$. Ty koefficienterna tillför K, kommer graden av produkterna vara summan av faktorernas grad. Eftersom varje faktor har grad 1 kan vi observera att graden av $f(x)$ är minst m, då vi har m faktorer samt $g(x)$. Notera att graden av $g(x)$ är $n-m$. Då vi inte kan ha ett polynom $g(x)$ med negativ grad per definition för polynom, måste följande gälla för graden av $g(x)$:

$$\deg g(x) = n - m \geq 0.$$

Från observation av ekvationen ser vi att $m \leq n$, vilket betyder att $f(x)$ kan maximalt ha n rötter. □

4.1.3 Talteori

Definition 4.1.13 ([19]). *k är **ordningen av** n **modulo** p om k är den minsta exponenten sådan att*

$$a^k \equiv 1 \pmod{p}.$$

Vi noterar detta som $\mid n \mid_p = k$.

Definition 4.1.14 ([19]). *Låt n vara ett heltal, och p ett primtal sådant att $\gcd(n,p) = 1$. Heltalet n är en **primitiv rot** modulo p om n är en generator till den multiplikativa gruppen $\mathbb{Z}_p^*$. Med andra ord är n en primitiv rot modulo p om*

$$|n|_p = \varphi(p) = p - 1.$$

Lemma 4.1.15. *Låt p vara ett primtal, och anta ett positivt heltal d där $d \mid p-1$. Kongruensen*

$$x^d - 1 \equiv 0 \pmod{p}$$

har då exakt d lösningar modulo p.

Bevis. Vi vill visa att lemmat gäller via sats 4.1.12. Då d är en faktor till $p-1$, låt $p-1 = dk$. Detta innebär att

$$x^{p-1} - 1 = x^{dk} - 1 = (x^d)^k - 1 = (x^d - 1)((x^d)^{k-1} + \dots x^d + 1).$$

Observera att vänsterledet, det vill säga $x^{p-1} - 1$, kommer att ha exakt $p-1$ rötter enligt Fermats lilla sats. Observera att högerledet, det vill säga $(x^d - 1)((x^d)^{k-1} + \dots x^d + 1)$ består av faktorerna $(x^d - 1)$ och $((x^d)^{k-1} + \dots x^d + 1)$. Dessa faktorer kommer att ha högst d rötter respektive $d(k-1)$ rötter enligt sats 4.1.12. Observera dock följande angående rötterna för faktorerna:

$$d + d(k-1) = d(k-1+1) = dk = p - 1.$$

Då vi vet att vår vänstra led har $p-1$ rötter innebär det att faktorerna i högerledet måste ha exakt d respektive $d(k-1)$ rötter för att ekvivalensrelationen ska upphållas. Därmed har

$$x^d - 1 \equiv 0 \pmod{p}$$

exakt d lösningar modulo p. □

Sats 4.1.16. *Låt p vara ett primtal. Då finns det en primitiv rot $n \in \mathbb{Z}_p^*$.*

Vi bevisar denna sats med hjälp av Davenports [5] bevis.

Bevis. 1. Låt p vara ett primtal, samt a och b vara heltal där a har ordningen k och b har ordningen l i $\mathbb{Z}_p^*$. Om k och l är relativt prima har produkten ab ordningen kl, vilket vi bevisar nu. Vi ser att $(ab)^{kl} \equiv 1 \pmod{p}$ då

$$(ab)^{kl} = a^{kl}b^{kl} = (a^k)^l(b^l)^k \equiv 1 \pmod{p}.$$

Med detta ser vi att ordningen av ab är en delare till lk. Vi vill nu visa att ab har ordningen exakt kl. Anta att ab har ordningen $k_1 l_1$ där $k_1 \mid k$ och $l_1 \mid l$. Då gäller

$$a^{k_1 l_1} b^{k_1 l_1} \equiv 1 \pmod{p}.$$

Om vi höjer båda led av kongruensen ovan med l_2 där $l_1 \cdot l_2 = l$. Vi får då att

$$(a^{k_1 l_1} b^{k_1 l_1})^{l_2} \equiv 1^{l_2} \pmod{p} \Leftrightarrow a^{k_1 l} b^{k_1 l} \equiv 1 \pmod{p} \Leftrightarrow a^{k_1 l} \cdot 1 \equiv 1 \pmod{p} \Leftrightarrow a^{k_1 l} \equiv 1 \pmod{p}.$$

Då $b^{k_1 l} \equiv 1 \pmod{p}$ innebär det att $k_1 l$ är en multipel av k. Då k och l är relativt prima innebär det också att k_1 är en multipel av k. Dock, ty vi definierade att $k_1 \mid k$, innebär detta att $k_1 = k$. På liknande sätt får vi att $l_1 = l$, vilket leder till att ab endast har ordningen kl.

2. Låt heltalet $p-1$ primtalsfaktoriseras till primtalspotenser, det vill säga:

$$p-1 = q_1^{a_1} q_2^{a_2} ...$$

där $q_1, q_2, ...$ är primtal. Om vi kan hitta ett heltal x_1 med ordningen $q_1^{a_1}$, x_2 med ordningen $q_2^{a_2}$ och så vidare kan vi med principen från steg 1 av detta bevis skapa en produkt $x_1 x_2 ...$ vars ordning är $p-1$ och därmed är en primitiv rot till p. Vi vill nu visa att om någon q^a är en faktor till $p-1$, så finns det ett heltal vars ordning modulo p är exakt q^a. Heltalet q^a ska därmed leda till att

$$x^{q^a} \equiv 1 \pmod{p} \text{ men inte } x^{q^{a-1}} \equiv 1 \pmod{p}.$$

3. Observera att enligt lemma 4.1.15 kommer $x^d - 1 \equiv 0 \pmod{p}$ ha exakt d lösningar modulo p. Vi ersätter nu d med q^a respektive q^{a-1} och får fram att

$$x^{q^a} \equiv 1 \pmod{p} \Leftrightarrow x^{q^a} - 1 \equiv 0 \pmod{p}$$

har q^a lösningar, och

$$x^{q^{a-1}} \equiv 1 \pmod{p} \Leftrightarrow x^{q^{a-1}} - 1 \equiv 0 \pmod{p}$$

har q^{a-1} lösningar. Antalet heltal q^a sådant att

$$x^{q^a} \equiv 1 \pmod{p} \text{ men inte } x^{q^{a-1}} \equiv 1 \pmod{p}$$

uppnås är därmed $q^a - q^{a-1}$ stycken. Då $q^a > q^{a-1}$ får vi att $q^a - q^{a-1} > 0$, och därmed finns det ett heltal vars ordning modulo p är exakt q^a om p är ett primtal.

□

4.1.4 Kvadratrötter av 1

Detta är en annan egenskap av modulär aritmetik för ett primtal p som kan användas för att se om ett heltal n är sammansatt. Då detta blir centralt för vissa primtalstester, som Miller-Rabins primtalstest vill vi presentera detta separat från andra delar. Låt oss börja med en definition för vad en kvadratrot för 1 är.

Definition 4.1.17 (Kvadratroten av 1 [6])**.** *Låt $1 \leq a < n$ vara ett heltal. Om*

$$a^2 \equiv 1 \pmod{n}$$

gäller så kallas a för en ***kvadratrot av 1*** $\pmod{n}$ *(square root of 1 på engelska).*

För ett heltal n finns det två kvadratrötter av 1 som alltid är triviala modulo n, nämligen 1 och $n-1 \equiv -1 \pmod{n}$. För primtalspotenser p^m där $m \geq 1$ kommer 1 och $p^m - 1$ att vara de endra tiviala kvadratrötter av 1. Vi anger följande sats för detta:

Sats 4.1.18 ([6],[15])**.** *Låt p vara ett udda primtal, och $1 \leq a < p$ ett heltal. Det finns endast två triviala kvadratrötter sådan att $a^2 \equiv 1 \pmod{p^m}$ där $m \geq 1$. Det är nämligen 1 och $p^m - 1$.*

Bevis. Anta att vi har ett heltal $1 \leq a < p$ sådant att

$$a^2 \equiv 1 \pmod{p^m}$$

gäller. Notera att

$$a^2 \equiv 1 \pmod{p^m} \Leftrightarrow a^2 - 1 = (a-1)(a+1) \equiv 0 \pmod{p^m}$$

gäller. Detta betyder att $p^m \mid (a-1)(a+1)$, och i den utsträckning att p^m antingen delar $a-1$ eller $a+1$. Anledningen till att p^m antingen delar $a-1$ eller $a+1$ är för att p som är ett udda primtal inte kan dela både $a-1$ och $a+1$. Därmed måste alla m faktorer av p i p^m finnas hos antingen $a-1$ eller $a+1$.

$$a - 1 \equiv 0 \Leftrightarrow a = 1$$

ger att inom $1 \leq a < 1$ får vi att $a = 1$. För $a - 1$ kan vi räkna så att

$$a + 1 \equiv 0 \Leftrightarrow a \equiv -1 \pmod{p^m}.$$

Då $1 \leq a < 1$ blir det enda tal som ger $-1 \pmod{p^m}$ inom den "ram" som vi har $p^m - 1$. Därmed är 1 och $p^m - 1$ de enda triviala kvadraträtterna som p^m har. □

4.2 Fermats lilla sats, Fermats primtalstest och Fermatvittnen

Fermats lilla sats är en essentiell del för att förstå Miller-Rabins primtalstest, men anses också själv fungera som en slags primtalstest. Därmed vill vi gå igenom det igen, även om majoriteten av läsarna har läst om det i tidigare kurser.

Sats 4.2.1 (Fermats lilla sats [12], [8]). *Låt p vara ett primtal och a ett heltal relativt prima med p. Då gäller*

$$a^{p-1} \pmod{p} \equiv 1.$$

Notera att vi bygger detta bevis på en multiplikativ grupp $\mathbb{Z}_m^*$, där gruppoperationen är multiplikation. Låt oss ange följande definition för underlätthetens skull:

Definition 4.2.2 ([6]). *För ett heltal $m \geq 1$, så definieras en multiplikativ grupp $\mathbb{Z}_m^*$ som följande:*

$$\mathbb{Z}_m^* = \{a \in \mathbb{Z} \mid 1 \leq a < m, \ \gcd(a, m) = 1\}.$$

Ordningen av $\mathbb{Z}_m^*$ beskrivs som följande:

$$\varphi(m) = \mid \mathbb{Z}_m^* \mid,$$

där $\varphi(m)$ kallas för Eulers φ-funktion.

Bevis sats 4.2.1 [8]. Observera att den multiplikativa gruppen $\mathbb{Z}_p^*$, vilket är gruppen som består av alla element relativt prima med p har ordningen $p - 1$ (det vill säga $\mid \mathbb{Z}_p^* \mid = \varphi(p) = p - 1$), då alla element under p förutom 0 är relativt prima. Notera att elementet $a \in \mathbb{Z}_p^*$ med ordningen k (i detta fall minsta exponenten sådan att $a^k = 1$ i $\mathbb{Z}_p^*$) sådant att

$$\{1, a, a^2, ..., a^{k-1}\}$$

skapar en egen delgrupp till $\mathbb{Z}_p^*$. Per sats 4.1.8 kommer alla element i $\mathbb{Z}_p^*$ att ha en ordning som delar $p - 1$. Med andra ord så kan ordningen $p - 1$ beskrivas som

$$p - 1 = nk$$

där k är ordningen till alla element $a \in \mathbb{Z}_p^*$, och n något heltal. Då k är ordningen till $a \in \mathbb{Z}_p^*$ innebär det att $a^k = 1 \pmod{p}$. Med detta får vi att

$$a^{p-1} = a^{nk} = (a^k)^n = 1^n = 1 \pmod{p}$$

vilket var det vi ville bevisa. □

Med hjälp av Fermats lilla sats kan vi undersöka om ett tal är ett sammansatt tal. Detta sker enligt följande: Låt oss ta ett tal n och prova om det är sammansatt genom att genomföra följande beräkning med ett annat heltal a:

$$a^{n-1} \mod n.$$

Om beräkningen ger ett svar annat än $a^{n-1} \equiv 1 \mod n$ kan vi utifrån Fermats sats definitivt säga att n är ett sammansatt tal. Vi kallar då talet a för ett vittne till att n är ett sammansatt tal. Detta leder till följande definition:

Definition 4.2.3 ([6]). *Ett tal a är ett* ***Fermatvittne*** *(även kallad för F-witness på engelska) om*

$$a^{n-1} \not\equiv 1 \pmod{n}.$$

För att förtydliga hur detta fungerar kan vi som exempel utföra detta test på $n = 21$ som vi tydligt ser inte är en primtal. Låt oss använda $a = 2$ som ser ut att vara populärt att använda. Vi får först att

$$2^{21-1} = 2^{20} = (2^5)^4 \equiv 11^4 = (11^2)^2 \equiv (-5)^2 \equiv 4 \pmod{21}.$$

Med att vi har räknat fram att $2^{20} \equiv 4 \pmod{21}$, så blir $a = 2$ ett Fermatvittne för $n = 21$.

Notera dock att vi har vissa sammansatta tal som uppnår kravet för Fermats lilla sats för vissa värden av a. Dessa värden av a kallas då för **Fermatlögnare** (förkortat till F-liars på engelska). Notera att följande gäller främst för udda sammansatta tal, då vi vet att alla jämna tal större än 2 är delbara med 2, och därmed är sammansatt tal utan att testa det. Låt oss visa ett exempel av en Fermatlögnare, nämligen $a = 3$ för det sammansatta talet $n = 121$. Om vi räknar ut $3^{121-1} = 3^{120} \pmod{121}$ får vi att

$$3^{120} = (3^2)^{60} = 9^{60} = (9^2)^{30} = 81^{30} = (81^2)^{15} \equiv 27^{15} \pmod{121} = 27^{14} \cdot 27 = (27^2)^7 \cdot 27 \equiv 3^7 \cdot 27 \pmod{121} = \Leftrightarrow$$

$$\Leftrightarrow = 3^7 \cdot 3^3 = 3^{10} = (3^2)^5 = 9^5 = 9^4 \cdot 9 = 81^2 \cdot 9 \equiv 27 \cdot 9 \pmod{121} = 243 \equiv 1 \pmod{121}.$$

Med detta har vi sett ett exempel på en Fermatlögnare.

Definition 4.2.4 ([14], [6]). *För ett sammansatt tal n kallas a en* ***Fermatlögnare****, eller* ***F-lögnare*** *om*

$$a^{n-1} \equiv 1 \pmod{n}.$$

Om vi har ett heltal a som F-lögnare för n, kommer vårt sammansatta heltal n att kallas för ***pseudoprimtal bas*** *a.*

För udda sammansatta tal n kommer 1 och $n - 1$ per automatik vara Fermatlögnare ([6]). För 1 är det naturligt då $1^{n-1} = 1$, och för $n - 1$ bevisas detta genom följande:

$$(n-1)^{n-1} \equiv (-1)^{n-1} \equiv 1 \pmod{n},$$

då vi har etablerat att n är ett udda sammansatt tal, och därmed blir $n-1$ ett jämnt tal. Låt oss sammanfatta denna del genom att ange följande lemma:

Lemma 4.2.5 ([6], [15]). *Låt $n \geq 2$ vara ett heltal.*

a) Om ett heltal $1 \leq a < n$ har egenskapen $a^r = 1 \pmod{n}$ för någon $r \geq 1$, så är $a \in \mathbb{Z}_n^*$. Det vill säga att a är relativt primt med n.

b) Om $a^{n-1} = 1 \pmod{n}$ gäller för alla a då $1 \leq a < n$, så är n ett primtal.

Bevis. För a) bevisas det som följande:

För $a^r = 1 \pmod{n}$, om $r = 1$ så är $a^r = 1 \Leftrightarrow a = 1$. Om $r \geq 2$ så får vi att

$$a^r = a \cdot a^{r-1} \equiv 1 \pmod{n}.$$

Från kongruensen ovan går det att observera att det finns ett par heltal x och y sådan att

$$ax + ny = 1$$

gäller, där $x = a^{r-1}$ i fallet ovan. Detta leder till att vänsterledet ovan är en multipel av $\gcd(a, n)$ vilket är 1. Därmed är a) bevisat.

För b), om alla $1 \leq a < n$ för något n uppfyller $a^{n-1} \equiv 1 \pmod{n}$, tyder det enligt 4.2.5 a) att $\mathbb{Z}_n^* = \{1, 2, ..., n-1\}$, vilket är alla nollskilda element $\bmod n$. Detta instämmer med definitionen för primtal, och därmed är n ett primtal. □

Med detta lemma kan vi se att om vårt heltal n uppnår b) så kan det kallas för ett primtal. Detta beror på att enligt lemmat kommer det alltid att finnas minst ett Fermatvittne $1 \leq a < n$ för ett udda sammansatt tal n. Med andra ord behöver vi endast se ett Fermatvittne bland alla $1 \leq a < n$ för att bestämma att ett heltal n är ett sammansatt tal. Med detta kan vi ange algoritmen för Fermats primtalstest, vilket är som följer:

Fermats Primtalstest
Input: $a \in \{2, ..., n-2\}$
if $a^{n-1} \neq 1 \pmod{n}$ **then**
| **Output:** Not prime (Fermatvittne)
else
| **Output:** Probable prime (Ickevittne)
end

Vi kan se att algoritmen följer lemma 4.2.5, där varje slumpvist vald a antingen ger att talet är sammansatt med output "Not prime(Fermatvittne)", eller så anges att 4.2.1 gäller för n vid vald a med output "Probable prime(Ickevittne)".

En generell regel med Fermats primtalstest är att risken att en Fermatlögnare blir vald som a i en omgång av algoritmen ovan är generellt mindre än $\frac{1}{2}$. Detta noteras som följande sats:

Sats 4.2.6 ([6],[15])**.** *Om ett $n \geq 3$ är en udda sammansatt tal, sådan att minst ett Fermatvittne för att n är sammansatt existerar i delmängden av den multiplikativa delgruppen $\mathbb{Z}_n^*$, så är sannolikheten att en Fermattest med ett likformigt vald a från $\{2, \ldots, n-2\}$ ger ett rätt svar, det vill säga att n är ett sammansatt tal, är större än $\frac{1}{2}$.*

Bevis. Notera först att en Fermatlögnare enligt definition 4.2.4 är ett tal a för ett sammansatt tal n som uppfyller $a^{n-1} \equiv 1 \pmod{n}$. Per lemma 4.2.5 betyder detta att en Fermatlögnare a tillhör den multiplikativa delgruppen $\mathbb{Z}_n^*$, då $n-1$ är en kandidat för r, och uppfyller $a^r \equiv 1 \pmod{n}$. Med andra ord är mängden med alla Fermatlögnare

$$L_n^F = \{a \mid 1 \leq a < n, a^{n-1} \equiv 1 \pmod{n}\}$$

en delmängd av $\mathbb{Z}_n^*$. Vi vill nu visa att L_n^F är en delgrupp av $\mathbb{Z}_n^*$ enligt Lemma 4.1.5.

För 1) kan vi lätt visa detta, då vi tidigare har etablerat att 1 per automatik är en Fermatlögnare. Vi vet att

$$1^{n-1} \equiv 1 \pmod{n},$$

och därmed är $1 \in L_n^F$.

För 2) vill vi visa att produkten mellan två element $h, i \in L_n^F$ också tillhör L_n^F, det vill säga $hi \in L_n^F$. Med att vi har definierat att $h, i \in L_n^F$ får vi att:

$$(hi)^{n-1} = h^{n-1}i^{n-1} \equiv 1 \pmod{n},$$

och därmed gäller $hi \in L_n^F$. Med detta har vi visat att L_n^F är en delgrupp av $\mathbb{Z}_n^*$. Notera även att om ett Fermatvittne existerar, kommer L_n^F att vara en "äkta delgrupp" $\mathbb{Z}_n^*$, då element som inte existerar i L_n^F existerar i $\mathbb{Z}_n^*$.

Med Lagranges sats (sats 4.1.8) kan vi därmed säga att ordningen för L_n^F delar ordningen för $\mathbb{Z}_n^*$. Då n är ett sammansatt tal per krav på sats, så är

$$| \mathbb{Z}_n^* | < n - 1$$

då ordningen för $\mathbb{Z}_n^*$ när n är ett primtal är $n-1$. Om L_n^F är en äkta delgrupp till $\mathbb{Z}_n^*$, kan vi per Lagranges sats säga att $\mid L_n^F \mid$ är en äkta delare till $\mid \mathbb{Z}_n^* \mid$, där

$$\mid L_n^F \mid \leq \frac{(n-2)}{2}.$$

Med detta blir den största sannolikheten att slumpmässigt valt a är en Fermatlögnare (och därmed ger ett ”fel” svar) utifrån $\{2, ..., n-2\}$ (då vi vet att $a = 1$ och $a = n-1$ är Fermatlögnare per definition) är:

$$\frac{\frac{(n-2)}{2} - 2}{n-3} = \frac{2(\frac{n-2}{2} - 2)}{2(n-3)} = \frac{n-2-4}{2(n-3)} = \frac{n-6}{2(n-3)}.$$

Observera att

$$\frac{n-6}{2(n-3)} < \frac{n-3}{2(n-3)} = \frac{1}{2}$$

vilket är en omformulering av det satsen säger. □

Vi vet nu att en omgång med Fermats primtalstest har mindre än $\frac{1}{2}$ risk att ge ett fel svar. Med andra ord kan en Fermats primtalstest vara riskabelt att nöja sig med endast en omgång. Därmed brukar man repetera primtalstestet en viss antal gånger. Med detta ger Fermats primtalstest följande sannolikheter:

1) n är ett primtal och 0 ges tillbaka som output.

2) n är ett sammansatt tal, där sannolikheten att en likformigt slumpvalt a är en Fermatlögnare är mindre än $\frac{1}{2}$ varje omgång om det finns ett a som är en Fermatvittne i $\{2, ..., n-2\}$.

Om vi gör Fermats primtalstest l omgångar blir sannolikheten att ett ”fel” svar, med andra ord att alla likformigt slumpvalda a i de l omgångarna är Fermatlögnare, är som högst $(\frac{1}{2})^l = 2^{-l}$ ([6] s.76). Till exempel innebär det att för 20 omgångar blir sannolikheten att alla 20 slumpvalda a är Fermatlögnare högst 2^{-20}. Vi har nu gått igenom Fermats lilla sats och dess användning som en probabilistisk primtalstest. Notera dock att testet inte är problemfritt, eftersom det finns vissa sammansatta tal där Fermats primtalstest ger ”fel” svar oavsett a som väljs. Detta kommer att bli huvudämnet för nästkommande delkapitel.

4.2.1 Carmichaeltal

Som vi har nyss sagt finns det vissa sammansatta tal där Fermats primtalstest ger ett ”fel” svar oavsett slumpvist valt a, vilket betyder att alla $a \in \{2, ..., n-2\}$ är Fermatlögnare för detta tal. För sådana tal n ges följande definition:

Definition 4.2.7 ([6], [15])**.** *En udda sammansatt tal n kallas för **Carmichaeltal** om n uppfyller Fermats lilla sats för alla $a \in \mathbb{Z}_n^*$, det vill säga*

$$a^{n-1} \equiv 1 \pmod{n}$$

gäller för alla $a \in \{2, ..., n-2\}$.

Det existerar oändligt många Carmichaeltal ([15] s. 102). Vi kan därmed inte vara säkra på om ett tal n verkligen är ett primtal även om vi får fram att n är primtal genom l omgångar av Fermats test. Kravet för dessa Carmichaeltal har getts ut 1899, 11 år innan första exempel av sådana tal publicerades ([15] s.102). Kriterierna kallade för **Korselts kriterium** är som följande:

Sats 4.2.8 ([15], [2])**.** *Ett heltal $n \geq 2$ är ett Carmichaeltal om och endast om följande kriterium uppfylls:*

1) Talet n är sammansatt, och kan ej delas av kvadrattalet av en primtal p. Med andra ord är det en kvadratfritt tal (kallad för ”squarefree” på engelska).

2) Om ett primtal p är en delare till n, så är $p-1$ en delare till $n-1$.

För att bevisa detta kriterium vill vi ta upp den kinesiska restklassatsen, som blir essentiell för beviset.

Sats 4.2.9 (**Kinesiska restklassatsen** [15], [3]). *För ett heltal $N > 1$, låt $N = n_1...n_r$ där n_i är parvis relativt prima, det vill säga*

$$\gcd(n_i, n_j) = 1$$

för $i \neq j$. För varje $b_1, b_2, ..., b_r \in \mathbb{Z}$ så existerar det en unik $0 \leq x \leq N-1$ för $\pmod N$ sådant att

$$x \equiv b_1 \pmod{n_1}$$

$$...$$

$$x \equiv b_r \pmod{n_r}.$$

Beviset är taget från [3] och [16].

Bevis. För att bevisa kinesiska restsatsen vill vi bevisa dels att en $0 \leq x \leq N-1$ alltid existerar (vi noterar detta som 1)), samt att denna lösning är unikt $\pmod N$ (vi noterar detta som 2)).

1) Låt oss skriva $m_i = \frac{N}{n_i}$ för $i = 1, 2, ..., r$. Notera att ty elementen n_1 är parvis relativt prima och m_i saknar just n_i, så kommer m_i och n_i att vara relativt prima. Observera också att ett tal m_j där $j \neq i$ kommer att vara en multipel till n_i. Med detta kommer det att finnas en invers k_i som är ett heltal $1 \leq k_i < n_i$ sådant att

   $$m_i k_i \equiv 1 \pmod{n_i}.$$

   Låt oss välja $x = b_1 m_1 k_1 + b_2 m_2 k_2 + ...b_r m_r k_r$. Observera att för varje $i = 1, 2, ..., r$ får vi att

   $$b_1 m_1 k_1 + b_2 m_2 k_2 + ...b_r m_r k_r \equiv b_i m_i k_i \pmod{n_i}$$

   då m_j är en multipel av n_i, då $j \neq i$. Då vi har etablerat k_i som invers till m_i, det vill säga $m_i k_i \equiv 1 \pmod{n_i}$, får vi därmed att

   $$b_i m_i k_i \equiv b_i \pmod{n_i}.$$

   Därmed uppfyller x kravet som anges i sats 4.2.9, och vi har visat 1). Vi behöver nu visa att vår lösning x är unik $\pmod N$. Med andra ord vill vi visa att x uppfyller 2).

2) Låt oss anta två lösningar x och y sådan att

   $$x \equiv b_i \pmod{n_i}$$

   respektive

   $$y \equiv b_i \pmod{n_i}$$

   för $i = 1, 2, ..., r$. Notera då att för varje i så är n_i en delare till $x - y$, då $x \equiv y \pmod{n_i}$. Med andra ord gäller det att:

   $$n_1 \mid (x - y),$$
   $$n_2 \mid (x - y),$$
   $$...$$
   $$n_r \mid (x - y).$$

   Då n_i är parvist relativt prima, så kommer produkten $n_1 n_2 .. n_r = N$ att vara en delare till $x - y$. Med andra ord så kommer

   $$N \mid (x - y) \Leftrightarrow x \equiv y \pmod N.$$

   Med detta ser vi att lösningarna x och y skiljer sig med en multipel av N, och därmed är lösningen x unik $\mod N$. Vi har därmed visat 2).

□

Med denna sats kan vi bevisa 4.2.8 baserat på bevisen från [2] och [15].

Bevis. 4.2.8
Antag att vi har en Carmichaeltal n. Vi vill visa att båda kriterium uppnås med vårt tal n.

För krav 1) antar vi först att n är delbart med primtal p, så att

$$n = p^k m,$$

och $\gcd(p, m) = 1$. Notera att det inte finns jämna Carmichaeltal (då det kan delas med 2), så i denna fall ska $p \geq 3$. Vi vill visa att detta kan endast gälla när $k = 1$ med hjälp av motsägelsebevis. Vi antar att $k \geq 2$ sådant att $p^2 \mid n$. Vi kan då med kinesiska restsatsen säga att en a sådan att

$$a \equiv 1 + p \pmod{p^k}$$

samt

$$a \equiv 1 \pmod{m}$$

existerar. Per definition av Carmichaeltal (definition 4.2.7) så gäller

$$a^{n-1} \equiv 1 \pmod{n},$$

och a och n är relativt prima. Notera att ty vi har antagit att $p^2 \mid n$, så betyder det att

$$a^{n-1} \equiv 1 \mod p^2 \Leftrightarrow (1+p)^{n-1} \equiv 1 \pmod{p^2}.$$

Observera att $(1+p)^{n-1}$ kan enligt binomialsatsen skrivas om som

$$(1+p)^{n-1} = \sum_{i=0}^{n-1} \binom{n-1}{i} p^i = 1 + (n-1)p + \sum_{i=2}^{n-1} \binom{n-1}{i} p^i.$$

Observera att för $i \geq 2$ så är alla termer i summan ovan delbart med p^2, och därmed får vi att:

$$(1+p)^{n-1} = 1 + (n-1)p + \sum_{i=2}^{n-1} \binom{n-1}{i} p^i \equiv 1 + (n-1)p \pmod{p^2}.$$

Observera då att vi har tidigare angett att $(1+p)^{n-1} \equiv 1$, och då vi har fått att $1 + (n-1)p \pmod{p^2}$, så betyder det att

$$(1-n)p \equiv 0 \pmod{p^2}.$$

Detta pekar på att $p^2 \mid (n-1)p$ och i utsträckning att $p \mid n-1$. Dock har vi antagit att $p \mid n$, vilket leder till en motsägelse. Därmed så måste $k = 1$, och vi har visat krav 1).

För krav 2) vill vi visa att för varje $p \mid n$ så $(p-1) \mid (n-1)$. Notera att med att vi har visat krav 1), så har vi visat att n består av flera distinkta primtal, och därmed så kommer p och $\frac{n}{p}$ att vara relativt prima. Observera att för en multiplikativ grupp $\mathbb{Z}_p^*$ så kommer gruppen att vara cyklisk då p är en primtal. Låt oss välja en generator g till $\mathbb{Z}_p^*$ med ordningen $p-1$. Enligt kinesiska restklasssatsen har vi en element b sådant att

$$b \equiv g \pmod{p}$$

samt

$$b \equiv 1 \pmod{\frac{n}{p}}.$$

Notera att $\gcd(b, n) = 1$ då p och $\frac{n}{p}$ saknar gemensam faktor. Då vi har antagit att n är en Carmichaeltal gäller följande:

$$b^{n-1} \equiv 1 \pmod{n}.$$

Då $p \mid n$ kan vi reducera kongruensen ovan med p så att

$$b^{n-1} \equiv g^{n-1} \equiv 1 \pmod{p}$$

Då vi har angett att g har ordningen $p-1$ (det vill säga $g^{p-1} \equiv 1 \pmod{p}$), och ovan gäller har vi visat att $n-1$ är en multipel av $p-1$. Därmed är krav 2) bevisad.

Vi har nu bevisat att krav 1) och 2) gäller vid ett hypotetisk Carmichaeltal n. Vi vill nu visa att ett heltal n som uppnår dessa krav är ett Carmichaeltal. Låt oss anta att ett heltal n uppnår krav 1) och krav 2). Vi antar också ett heltal $a \in \{1, 2, ..., n-1\}$ sådan att $\gcd(a, n) = 1$. För varje primtal p som delar n kommer $\gcd(a, p) = 1$, vilket enligt Fermats lilla sats leder till att

$$a^{p-1} \equiv 1 \pmod{p}.$$

Då per krav 2) så ska $(p-1) \mid (n-1)$ om $p \mid n$, vilket tyder på att

$$a^{n-1} \equiv 1 \pmod{p},$$

samt ty enligt krav 1) består n av distinkta primtalsfaktorer kan vi skriva om kongruensen ovan som

$$a^{n-1} \equiv 1 \pmod{n}.$$

Detta stämmer överens med kravet för definition 4.2.7, vilket visar att ett sammansatt tal n som uppnår Korselts kriterium är ett Carmichaeltal. □

En känd egenskap av Carmichaeltal som bygger på denna sats blir nedan:

Lemma 4.2.10. *[6],[15]] Om n är ett Carmichaeltal, så består n av minst tre distinkta primtalsfaktorer*

Bevis. Vi bevisar detta med ett motsägelsebevis. Låt oss anta att ett Carmichaeltal n består av två distinkta primtal p och q sådant att $3 \leq p < q$. Enligt krav 2) av Korselts kriterium (sats 4.2.8 så ska $p-1$ och $q-1$ dela $n-1$. Notera dock att

$$n - 1 = pq - 1 = p(q-1) + p - 1,$$

vilket pekar på att

$$n - 1 \equiv p - 1 \pmod{q-1}.$$

Notera dock att

$$p - 1 \not\equiv 0 \pmod{q-1}$$

då vi har angett att $3 \leq p < q$. Vi får därmed en motsägelse då krav 2) inte uppnås om $n - 1 \equiv p - 1 \pmod{q-1}$. Med detta har vi bevisat denna sats. □

Med detta har vi sett att Fermats sats, även om det är ett användbart test har denna stora nackdel med existensen av Carmichaeltal. Vi kommer att se senare i sektionen Miller-Rabbin test om hur detta motverkas, då Miller-Rabbin bygger på Fermats lilla sats.

4.3 Wilsons sats och Wilsons primtalstest

Med att vi har visat ett exempel på ett probabilistisk primtalstest i form av Fermats primtalstest är det nog värt att visa ett exempel på ett deterministiskt primtalstest. Vi kommer att använda oss av Wilsons sats för att konstruera primtalstestet.

Sats 4.3.1 (Wilsons sats [20])**.** *För ett heltal p, gäller* $(p-1)! \equiv -1 \pmod{p}$ *om och endast om p är ett primtal.*

Bevis. Vi börjar med att visa att $(p-1)! \equiv -1 \pmod{p}$ endast gäller när p är ett primtal.

Vi antar först att p är ett sammansatt tal, vilket betyder att p har en primtalsfaktor q där $2 \leq q \leq p-2$. För en motsägelse antar vi också att $(p-1)! \equiv -1 \pmod{p}$ gäller. Notera att $(p-1)!$ är delbart med q, vilket betyder att

$$(p-1)! \equiv 0 \pmod{q}.$$

Dock, när vi har antagit att $(p-1)! \equiv -1 \pmod{p}$ betyder det också att $(p-1)! \equiv -1 \pmod{q}$ ska gälla. Vi får därmed en motsägelse här, vilket leder till att p måste vara ett primtal.

Vi visar nu att ett primtal p ger att $(p-1)! \equiv -1 \pmod{p}$. Vi antar nu ett primtal $p > 2$, då det är självklart för $p = 2$. Notera att i $\mathbb{Z}_p^*$ kommer alla element $1 \leq a \leq p-1$ att ha en multiplikativ invers b så att $ab \equiv 1 \pmod{p}$ i mängden. Notera att 1 och $p-1$ kommer att ha sig själva som invers, då de är de enda triviala kvadratrötterna modulo p enligt sats 4.1.18. Med detta kan vi säga att i

$$(p-1)! = 1 \cdot 2 \cdot \ldots \cdot (p-1) \pmod{p}$$

så kommer alla element förutom 1 och $p-1$ att kunna paras med dess multiplikativa invers. Vi får därmed att

$$(p-1)! = 1 \cdot 2 \cdot \ldots \cdot (p-1) \equiv 1 \cdot (-1) = -1 \pmod{p}.$$

Vi har nu visat det vi ville visa, och därmed är beviset klar. □

Denna sats utgör grunden till ett deterministiskt primtalstest, som är som nedan.

Wilsons Primtalstest
Input: positivt heltal $n > 1$

if $(n-1)! \equiv -1 \pmod{n}$ **then**
| **Output:** Prime
else
| **Output:** Not prime
end

Detta primtalstest är tyvärr inte användbart då $(n-1)!$ är komplext att beräkna och tar för många operationer för att göra primtalstestet praktiskt.

5 Miller-Rabins primtalstest

I vår tidigare sektion ser vi hur Fermats lilla sats kunde användas som ett probabilistiskt primtalstest. Med tillräckligt antal omgångar kan vi med Fermats primtalstest bestämma med någorlunda stor säkerhet om ett valt heltal n är ett primtal eller inte. Dock ser vi också att detta primtalstest hade en stor nackdel i form av Carmichaeltal, sammansatta tal som gavs ut som primtal enligt Fermats primtalstest. Vi vill nu i denna sektion gå igenom Miller-Rabins primtalstest som bygger på Fermats primtalstest, och är en av de mest populära primtalstesterna inom kryptografisk användning.

Från tidigare sektion introducerades vi till kvadratrötter av 1, där vi med sats 4.1.18 visade att endast 1 och $p^m - 1$ är kvadratrötter till 1 för en primtalspotens p^m där $m \geq 1$. Med denna egenskap kan ett enkelt primtalstest göras, där vi letar efter så kallade icke-triviala kvadratrötter för att bestämma om ett heltal n är ett sammansatt tal eller ej. Låt oss använda $n = 91$ som ett exempel. Om vi provar $a = 27$, får vi det att

$$27^2 = 729 = 91 \cdot 8 + 1 \equiv 1 \pmod{91},$$

och därmed är $a = 27$ en icke-trivial kvadratrot till 1 som tyder på att $n = 91$ är ett sammansatt tal enligt sats 4.1.18. Exempelvis är alla enhetsrötter modulo 91 följande: 1, 27, 64 och 90 ([6] s.78).

Dock är det opraktiskt att söka efter icke-triviala kvadratrötter genom att slumpmässigt välja ett a. Vi kombinerar denna egenskap med Fermats lilla sats för att skapa det som kallas Miller-Rabins primtalstest.

5.1 Grunden av Miller-Rabins primtalstest

Låt oss observera Fermats primtalstest, där för något heltal n och relativt prima heltal a så ska

$$a^{n-1} \pmod{n} \equiv 1$$

gälla för alla $a \in \mathbb{Z}_n^*$. Notera att vi fokuserar på udda heltal $n \geq 3$, då vi vet att alla jämna n är sammansatta tal. Då n är ett udda tal, så är $n - 1$ ett jämnt tal och kan skrivas om enligt följande:

$$n - 1 = u \cdot 2^k,$$

där u är ett udda heltal och $k \geq 1$. Om vi använder detta och skriver om ovan till följande:

$$a^{n-1} \equiv ((a^u) \pmod{n})^{2^k} \pmod{n},$$

kan vi göra följande observation. Om vi har ett primtal p, så måste resterna till a^u och $a^{2^i u}$ (där $0 \leq i \leq k-1$) för en a som inte är delbar med p uppnå något slags krav ([15] s.105). Detta krav beskrivs enligt följande proposition:

Proposition 5.1.1 ([15],)**.** *Låt p vara ett primtal och anta att*

$$p - 1 = 2^k u$$

Där u är ett udda tal och $k \geq 1$. Då måste något av följande krav uppfyllas för varje relativt primt heltal a:
(1) $a^u \equiv 1 \pmod{p}$

eller

(2) $a^{2^i u} \equiv p - 1 \pmod{p}$ *för något* $0 \leq i \leq k - 1$.

Bevis. Notera att per Fermats sats (4.2.1) så ska

$$a^{p-1} \equiv 1 \pmod{p},$$

Vilket per proposition är ekvivalent med

$$a^{2^k u} \equiv 1 \pmod{p}.$$

Modifiera detta och vi får att

$$a^{u \cdot 2^k} - 1 \equiv 0 \pmod{p}.$$

Notera att $a^{u \cdot 2^k} - 1$ är i formen $a^2 - b^2$ då $a^{u \cdot 2^k} - 1 = (a^{2^{k-1} u})^2 - 1$, och konjugatregeln kan användas. Om vi använder konjugatregeln får vi därmed att

$$(a^{u2^{k-1}})^2 - 1 = (a^{u2^{k-1}} - 1)(a^{u2^{k-1}} + 1) = (a^{u2^{k-2}} - 1)(a^{u2^{k-2}} + 1)(a^{u2^{k-1}} + 1).$$

Notera att vi fortfarande kan faktorisera $(a^{u2^{k-2}} - 1)$ med konjugatregeln. Om vi upprepar faktoriseringen får vi slutligen att

$$(a^{u2^{k-2}} - 1)(a^{u2^{k-2}} + 1)(a^{u2^{k-1}} + 1) =$$
$$= (a^{u2^{k-3}} - 1)(a^{u2^{k-3}} + 1)(a^{u2^{k-2}} + 1)(a^{u2^{k-1}} + 1) =$$
$$...$$
$$= (a^u - 1)(a^u + 1)(a^{2u} + 1)(a^{4u} + 1)...(a^{u2^{k-2}} + 1)(a^{u2^{k-1}} + 1).$$

Vi får därmed att

$$(a^u - 1)(a^u + 1)(a^{2u} + 1)(a^{4u} + 1)...(a^{u2^{k-2}} + 1)(a^{u2^{k-1}} + 1) \equiv 0 \pmod{p}.$$

Från observation ser vi att för att denna kongruens ska gälla måste något av följande fall ske:

Fall 1. $a^u \equiv 1 \pmod{p}$.

Fall 2. $a^{2^i u} \equiv -1 \equiv p - 1$ för $0 \leq i \leq k - 1$.

Observera att dessa två fall är exakt de två krav som angavs i propositionen, där fall 1 är krav (1), och fall 2 är krav (2). Därmed är proposition 5.1.1 bevisad. □

Med detta har vi konstruerat kraven för vår test. Från observation kan vi se att om vi har ett heltal a i $1, 2, ..., n - 1$ för ett heltal n sådan att inget av kraven i proposition 5.1.1 uppnås så är n ett sammansatt tal. Vi har därmed konstruerat en enkel primtalstest lik Fermats lilla sats.

Definition 5.1.2 ([4],[15])**.** *Låt $n \geq 3$ vara ett udda heltal, och skriv $n - 1 = 2^k u$ där $k \geq 1$, och u är ett udda heltal. Låt oss välja ett heltal a.*

1). Heltalet a i $\{2, ...n-2\}$ är ett ***Miller-Rabin vittne****, eller* ***MR-vittne*** *om proposition 5.1.1 inte uppnås, det vill säga*

$$a^u \not\equiv 1 \pmod{n},$$

och

$$a^{2^i u} \not\equiv n - 1 \pmod{n}$$

för alla $i \in \{0, 1, 2, ..., k - 1\}$.

2). Om heltalet a i $\{1, 2, ...n - 1\}$ uppnår något av kraven för proposition 5.1.1, det vill säga

$$a^u \equiv 1 \pmod{n},$$

eller

$$a^{2^i u} \equiv n - 1 \pmod{n}$$

för något $i \in \{0, 1, 2, ..., k-1\}$, samt om n är ett sammansatt tal så är a ett ***Miller-Rabin lögnare*** *eller* ***MR-lögnare****. Mängden med alla MR-lögnare L_n^{MR}noteras som följande:*

$$L_n^{MR} = \{a = \{1, 2, ..., n-1\} \mid a^u \equiv 1 \pmod{n} \text{ eller } a^{2^i u} \equiv n-1 \pmod{n} \text{ för något } i \in \{0, 1, 2, ..., k-1\}\}.$$

Om vi kombinerar definition 5.1.2 med Fermats primtalstest, nämligen konceptet att välja a utifrån $\{2, ..., n-2\}$, så har vi konstruerat det som nu är känt som **Miller-Rabins primtalstest**. Algoritmen för detta primtalstest, baserat på sida 81 på [6] och sida 109 på [15], är som följer:

Miller-Rabins primtalstest
Input: Ett heltal $n \geq 3$
Välj slumpvist ett heltal $a \in \{2, ..., n-2\}$
if *(n är jämn) eller (*$\gcd(n, a) \neq 1$*)* **then**
  Output: Not Prime (MR-vittne)
else
  Sök efter u och k sådan att $n - 1 = u \cdot 2^k$.
  $b \leftarrow a^u \pmod{n}$
  if $b \in \{1, n-1\}$ **then**
    Output: Probable Prime (Icke-vittne)
  else
    for $i \in \{1, ..., k-1\}$ **do**
      $b \leftarrow b^2 \pmod{n}$
      if $b = n - 1$ **then**
        Output: Probable Prime (Icke-vittne)
      end
      if $b = 1$ **then**
        Output: Not Prime (MR-vittne)
      end
    end
    Output: Not Prime (MR-vittne)
  end
end

Om vi förklarar algoritmen med ord blir det som följer:

1. Vi väljer ett heltal $n \geq 3$ som vi vill testa, och slumpvist väljer ett heltal $2 \leq a \leq n - 2$. Notera att om n vi har valt är jämnt, eller inte är relativt prima med a så uppfyller det inte kravet för definition 5.1.2, och därmed blir per automatik ett sammansatt tal.

2. Vi subtraherar 1 från n, och delar $n - 1$ på 2 tills vi får ett udda heltal u och 2^k där $k \geq 1$.

3. Vi vill nu testa om proposition 5.1.1 uppnås för vårt heltal n. Vi provar först om krav (1) uppnås, det vill säga om $a^u \equiv 1 \mod n$. Vi sätter en variabel $b = a^u \pmod{n}$ och testar om krav (1) uppfylls. Om detta uppnås kan vi säga att a är en icke-vittne till n.

4. Om krav (1) inte uppfylls provar vi om krav (2) uppnås, det vill säga om $a^{2^i u} \equiv n - 1$ för något $0 \leq i \leq k - 1$. Vi kvadrerar b i gånger och testar om någon av kvadraterna uppfyller krav (2).

   (a) Om någon av kvadraterna är kongruent med $n - 1$ så uppfylls krav (2) och a är en icke-vittne till n.

   (b) Om någon kvadrat b_i är kongruent med 1 så kommer det betyda enligt [15] att alla kvadrater efter b_i (det vill säga $b_{i+1} = b_i^2, b_{i+2} = b_{i+1}^2 ..., b_{k-1} = b_{k-2}^2$) kommer att vara kongruent med 1. Notera också att alla kvadrater innan b_i har gett ett tal utanför $\{1, n-1\}$, det vill säga $b_1 = a^{2u}, ..., b_{i-1} = b_{i-2}^2 \notin \{1, n-1\}$. Därmed uppfylls inte krav två, och a blir ett MR-vittne till n.

5. Om krav (2) inte uppnås av någon av kvadraterna kan vi säga att vårt heltal n inte uppnår proposition 5.1.1, och per definition 5.1.2 är a ett MR-vittne till n:s sammansatthet.

5.2 Komplexiteten av Miller-Rabins primtalstest

Med att algoritmen är presenterad vill vi kort presentera komplexiteten av algoritmen för att kunna svara på vår sista frågeställning. Denna sektion bygger på Dietfelbingers [6] beräkning på sida 81.

Sats 5.2.1. *Komplexiteten för en omgång av Miller-Rabins primtalstest med naiv metod är* $O((\log n)^3)$ *bit-operationer.*

Bevis. Vi går igenom steg för steg.

1. I steg 1 utför vi en direkt kontroll om n är ett jämnt tal eller om $\gcd(n, a) \neq 1$. Det första fallet är enkel division som tar $O((\| n \| - \| 2 \| +1) \cdot \| 2 \|) = O(\log_2(n+1) - \log_2(2+1) + 1) \cdot \log_2(2+1))$ bit-operationer, och andra fallet per lemma 3.2.7 har bit-komplexiteten $O(\log_2(n) \log_2(a))$.
2. I steg 2 söker vi efter u och k. Processen tar maximalt $O(\log n)$ bit-operationer.
3. För beräkningen av $a^u \mod n$ i steg 3 kan vi påstå att en modulär multiplikation $a \cdot a \mod n$ tar $O(\log_2 n \cdot \log_2 n) = O((\log_2 n)^2)$ bit-operationer. För att räkna ut $a^u \mod n$ behöver vi utföra runt $O(\log n)$ modulära multiplikationer med hjälp av "repeated squaring". Detta kommer att ta runt $O(\log_2 n)$ bit-operationer. Totalt tar beräkningen av $a^u \mod n$ $O((\log_2 n)^3)$ bit-operationer.
4. I steg 4 utförs det en modulär multiplikation varje gång vi kvadrerar $b = a^u \mod n$. Vi gör det k gånger (då $0 \leq i \leq k-1$, och vi kvadrerar b i gånger), och då $k \leq \log_2 n$, betyder det att vi utför maximalt $O(\log_2 n)$ modulära multiplikationer. Från lemma 3.2.5 har vi att komplexiteten för $b = a^n \mod m$ är $O(\log_2 n \log_2^2 m)$. Därmed tar varje modulär multiplikation tar $O(\log_2 n \cdot \log_2 n) = O((\log_2 n)^2)$ bit-operationer. Med detta blir den totala komplexiteten för detta steg också $O(\log_2 n \cdot \log_2^2 n) = O((\log_2 n)^3)$ bit-operationer.

Från ovan ser vi att de mest komplexa stegen i algoritmen tar $O((\log_2 n)^3)$ bit-operationer, och därmed är bit-komplexiteten av en omgång av Miller-Rabins primtalstest $O((\log_2 n)^3)$. □

För k omgångar av testet blir komplexiteten $O(k \cdot (\log_2 n)^3)$. Notera att komplexiteten kan minskas med mer avancerade metoder, nämligen $\tilde{O}(k \cdot (\log_2 n)^2)$. För detta referar jag till [6] sida 81.

5.3 Sannolikheten för fel svar i Miller-Rabins primtalstest

Med algoritmen för Miller-Rabins primtalstest kan vi se att om outputten "MR-vittne" ges för något heltal a för ett valt heltal n, är n definivt ett sammansatt tal. Dock finns det också för detta test några sammansatta tal n sådan att primtalstestet ger outputten "Icke-vittne" för ett slumpmässigt valt heltal a. För sådana tal har vi följande definition:

Definition 5.3.1 ([15])**.** *Ett sammansatt tal* n *sådant att* a *inom* $2 \leq a \leq n-2$ *är en MR-lögnare kallas för en* ***stark pseudoprimtal bas*** a*.*

Vi vill nu söka efter sannolikheten att vi träffar dessa MR-lögnare för varje omgång. Från Dietzfelbinger [6] och Gallier [15] noteras det först att denna sannolikhet är mindre än $\frac{1}{2}$, och även mindre än $\frac{1}{4}$. Låt oss med hjälp av Crandal & Pomerance [14] samt Gallier [15] bevisa att sannolikheten för en MR-lögnare vid en omgång av Miller-Rabins primtalstest är mindre än $\frac{1}{4}$. För att bevisa detta vill vi först ange några lemmor och därefter presentera det som översatt kallas för Monier-Rabins sats.

Lemma 5.3.2 ([14])**.** *Låt* n *vara ett udda heltal sådant att* $n - 1 = 2^k u$*, där* u *är ett udda heltal. Låt* $v(n)$ *vara det största heltal sådant att* $2^{v(n)} \mid p-1$ *för varje primtal* p *som delar* n*. Om* n *är en stark pseudoprimtal bas* a *så gäller*

$$a^{2^{v(n)-1}u} \equiv \pm 1 \pmod{n}.$$

Bevis. Låt oss använda kraven från proposition 5.1.1, där vi har två fall:

$$a^u \equiv 1 \pmod{n}, \text{ och } a^{2^i u} \equiv -1 \pmod{n} \text{ för något } i \text{ med } 0 \leq i \leq k-1.$$

Vi vill visa att vårt lemma gäller för dessa två fall. För vårt första fall kan vi se att vårt lemma håller, då för $a^u \equiv 1 \pmod{n}$ medför det att

$$a^{2^{v(n)-1}u} \equiv 1 \pmod{n}$$

då om $v(n) = 1$ där $1 \mid p-1$ för någon p som delar n får vi att

$$2^{v(n)-1}u = 2^{1-1}u = u.$$

Låt oss därmed fokusera på vårt andra fall. Låt oss anta att

$$a^{2^i u} \equiv -1 \pmod{n} \text{ för något } i \text{ med } 0 \leq i \leq k-1,$$

och låt p vara en primtalsfaktor till n. Då kan vi säga att ty ovan gäller även

$$a^{2^i u} \equiv -1 \pmod{p}.$$

Låt oss anta att vi har ett tal g som är ordningen av $a \pmod{p}$, det vill säga g är det minsta heltal som ger $a^g \equiv 1 \pmod{p}$. Notera att $g \nmid 2^i u$ ty ovan, men då

$$(a^{2^i u})^2 = a^{2^{i+1}u} \equiv (-1)^2 = 1 \pmod{p}$$

så är g en delare till $2^{i+1}u$. Då $g \mid 2^{i+1}u$ kan vi säga att den enda exponenten 2 kan ha när vi primtalsfaktoriserar g blir $i+1$, då ingen primtalsfaktor kan vara större än $2^{i+1}u$, samt om i är den största exponenten 2 kan ha, så bör $g \mid 2^i u$ gälla, vilket det inte gäller. Därmed gäller $2^{i+1} \mid g$.

Notera också att ty g är ordningen till $a \pmod{p}$, så kan vi med Lagranges sats (sats 4.1.8) säga att $g \mid p-1$. Ty vi har etablerat att 2^{i+1} är en faktor till g, innebär det att $2^{i+1} \mid p-1$. Ty detta gäller för varje $p \mid n$, så har vi att $i+1 \leq v(n)$. Notera att om $i+1 < v(n)$, så måste vi multiplicera $2^{v(n)-1-i}$ för att konvertera $a^{2^i u}$ till $a^{2^{v(n)-1}u}$. Vi får då att

$$a^{2^{v(n)-1}u} \equiv (-1)^{2^{v(n)-1-i}} \equiv 1 \pmod{n},$$

vilket stämmer med lemmat. Om $i+1 = v(n)$ får vi däremot att

$$a^{2^{v(n)-1}u} = a^{2^{i+1-1}u} = a^{2^i u} \equiv -1 \pmod{n}.$$

Då lemmat håller då $i+1 < v(n)$, har vi visat att lemmat fungerar för fall 2 i proposition 5.1.1. Då lemmat fungerar i båda fallen så är därmed lemma 5.3.2 bevisad. □

Från Crandall & Pomerance [14] sida 139, låt oss definiera mängden $S(n)$ för alla a där n är ett starkt pseudoprimtal enligt följande:

$$S(n) = \{a \mod n \mid a^{2^{v(n)-1}u} \equiv \pm 1 \pmod{n}\}.$$

Vi kan säga följande om mängden $S(n)$.

Proposition 5.3.3 ([15])**.** *För någon sammansatt heltal n gäller följande:*

(1) L_n^{MR} *är inkluderad i* $S(n)$*, eller med andra ord* $L_n^{MR} \subseteq S(n)$*.*

(2) $S(n)$ *är en delrupp till den multiplikativa gruppen* $\mathbb{Z}_n^*$*, det vill säga* $S(n) \subset \mathbb{Z}_n^*$*.*

Bevis. Vi vill visa att båda påståenden stämmer. För påstående (1), låt $a \in L_n^{MR}$. Notera att enligt definition 5.3.1 så är n ett starkt pesudoprimtal bas a om någon $2 \leq a \leq n-2$ är en MR-lögnare. Med andra ord så är n ett starkt pseudoprimtal bas a om $a \in L_n^{MR}$. Notera att per lemma 5.3.2 så gäller

$$a^{2^{v(n)-1}u} \equiv \pm 1 \pmod{n}$$

om n är ett starkt pseudoprimtal bas a. Sammanfattat kan vi säga att ovan gäller om $a \in L_n^{MR}$. Observera att

$$S(n) = \{a \mod n \mid a^{2^{v(n)-1}u} \equiv \pm 1 \pmod{n}\},$$

vilket betyder att om $a \in L_n^{MR}$, så kommer $a \in S(n)$ också gälla. Då alla $a \in L_n^{MR}$ också är $a \in S(n)$ så kan vi därmed säga att $L_n^{MR} \subseteq S(n)$.

För (2), notera att $\mathbb{Z}_n^*$ är en ändlig grupp. Därmed räcker det med att kontrollera om lemma 4.1.5 uppfylls här. Från lemma 4.1.5 har vi följande krav:

1) Identitetselementet e ingår i H
2) H är sluten inom gruppoperationen för G. Med andra ord, för två element $a, b \in H$ gäller

$$ab \in H.$$

För 1) ser vi att identitetselementet $1 \in S(n)$, då

$$1^{2^{v(n)-1}u} \equiv 1 \pmod{n}.$$

För 2), låt oss välja två element $a, b \in S(n)$. Vi vill nu visa att produkten $ab \in S(n)$. Notera att per hur vi har definierat $S(n)$ har vi att

$$a^{2^{v(n)-1}u} \equiv \pm 1 \pmod{n}$$

samt

$$b^{2^{v(n)-1}u} \equiv \pm 1 \pmod{n}.$$

Observera följande fyra möjliga fall för produkten ab.
Fall 1. $a^{2^{v(n)-1}u} \equiv 1 \pmod{n}$ samt $b^{2^{v(n)-1}u} \equiv 1 \pmod{n}$
I detta fall ser vi att

$$(ab)^{2^{v(n)-1}u} \equiv a^{2^{v(n)-1}u} \cdot b^{2^{v(n)-1}u} \equiv 1 \cdot 1 \equiv 1 \pmod{n}.$$

Fall 2. $a^{2^{v(n)-1}u} \equiv -1 \pmod{n}$ samt $b^{2^{v(n)-1}u} \equiv 1 \pmod{n}$
I detta fall ser vi att

$$(ab)^{2^{v(n)-1}u} \equiv a^{2^{v(n)-1}u} \cdot b^{2^{v(n)-1}u} \equiv -1 \cdot 1 \equiv -1 \pmod{n}.$$

Fall 3. $a^{2^{v(n)-1}u} \equiv 1 \pmod{n}$ samt $b^{2^{v(n)-1}u} \equiv -1 \mod n$
I detta fall ser vi att

$$(ab)^{2^{v(n)-1}u} \equiv a^{2^{v(n)-1}u} \cdot b^{2^{v(n)-1}u} \equiv 1 \cdot -1 \equiv -1 \pmod{n}.$$

Fall 4. $a^{2^{v(n)-1}u} \equiv -1 \pmod{n}$ samt $b^{2^{v(n)-1}u} \equiv -1 \mod n$
I detta fall ser vi att

$$(ab)^{2^{v(n)-1}u} \equiv a^{2^{v(n)-1}u} \cdot b^{2^{v(n)-1}u} \equiv -1 \cdot -1 \equiv 1 \pmod{n}.$$

Notera att i alla fyra fallen så ser vi att produkten ab uppnår kraven för $S(n)$, det vill säga

$$(ab)^{2^{v(n)-1}u} \equiv \pm 1 \pmod{n}.$$

Med detta har vi visat att båda kraven för lemma 4.1.5 uppnås av mängden $S(n)$, och därmed är $S(n)$ en delgrupp till $\mathbb{Z}_n^*$.

Med detta har vi visat att båda påståenden i proposition 5.3.3 gäller, och därmed är denna proposition bevisad. □

Vi har nu visat kopplingen mellan L_n^{MR} och $S(n)$. Med följande lemma får vi veta ordningen av $S(n)$.

Lemma 5.3.4 ([14], [15])**.** *Låt $\omega(n)$ vara antalet distinkta primtalsfaktorer av n. Vi har då att*

$$\mid S(n) \mid = 2 \cdot 2^{(v(n)-1)\omega(n)} \prod_{p \mid n} \gcd(u, p-1).$$

Bevis. Vi vill visa att ordningen av $S(n)$ är samma som lemma 5.3.4, det vill säga att antalet lösningar till

$$a^{2^{v(n)-1}u} \equiv \pm 1 \pmod{n}$$

är

$$2 \cdot 2^{(v(n)-1)\omega(n)} \prod_{p \mid n} \gcd(u, p-1).$$

Låt oss först definiera m som $m = 2^{v(n)-1}u$. Låt n primtalsfaktoriseras som följande:

$$n = p_1^{j_1} p_2^{j_2} ... p_k^{j_k}$$

där $k = \omega(n)$, och $j_i \geq 1$ för $i = 1, 2, ..., k$. Observera att ty n är udda, så kommer $p_1^{j_1} p_2^{j_2} ... p_k^{j_k}$ också att vara udda. Med detta kan vi säga att

$$a^m \equiv 1 \pmod{n}$$

gäller om och endast om

$$a^m \equiv 1 \pmod{p_i^{j_i}}, i = 1, 2, ...k.$$

Från den primitiva rotsatsen (sats 4.1.16) kan vi säga att för ett primtal p och heltalet j, så är den multiplikativa gruppen $\mathbb{Z}_{p^j}^*$ cyklisk av ordningen $\varphi(p^j) = p^{j-1}(p-1)$. Med andra ord existerar det en primitiv rot g modulo p^j (detta nämns i [14] sida 139, men mer konkret som sats 4.42. i [15] sida 80). Med detta kan vi säga när $a \pmod{p_i^{j_i}}$ är en lösning till $a^m \equiv 1 \pmod{p_i^{j_i}}$. Låt $a = g^k$ för någon $1 \leq k \leq \varphi(p_i^{j_i})$, så måste vi ha någon

$$g^{km} \equiv 1 \pmod{p_i^{j_i}}$$

där $g \in \mathbb{Z}_m^*$ har ordningen $\varphi(p_i^{j_i})$, så $\varphi(p_i^{j_i})$ måste dela km. Låt $d = \gcd(m, \varphi(p_i^{j_i}))$, samt $m = dm_1$ och $\varphi(p_i^{j_i}) = dn_1$. Då måste $\gcd(m_1, n_1) = 1$ gälla. Samtidigt måste $\varphi(p_i^{j_i})$ vara en delare till km om och endast om $n_1 \mid km_1$. Då vi har etablerat att $\varphi(p_i^{j_i}) \mid km$ gäller detta, samt ty $\gcd(m_1, n_1) = 1$, så måste $n_1 \mid k$. Notera att vi har etablerat att $1 \leq k \leq \varphi(p_i^{j_i})$, vilket leder till att vi har d olika lösningar

$$n_1, 2n_1, ..., (d-1)n_1, dn_1 = \varphi(p_i^{j_i}).$$

Vi har nu visat att

$$a^m \equiv 1 \pmod{p_i^{j_i}}$$

har exakt $\gcd(m, \varphi(p_i^{j_i}))$ lösningar. Då vi har definierat att $m = 2^{v(n)-1}u$ vet vi att $m \mid n-1$, och att $p_i \nmid m$ då $p_i \mid n$. Därmed påverkar $p_i^{j_i}$ inte $\gcd(m, \varphi(p_i^{j_i}))$. Notera också att ordningen $\varphi(p_i^{j_i}) = p_i^{j_i-1}(p-1)$. Slutligen, då per definitionen i lemma 5.3.2 gäller $2^{v(n)} \mid p-1$, och därmed är $2^{v(n)-1}$ en faktor i $p-1$. Med detta kan vi säga att

$$\gcd(m, \varphi(p_i^{j_i})) = \gcd(m, p_i^{j_i-1}(p_i-1)) = \gcd(m, p_i-1) = 2^{v(n)-1} \cdot \gcd(u, p_i-1).$$

Observera att detta gäller för varje p_i. Från kinesiska restklassatsen (sats 4.2.9) kan vi säga att antalet lösningar för

$$a^m \equiv 1 \pmod{n}$$

sådant att ovan gäller för alla p_i, $i = 1, 2, ..., k$ existerar. Antalet bestäms av produkten

$$\prod_{i=1}^{k}(2^{v(n)-1} \cdot \gcd(u, p_i - 1)) = 2^{(v(n)-1)k} \cdot \prod_{1}^{k} \gcd(u, p_i - 1).$$

Notera att $n = p_1^{j_1} p_2^{j_2} ... p_k^{j_k}$ samt att $k = \omega(n)$, vilket leder till att

$$2^{(v(n)-1)k} \cdot \prod_{1}^{k} \gcd(u, p_i - 1) = 2^{(v(n)-1)\omega(n)} \cdot \prod_{p|n}(u, p - 1).$$

Med detta har vi visat att antalet lösningar för

$$a^m \equiv 1 \pmod{n}$$

är

$$2^{(v(n)-1)\omega(n)} \cdot \prod_{p|n}(u, p - 1).$$

Vi vill nu visa att lika stort antal lösningar existerar för

$$a^m \equiv -1 \pmod{n}.$$

Observera att enligt sats 4.1.18 gäller

$$a^m \equiv -1 \pmod{p_i^{j_i}}$$

om och endast om

$$(a^m)^2 = a^{2m} \equiv 1 \pmod{p_i^{j_i}}$$

samt

$$a^m \not\equiv 1 \pmod{p_i^{j_i}}.$$

Låt oss först söka efter antalet lösningar till

$$(a^m)^2 = a^{2m} \equiv 1 \pmod{p_i^{j_i}}$$

vilket blir

$$\gcd(2m, p_i^{j_i - 1}(p - 1)).$$

Observera att $2m = 2(2^{v(n)-1}u)$, och $p_i \nmid 2m$, så

$$\gcd(2m, p_i^{j_i-1}(p_i - 1)) = \gcd(2^{v(n)}u, p_i^{j_i-1}(p_i - 1)) = \gcd(2^{v(n)}u, p_i - 1).$$

Vi har etablerat från definitionen i lemma 5.3.2 att $2^{v(n)} \mid p - 1$, vilket betyder att vi garanterat kommer att ha $2^{v(n)}$ i vår största gemensamma delare. Därmed är

$$\gcd(2^{v(n)}u, p_i - 1) = 2^{v(n)} \cdot \gcd(u, p_i - 1).$$

Notera att vi redan har etablerat att $a^m \equiv 1 \pmod{p_i^{j_i}}$ har $2^{v(n)-1} \cdot \gcd(u, p_i - 1)$ lösningar, vilket betyder att antalet lösningar för $a^m \equiv -1 \pmod{p_i^{j_i}}$ är

$$2^{v(n)} \cdot \gcd(u, p_i - 1) - 2^{v(n)-1} \cdot \gcd(u, p_i - 1) = 2^{v(n)-1} \cdot \gcd(u, p_i - 1).$$

Då vi har samma antal lösningar för $a^m \equiv -1 \pmod{p_i^{j_i}}$ som för $a^m \equiv 1 \pmod{p_i^{j_i}}$ för varje $p_i^{j_i}$ betyder det att antalet lösningar för

$$a^m \equiv -1 \pmod{n}$$

är

$$2^{(v(n)-1)\omega(n)} \cdot \prod_{p|n}(u, p_i - 1).$$

Med detta har vi att antalet lösningar för

$$a^m \equiv \pm 1 \pmod{n},$$

vilket är antalet baser a där talet n är ett starkt pseudoprimtal, är

$$2 \cdot 2^{(v(n)-1)\omega(n)} \prod_{p|n} \gcd(u, p-1)$$

vilket var det vi ville visa. □

Med detta kan vi ta upp vår sista sats för detta kapitel:

Sats 5.3.5 (Monier-Rabins sats [14], [15])**.** *För varje udda sammansatt heltal $n > 9$ så är*

$$\mid L_n^{MR} \mid \leq \mid S(n) \mid \leq \frac{1}{4}\varphi(n)$$

Med andra ord är sannolikheten, att ett likformigt slumpvist valt heltal a från mängden $\{2, ..., n-2\}$ är en MR-lögnare för varje omgång, är högst en fjärdedel.

Bevis. Notera att vi behöver bevisa att

$$\frac{\mid S(n) \mid}{\varphi(n)} \leq \frac{1}{4},$$

vilket också betyder att det räcker att visa att

$$\frac{\varphi(n)}{\mid S(n) \mid} \geq 4.$$

Anta som tidigare att $n = p_1^{j_1} p_2^{j_2} ... p_k^{j_k}$. Vi har också $\varphi(p_i^{j_i}) = p_i^{j_i - 1}(p_i - 1)$. Från detta kan vi säga att

$$\varphi(n) = p_1^{j_1-1}(p_1 - 1)p_2^{j_2-1}(p_2 - 1)...p_k^{j_k-1}(p_k - 1).$$

Från lemma 5.3.4 har vi att

$$2 \cdot 2^{(v(n)-1)\omega(n)} \prod_{p|n} \gcd(u, p-1)$$

vilket leder till att

$$\frac{\varphi(n)}{\mid S(n) \mid} = \frac{1}{2} \prod_{p^k || n} p^{k-1} \frac{p-1}{2^{v(n)-1} \gcd(u, p-1)}.$$

$p^k \mid\mid n$ innebär den exakta potensen av p vid primtalsfaktoriseringen av n ([14] s.140). Notera att vi vet per definition att

$$2^{v(n)} \mid p - 1$$

vilket betyder att $2^{v(n)-1} \mid p - 1$, samt att $\gcd(u, p-1)$ är udda, då u är ett udda heltal. Därmed är varje faktor

$$\frac{p-1}{2^{v(n)-1} \gcd(u, p-1)}$$

ett heltal. Observera att det även är ett jämnt heltal, då åtminstone en jämn faktor existerar i kvoten (då $2^{v(n)} \mid p-1$ och $\gcd(u, p-1)$ är udda). Med att

$$\frac{p-1}{2^{v(n)-1}\gcd(u, p-1)}$$

är ett jämnt heltal så kommer $\frac{\varphi(n)}{|S(n)|}$ att vara ett heltal.

Vi har nu fyra fall som påverkar $\frac{\varphi(n)}{|S(n)|}$.

Fall 1: $\omega(n) \geq 3$.

Med att varje faktor

$$\frac{p-1}{2^{v(n)-1}\gcd(u, p-1)}$$

är ett jämnt heltal kan vi säga att

$$\frac{p-1}{2^{v(n)-1}\gcd(u, p-1)} \geq 2.$$

Om $\omega(n) \geq 3$ betyder det att vi har minst tre jämna faktorer, och den minsta produkten av faktorerna blir 8. Med detta får vi fram att den minsta möjliga kvoten $\frac{\varphi(n)}{|S(n)|}$ vi kan ha är

$$\frac{\varphi(n)}{\mid S(n) \mid} = \frac{1}{2} \cdot 8 \cdot \prod_{p^k || n} p^{k-1} = 4 \cdot \prod_{p^k || n} p^{k-1}.$$

Därmed kan vi säga att i detta fall, är

$$\frac{\varphi(n)}{\mid S(n) \mid} \geq 4.$$

Fall 2: $\omega(n) = 2$ och n är inte ett kvadratfritt tal.

Notera att vi kommer att ha två primtalsfaktorer, där någon av exponenterna $k-1 \geq 1$. Produkten av primtalsfaktorerna p^{k-1} kommer att minst vara 3, och därmed kommer vår minsta möjliga kvot att vara

$$\frac{\varphi(n)}{\mid S(n) \mid} = \frac{1}{2} \cdot 3 \cdot 4 = 6,$$

och därmed kommer

$$\frac{\varphi(n)}{\mid S(n) \mid} \geq 6$$

i detta fall.

Fall 3: $n = pq$.

I detta fall antar vi att $n = pq$, där $p < q$. Om $2^{v(n)+1} \mid q-1$, kan vi säga att $q - 1 = 2^{v(n)+1}t$, där t är udda. Vi får därmed att

$$2^{v(n)-1}\gcd(u, q-1) = 2^{v(n)-1}\gcd(u, 2^{v(n)+1}t) = 2^{v(n)-1}\gcd(u, t).$$

Notera att $2^{v(n)-1}\gcd(u, t) \leq 2^{v(n)-1}u$, och

$$2^{v(n)-1}u = 2^{v(n)+1}u \cdot 2^{-2} = \frac{q-1}{4}.$$

Vi får därmed att

$$\frac{q-1}{2^{v(n)-1}\gcd(u,q-1)} \geq \frac{q-1}{\frac{q-1}{4}} = 4.$$

Med att vår andra faktor med $p-1$ kommer minst vara 2, så kommer

$$\frac{\varphi(n)}{\mid S(n) \mid} \geq 4.$$

Om $2^{v(n)} \mid\mid q-1$, så kan vi skriva detta som $q-1 = 2^{v(n)}t$, där t är ett udda heltal. Observera att

$$n-1 = pq-1 = p(q-1)+p-1$$

vilket betyder att $n-1 \equiv p-1 \mod q-1$, och därmed är $q-1 \nmid n-1$.

Då $q-1 \nmid n-1$ så kommer $t \nmid u$. Vi kan då anta ett udda primtalsfaktor r som har högre potens i t än i u så att t inte delar u. Då r har högre potens i t än u innebär det att u saknar minst en faktor r, vilket leder till att vi får följande för $\gcd(t,u)$:

$$\gcd(t,u) \leq \frac{t}{r} \leq \frac{t}{3}.$$

Med detta får vi att

$$2^{v(n)-1}\gcd(u,q-1) = 2^{v(n)-1}\gcd(u,2^{v(n)}t) = 2^{v(n)-1}\gcd(u,t).$$

Eftersom $\gcd(t,u) \leq \frac{t}{3}$ kan vi säga att

$$2^{v(n)-1}\gcd(u,t) \leq 2^{v(n)-1}\frac{t}{3} = \frac{2^{v(n)}t}{2\cdot 3} = \frac{q-1}{6}.$$

Med detta får vi att

$$\frac{q-1}{2^{v(n)-1}\gcd(u,q-1)} \geq \frac{q-1}{\frac{q-1}{6}} = 6.$$

Då vår andra faktion kommer minst vara 2, så kommer

$$\frac{\varphi(n)}{\mid S(n) \mid} \geq 6,$$

och därmed kommer

$$\frac{\varphi(n)}{\mid S(n) \mid} \geq 4$$

i fall 3.

Fall 4: $n = p^a$, $a \geq 2$.

I vårt sista fall kommer vi ha att

$$n-1 = p^a - 1 = 2^k u$$

där u är udda. Ty $a \geq 2$ kan vi faktorisera $p^a - 1$ som

$$p^a - 1 = (p-1)(p^{a-1} + ... + p + 1).$$

Då $2^{v(n)} \mid p-1$, låt $p-1 = 2^{v(n)}t$, där t är udda. Vi kan från observation säga att $t \mid u$, och därmed

$$\gcd(u,p-1) = \gcd(u,2^{v(n)}t) = \gcd(u,t) = t.$$

Med detta får vi att

$$\frac{p-1}{2^{v(n)-1}\gcd(u,p-1)} = \frac{2^{v(n)}t}{2^{v(n)-1}t} = 2.$$

Därmed blir

$$\frac{\varphi(n)}{\mid S(n) \mid} = \frac{1}{2} \cdot p^{a-1} \cdot 2 = p^{a-1}.$$

Vi får då att

$$\frac{\varphi(n)}{\mid S(n) \mid} \geq 5$$

förutom när $n = p^a = 9$, då $p = 3$ och $a = 2$. Med detta har det visats att $\frac{\varphi(n)}{|S(n)|} \geq 4$ i alla fall förutom $n = 9$, och med andra ord så är sannolikheten att ett slumpvist valt heltal a är en MR-lögnare för varje omgång, högst en fjärdedel.

□

Med detta kan vi säga att sannolikheten att ett heltal $n > 9$ som vi prövar med Miller-Rabins primtalstest ger en MR-lögnare efter l omgångar är högst 4^{-l}. Med detta kan vi vara rätt så säker att ett heltal n är ett primtal med ett betydligt mindre antal prövningar än när vi exempelvis provar att faktorisera n med alla primtal $p \leq \sqrt{n}$, och den besparade tiden är värt risken att n är ett sammansatt tal.

6 AKS-primtalstest

I vår tidigare sektion gick vi kort igenom Miller-Rabins primtalstest, ett probabilistiskt primtalstest som kunde ge ett rimligt resultat om ett valt heltal n var sammansatt eller inte. Dock kan vi inte vara säker om vårt heltal n är ett primtal, även efter flertal prövningar. För att vara säker vill vi använda oss av ett deterministiskt primtalstest. Vi vill nu presentera ett deterministiskt primtalstest som sägs vara det snabbaste deterministiska primtalstestet som är bevisat att ge ett primtal utan att använda sig av obevisade hypoteser som Riemannhypotesen. Notera att det är det snabbaste ”bevisade” testet, och betyder inte att det är det snabbaste deterministiska testet, vilket det finns andra betydligt snabbare exempel på. Dock då den är presenterad i Hoffstein et. al. [8], och är ett teoretiskt signifikant primtalstest då det är bevisad att ge ett deterministiskt resultat har jag valt detta test som en representation av deterministiska primtalstester.

Ett av problemen med deterministiska primtalstester var att antingen var de slöa, eller så använde de sig av obevisade hypoteser. Exempelvis var Miller-Rabins primtalstest först givet som ett deterministiskt primtalstest som använde sig av antagandet att den utökade Riemannhypotesen var sann, innan det modifierades till ett probabilistiskt primtalstest. Agrawal, Kayal & Saxena (Agrawal et. al.) konstruerade ett primtalstest som kom att bli teoretiskt signifikant, då detta test var det första testet som bevisade sig att deterministiskt visa om ett prövat primtal n är ett primtal (det vill säga inte byggde på obevisade hypoteser), samtidigt som testet opererade under polynomisk tid. Detta primtalstest hade uppdaterats flera gånger sedan dess första publikation. Vi kommer i denna uppsats att presentera den uppdaterade versionen noterad som ”V6” av Agrawal et. al., för enkelhetens skull.

6.1 Byggande princip och algoritmen

Testet bygger på följande lemma, som enligt Agrawal, Kayal & Saxena [10] är en generalisation av Fermats lilla sats:

Lemma 6.1.1 ([6], [10])**.** *Låt $n \in \mathbb{Z}$ där $n \geq 2$, och a ett heltal relativt prima med n. Då är följande påståenden ekvivalenta med varandra:*

$$n \text{ är ett primtal} \Leftrightarrow (X+a)^n = X^n + a \mod n.$$

Bevis. Vi baserar detta bevis från Dietzfelbingers [6] bevis, där vi vill visa att ekvivalensrelationen mellan båda påståenden är sant.Med andra ord vill vi visa att båda implikationerna ”⇒” och ”⇐” gäller. Notera att enligt binomialsatsen får vi att

$$(X+a)^n = X^n + \sum_{0<i<n} \binom{n}{i} a^i X^{n-i} + a^n.$$

Fall ”⇒”:
Låt n vara ett primtal. Vi vill nu visa att $(X+a)^n \mod n$ ger utvecklingen vi ser i lemma 6.1.1. Notera att

$$\binom{n}{i} = \frac{n(n-1)...(n-i+1)}{i!},$$

där n delar täljaren men inte nämnaren då $0 < i < n$. Därmed kommer $n \mid \binom{n}{i}$. Notera också att då $\gcd(a,n) = 1$, får vi enligt Fermats lilla sats (sats 4.2.1) att

$$a^n \equiv a \pmod n$$

då $a^n = a^{n-1} \cdot a$.Med detta får vi därmed att

$$(X+a)^n = X^n + \sum_{0<i<n} \binom{n}{i} a^i X^{n-i} + a^n \equiv X^n + 0 + a \mod n.$$

Därmed gäller fall "⇒".

Fall "⇐": Låt n vara ett sammansatt tal med en primtalsfaktor p med exponent $s \geq 1$ sådant att $p^s \mid n$ men $p^{s+1} \nmid n$. Notera att vi får en binomialkoefficient

$$\binom{n}{p} a^p X^{n-p}$$

bland

$$\sum_{0<i<n} \binom{n}{i} a^i X^{n-i}.$$

Notera att

$$\binom{n}{p} = \frac{n(n-1)...(n-p+1)}{p!},$$

där täljaren $n(n-1)...(n-p+1)$ har n som är delbart med p^s och $(n-1)...(n-p+1)$ är relativ prima till p. Observera också att p delar nämnaren (då $p! = p \cdot (p-1) \cdot ... \cdot 1$). Detta innebär att $\gcd(n(n-1)...(n-p+1), p!) = p$, och i utsträckning att $p^{s-1} \mid \binom{n}{p}$, men $p^s \nmid \binom{n}{p}$. Notera samtidigt att ty $\gcd(a, n) = 1$ så är $p \nmid a^p$. Därmed är

$$p^s \nmid \binom{n}{p} a^p$$

och i utsträckning är n inte en delare till $\binom{n}{p} a^p$. Därmed kommer

$$\sum_{0<i<n} \binom{n}{i} a^i X^{n-i}.$$

inte att vara kongruent med $0 \mod n$, vilket leder till att

$$(X+a)^n \neq X^n + a \pmod n.$$

Med att vi har en motsägelse när n är sammansatt och $(X+a)^n = X^n + a \pmod n$, så gäller det att om $(X+a)^n = X^n + a \pmod n$ så är n ett primtal. Därmed gäller "⇐".

Vi har nu visat att påståendet går åt båda håll, och därmed är ekvivalent, vilket var det lemma 6.1.1 påstod. □

Med detta lemma har vi ett säkert koncept för att pröva om ett heltal n är ett primtal, där konceptet pekar på att om n är ett primtal, så kommer alla termer mellan X^n och a vara delbara med n. På grund av att vi behandlar polynomer, kan det dock ta en lång tid att kontrollera om vänstra ledet är kongruent med högra ledet. Beroende på hur polynomen $(X+a)^n$ ser ut, kan man i värsta fall behöva kontrollera n koefficienter. På sida 116 anger Dietzfelbinger [6] att denna kontroll tar fler steg än att försöka dividera med kända primtalsfaktorer under $\sqrt{n}$. Med andra ord blir detta koncept opraktiskt att använda som primtalstest i denna form.

För att göra lemma 6.1.1 mer användbart använder vi oss av polynomet $X^r - 1$ för en så liten exponent r som möjligt. Med andra ord, räcker det att kontrollera om

$$(X+a)^n = X^n + a \pmod{X^r - 1, n}$$

gäller för ett potentiellt primtal n. Låt oss notera detta som ett nytt lemma för lätthetens skull.

Lemma 6.1.2. *För ett primtal n, gäller*

$$(X+a)^n = X^n + a \pmod{X^r - 1, n}$$

för alla heltal a och $r \geq 1$.

Bevis. Vi antar att n är ett primtal. Notera att från lemma 6.1.1 vet vi att för ett primtal n gäller

$$(X+a)^n = X^n + a \pmod{n}.$$

Det räcker därmed att relationen ovan inte ändras av $\mod X^r - 1$, det vill säga att

$$(X+a)^n \pmod{X^r - 1, n} \text{ och } X^n + a \pmod{X^r - 1, n}$$

ger samma resultat. Observera att i $\mod X^r - 1$ så är $X^r \equiv 1$. Vi får därmed att

$$X^n \equiv X^{n \mod r} \pmod{X^r - 1}$$

där endast exponenten n blir begränsad modulo r. Med detta kan vi säga att

$$X^n + a \equiv X^{n \mod r} + a \pmod{X^r - 1}$$

då endast exponenten n påverkas av modulo $X^r - 1$. Då $X^n + a \pmod{X^r - 1} = X^{n \mod r} + a$, och vi vet från lemma 6.1.1 att

$$(X+a)^n = X^n + a \pmod{n}.$$

Vi får därmed att $(X+a)^n = X^n + a \pmod{X^r - 1, n}$ gäller. Notera också att värdet på r eller a inte påverkar denna relation när n är ett primtal, och därmed gäller detta för alla värden av a eller r. □

Med detta lemma kan vi reducera antalet koefficienter vi behöver kontrollera beroende på valet av r, vilket leder till en mycket snabbare beräkning. Dock noterar Agrawal et. al. på sida 2 i [10] att vissa sammansatta tal som kan uppnå kravet för vissa a och r också finns. Vårt mål blir därmed att hitta ett adekvat r, där endast ett primtal n kan uppnå, och att visa att lemma 6.1.2 gäller för vissa a.

Med denna modifikation av lemmat, och de noteringar som har gjorts, kan vi visa algoritmen för AKS-testet. Vår följande algoritm i form av pseudokod kommer från Dietzfelbinger [6] sida 117 men har element från Crandall & Pomerance [14] genom att ersätta $4(\log_2 n)^2$ med $\log_2^2 n$.

AKS-primtalstest

Input: Ett heltal $n \geq 2$

if $n = a^b$ **then**

  Output: Not Prime

else

  $r \leftarrow 2$

  while $r < n$ **do**

    if $r \mid n$ **then**

      Output: Not Prime

    else

      if $\gcd(r, n) = 1$ **then**

        if $n^i \pmod r \neq 1$ *för alla* $1 \leq i \leq \lceil \log_2^2 n \rceil$ **then**

          break

        end

      end

    end

    $r \leftarrow r + 1$

  end

  if $r \geq n$ **then**

    Output: Prime

  end

  for $1 \leq a \leq \lceil \sqrt{\varphi(r)} \rceil \lceil \log_2(n) \rceil$ **do**

    if $(X + a)^n \not\equiv X^{n \mod r} + a \pmod{X^r - 1, n}$ **then**

      Output: Not Prime

    end

  end

  Output: Prime

end

Från observation av algoritmen kan vi se att algoritmen kan delas in i tre delar, och kan förklaras som följande:

1. Vi har vår input som är ett heltal n som är större eller lika med 2. Vi kontrollerar först om vårt heltal n är en potens av något tal b, vilket, om så är fallet, betyder att n är ett sammansatt tal.
2. Om n inte är en potens, vill vi nu bestämma ett heltal r för testet. Vi sätter först att r är lika med 2. I detta steg vill vi hitta ett r som är så litet som möjligt, och samtidigt uppnår kravet.
   (a) Vi prövar först om r delar n, och om så säger att n är ett sammansatt tal (då per definition för primtal så bör n endast vara delbart med sig själv eller med 1).
   (b) Om $r \nmid n$, och $\gcd(r, n) = 1$ prövar vi om

$$n^i \equiv 1 \pmod r$$

   för något i inom intervallet $1 \leq i \leq \lceil \log_2^2 n \rceil$, det vill säga från 1 till det närmsta heltalet avrundat uppåt till $\log_2^2 n$. Med detta prövning vill vi se om vårt r är tillräckligt stor nog för att uppnå kravet, det vill säga ordningen av n i $\mathbb{Z}_r^*$ är större än $\log_2^2 n$. Om det för något i ger ovan så är vårt r inte tillräckligt stor, och vi provar med ett annat r, vilket i algoritmen betyder att vi ökar

r med 1 (till exempel om $r = 2$ så blir vårt nya $r = 2 + 1 = 3$). Om vi får att

$$n^i \not\equiv 1 \pmod{r}$$

för alla $1 \leq i \leq \lceil \log_2 n \rceil^2$, så betyder det att vårt r är tillräckligt stor nog för att kunna användas, och vi avslutar loopen.

Vi upprepar detta steg tills vi antingen hittar ett användbar r och vi kan fortsätta till steg 3), eller tills $r = n$. Om vi når $r = n$ innebär det att vi har provat att dela n med alla $r \leq n$ (om $r \mid n$ så är n ett sammansatt tal per steg 2), och därmed vet att är n en primtal.

3. Vi prövar slutligen om n är ett primtal med hjälp av lemma 6.1.1. Vi provar nu med vår r som vi hittade i steg 2) om
$$(X + a)^n = X^n + a \pmod{X^r - 1, n}$$
gäller för alla $1 \leq a \leq \lceil \sqrt{\varphi(r)} \rceil \lceil \log(n) \rceil$ (Vi avrundar upp det till närmaste heltal). Om det inte gäller för någon a så är n ett sammansatt tal, och talparet (a, r) blir vittne till detta sammansatthet. När alla a är prövade, och det visade sig att lemma 6.1.1 gällde så vet vi att n är ett primtal.

6.2 Komplexiteten av AKS-primtalstest

Likväl som Miller-Rabins primtalstest räknar vi kort om komplexiteten av AKS-primtalstest. För att kunna beräkna komplexiteten vill vi gå igenom ett lemma och en sats från sida 3-4 i [10].

Lemma 6.2.1 ([10])**.** *Låt $LCM(m)$ vara det minsta gemensamma multipeln av de m första heltalen. För $m \geq 7$, så är*
$$LCM(m) \geq 2^m.$$

Med detta lemma kan vi ta upp följande sats.

Sats 6.2.2 ([10])**.** *Det finns ett heltal $r \leq \max[3, \lceil \log_2^5 n \rceil]$ sådan att ordningen av n i $\mathbb{Z}_r^*$ är strikt större än $\log_2^2 n$.*

Beviset är baserat på en errata som publicerades 2019 av Agrawal et. al. ([11]).

Bevis. Vi vill visa att ett r i intervallet $[3, \lceil \log_2^5 n \rceil]$ som uppnår satsen finns. Notera att för $n = 2$, $r = 3$ får vi att
$$\log_2^2 2 = 1$$
och ordningen av 2 i $\mathbb{Z}_3^*$ är 2 vilket innebär att satsen stämmer. Antag att $n > 2$. Notera att $\log_2^5 n > 10$, och lemma 6.2.1 kan appliceras. Observera också att för ett tal
$$m^k \leq B = \lceil \log_2^5 n \rceil$$
där $m \geq 2$, så är det största möjliga värdet för k lika med $\lfloor \log_2 B \rfloor$. Låt N definieras som följande:
$$N = n^{\lfloor \log_2 B \rfloor} \cdot \prod_{i=1}^{\lfloor \log_2^2 n \rfloor} (n^i - 1).$$
Låt s vara det minsta heltal som inte delar N, och observera följande olikhet:
$$N < n^{\lfloor \log_2 B \rfloor + \frac{1}{2} \log_2^2 n (\log_2^2 n - 1)} \leq n^{\log_2^4 n} = 2^{\log_2^5 n} \leq 2^B.$$
Per lemma 6.2.1 så är produkten av de B första heltalen minst 2^B. Om varje heltal $1, \ldots, B$ delar N, kommer $LCM(1, \ldots, B)$ dela N. Dock eftersom $LCM(1, \ldots, B) \geq 2^B$, kommer detta att vara en motsägelse vilket

innebär att ett heltal $s \leq B$ som inte delar N. Därmed är $s \leq B$. Faktorn av s som är relativt prima med n kommer att vara

$$r := \frac{s}{\gcd(s, n^{\lfloor \log_2 B \rfloor})}.$$

Observera att då s inte delar N, och $r = \frac{s}{\gcd(s, n^{\lfloor \log_2 B \rfloor})}$ innebär det att $r \nmid N$ och i utsträckning innebär det att r inte delar

$$\prod_{i=1}^{\lfloor \log_2^2 n \rfloor} (n^i - 1).$$

Detta betyder att $r \nmid (n^i - 1)$ för alla $1 \leq i \leq \log_2^2 n$, och därmed är ordningen av n i $\mathbb{Z}_r^*$ strikt större än $\log_2^2 n$.

Vi har därmed visat att ett r sådant att satsen gäller finns i intervallet $[3, \lfloor \log_2^5 n \rfloor]$, och satsen är bevisad. □

Med denna sats får vi att $r \leq \lceil \log_2^5 n \rceil$, och fallet där $n = r$ i algoritmen gäller endast när $n \leq 5690034$ enligt [10] sida 3. Vi räknar komplexiteten för situationen när $n > 5690034$. Anledningen till detta är för att inom kryptografisk användning krävs det primtal som ger tillgång till stora nycklar (i tidigare sektion ser vi att med ASCII-koder kan det krävas tal med mer än 80 siffror för att få plats med 10 bokstäver). 5690034 med endast 7 siffror räcker inte till skapandet av tillräckligt stora nycklar, och därmed krävs betydligt större primtal. Ty vi räknar med $n > 5690034$, innebär detta att vi inte räknar komplexiteten med fallet att $n = r$.

Med detta, låt oss räkna ut algoritmens komplexitet med naiv beräkning:

Sats 6.2.3. *AKS-primtalstest har en bit-komplexitet på $O(\log^{16.5} n)$ med naiv metod.*

Med hjälp från [6] sida 118-119, har vi följande bevis:

Bevis. Beviset bygger på att räkna ut antalet aritmetiska operationer som behövs, och därefter lägga till bit-kostnaden för varje aritmetisk operation i algoritmen. Varje tal i AKS-algoritmen är presenterad i binär form, och är begränsad av n^2. Bit-längden är därmed begränsad av $2 \log n$, där varje aritemtisk operation på ett sådant tal kostar $O((\log_2 n)^2)$ bit-operationer. Med detta, låt oss räkna ut den aritmetiska kostnaden för AKS-primtalstest.

Steg 1. Vi kontrollerar om vårt input n är en potens till ett primtal. Den aritmetiska komplexiteten för att kontrollera detta är som högst $O((\log n)^2 \log \log n)$ enligt lemma 3.2.6.

Steg 2. Vi söker efter ett lämpligt r för att pröva i steg 3, genom att pröva de egenskaper r ska ha på $r = 3, 4, ...$ Från sats 6.2.2 vet vi att vi prövar maximalt $\log_2^5 n$ möjliga r.

   (a) För steg 2 (a) prövar vi om r dividerar n, där vi utför en division för varje r som prövas. Därmed blir den maximala kostnaden för steg 2 (a) $O((\log_2 n)^5)$.

   (b) Inför steg 2 (b) kontrollerar vi om $\gcd(r, n) = 1$. Om vi använder oss av euklides algoritm till kontrollen tar det $O(\min\{\log(n) \log(m)\})$ aritmetiska operationer enligt lemma 3.2.7. I vårt fall innebär detta $O(\min\{\log(r), \log(n)\})$, och då $r < n$ (per krav för loop) vilket leder till att det tar $O(\log_2 r) = O(\log_2 \log_2^5 n) = O(5 \log_2 \log_2 n)$ aritmetiska operationer per omgång. Notera att vi upprepar detta maximalt r gånger, så det tar som mest $O(r \cdot \log_2 r) = O(\log_2^5 \cdot 5 \log_2 \log_2 n) = O(\log_2^5 \log_2 \log_2 n)$ aritmetiska operationer här. (Notera att i andra verk som behandlar komplexitetsanalys av AKS primtalstest tas $O(\log n)$ istället för $O(\log r)$ för att underlätta beräkningen då detta steg inte påverkar den slutligiltiga komplexiteten som kan ses nedan.)

   (c) Inom steg 2 (b) utför vi moduloberäkning i form av $n^i \mod r$. En omgång av en sådan moduloberäkning tar $O(\log_2^2 n)$ aritmetiska operationer. Lik tidigare steg upprepar vi för maximala antalet r, vilket innebär att tidskomplexiteten för aritmetiska operationer i steg 2 (b) är $O(\log_2^2 n \cdot (\log_2 n)^5) = O((\log_2 n)^7)$.

Steg 3. Notera att vi kontrollerar om lemma 6.1.2 gäller för $1 \leq a \leq \lceil \sqrt{\varphi(r)} \rceil \lceil \log_2 n \rceil$. Detta innebär att vi kontrollerar som mest $\sqrt{\varphi(r)} \log_2 n$ gånger. Varje omgång 6.1.2 prövas tar det $O(r^2 \cdot \log_2 n)$ naiva aritmetiska operationer. I detta steg tar det totalt

$$O(\sqrt{\varphi(r)} \log_2 n \cdot r^2 \cdot \log_2 n) = O(r^{2+\frac{1}{2}} \cdot \log_2^2 n) = O(((\log_2 n)^5)^{\frac{5}{2}} \cdot \log_2^2 n) = O((\log_2 n)^{\frac{25}{2}} \cdot \log_2^2 n)$$

$$= O((\log_2 n)^{\frac{29}{2}}) = O((\log_2 n)^{14.5})$$

aritmetiska operationer.

Av stegen ovan ser vi att steg 3 kräver mest aritmetiska operationer, vilket leder till att algoritmen har en aritmetisk komplexitet på $O((\log n)^{14.5})$. Med att vi räknade att bit-komplexiteten för en aritmetisk operation i denna algoritm var $O((\log_2 n)^2)$, kommer bit-komplexiteten med naiv beräkning vara $O((\log n)^{16.5})$ vilket var det vi ville visa. □

Med avancerade metoder kan bit-komplexitetet minskas till $\tilde{O}((\log_2 n)^{10.5})$. Ytterligare, genom en förbättring av begränsningen av r kan algoritmens bit-komplexitet minskas till $O((\log_2 n)^6)$ (se sida 120-122 i Dietzfelbinger [6] för detaljer).

6.3 Bevis av algoritmens korrekthet

Vi har nu presenterat det bakomliggande konceptet bakom AKS-primtalstest, angett den resulterande algoritmen samt beräknat komplexiteten. Notera att för probabilistiska primtalstester som Fermats lilla sats, och Miller Rabins primtalstest behövde vi endast bevisa att sannolikheten för ”fel” svar per omgång är mindre än $\frac{1}{2}$ ($\frac{1}{4}$ för Miller Rabin). Eftersom AKS-primtalstest är deterministiskt vill vi nu bevisa att denna algoritm fungerar som den ska, det vill säga att algoritmen ger att ett tal n är ett primtal om och endast om n är ett primtal.

6.3.1 AKS-primtalstestets huvudsats

Vårt bevis för att AKS-primtalstest är korrekt baseras på följande sats:

Sats 6.3.1 ([14])**.** *Låt n och r vara heltal sådana att:*

a) $n \geq 2$.

b) r *är ett positiv heltal där* $\gcd(r, n) = 1$.

c) Ordningen av n i $\mathbb{Z}_r^*$ är större än $\log_2^2 n$.

d) $(x + a)^n \equiv x^n + a \pmod{x^r - 1, n}$ *för alla* $0 \leq a \leq \sqrt{\varphi(r)} \log_2 n$.

Med dessa krav, om n har en primtalsfaktor $p > \sqrt{\varphi(r)} \log_2 n$ så är $n = p^m$ med ett heltal m.

Beviset för denna sats kommer från Crandall & Pomerance [14], sida 203-205. I vårt bevis anges det också en textuell förklaring för att försöka underlätta förståelsen av beviset, då det kan vara förvirrande utan det.

Bevis. Målet med beviset är att visa att om n har en primtalsfaktor $p > \sqrt{\varphi(r)} \log_2 n$ så kan n endast vara en potens av p i formen $n = p^m$ under satsens krav a) - d).

1. Vi antar först att n har en primtalsfaktor $p > \sqrt{\varphi(r)} \log_2 n$.
2. Med detta antagande definierar vi en mängd G som har alla polynomer $g(x)$ i $\mathbb{F}_p[x]$ där alla koefficienter beskrivs under modulo p som uppnår kravet $g(x)^n \equiv g(x^n) \pmod{x^r - 1}$. Definitionen blir som följande:

$$G = \{g(x) \in \mathbb{F}_p[x] : g(x)^n \equiv g(x^n) \pmod{x^r - 1}\}.$$

Observera att från kravet $g(x)^n \equiv g(x^n) \pmod{x^r - 1}$ kan vi se att G är sluten inom multiplikation. För att förtydliga, för $g_1(x), g_2(x) \in G$ får vi att produkten $g_1 g_2$ uppnår kravet på följande vis

$$(g_1(x)g_2(x))^n = g_1(x)^n g_2(x)^n \equiv g_1(x^n)g_2(x^n) \pmod{x^r - 1}$$

och därmed leder till att $g_1(x)g_2(x) \in G$. Vi vill nu visa följande:

(a) G inkluderar alla polynomer $x + a$ för $0 \leq a \leq \sqrt{\varphi(r)} \log_2 n$,

(b) G är en union av restklasser $\mod x^r - 1$.

För (a), notera att från krav d) i satsen så kommer $(x + a) \in G$. Då G är sluten under multiplikation kommer polynomen

$$\prod_{0 \leq a \leq \sqrt{\varphi(r)} \log_2 n} (x + a)^{\epsilon_a}$$

där ϵ_a är ett icke-negativt heltal, vara inkluderad i G. Samtidigt då vi har antagit att $p > \sqrt{\varphi(r)} \log_2 n$, kommer alla värden $0 \leq a \leq \sqrt{\varphi(r)} \log_2 n$ vara distinkta modulo p, vilket innebär att varje polynom från uttrycket kommer att vara distinkta i G.

För (b) vill vi visa att för $g_1 \in G$, $g_2 \in \mathbb{F}_p[x]$, om $g_1 \equiv g_2 \pmod{x^r - 1}$ så är $g_2 \in G$. Då $g_1(x) \equiv g_2(x) \pmod{x^r - 1}$, gäller även

$$g_1(x^n) \equiv g_2(x^n) \pmod{x^{nr} - 1}.$$

Då $x^{nr} - 1$ är en multipel av $x^r - 1$, gäller också att

$$g_1(x^n) \equiv g_2(x^n) \pmod{x^r - 1}.$$

Därmed får vi att g_2 uppnår kravet för G ty

$$g_2(x)^n \equiv g_1(x)^n \equiv g_1(x^n) \equiv g_2(x^n) \pmod{x^r - 1},$$

och därmed är $g_2 \in G$.

Med detta vill vi definiera en mängd J bestående av exponenter som har samma egenskap som n i definitionen av G, för alla polynom i G. Vi definierar J som följande:

$$J = \{j \in \mathbb{Z} : j > 0, g(x)^j \equiv g(x^j) \pmod{x^r - 1} \text{ för varje } g(x) \in G\}.$$

Från kravet till G vet vi att n är en element till J, och också att $1 \in J$. Observera att för varje polynom $g \in \mathbb{F}_p[x]$, gäller $g(x)^p = g(x^p)$, vilket innebär att relationen håller modulo $x^r - 1$ för alla $g \in G$. Därmed är p ett element i J. Vi vill nu visa att J är sluten inom multiplikation, det vill säga att för $j_1, j_2 \in J$ så är produkten $j_1 j_2 \in J$. Vi antar att $j_1, j_2 \in J$ och $g(x) \in G$. Observera att G är sluten under restklasser. Om $g(x)^{j_i} \in G$ kan vi se att $g(x)^{j_i} = g(x^{j_i}) \pmod{p}$, och därmed är $g(x^{j_i}) \in G$ också. Med detta kan vi observera att $j_1 j_2$ uppnår kravet för J som följande:

$$g(x)^{j_1 j_2} \equiv g(x^{j_1})^{j_2} \equiv g(x^{j_2})^{j_1} = g(x^{j_1 j_2}) \pmod{x^r - 1}.$$

Därmed är $j_1 j_2 \in J$ och J är sluten inom multiplikation.

Slutligen inom denna steg, definierar vi en splittningskropp K till $x^r - 1$ över kroppen $\mathbb{F}_p$. Per definition 4.1.11 kommer K ha alla element i $\mathbb{F}_p$, samt inkludera alla rötter till $x^r - 1$. Med detta vill vi säga att K är en ändlig kropp med karaktäristik p (vilket innebär att ordningen av elementet 1 är lika med p), samt innehåller alla r:te enhetsrötter (från [12] sida 290: alla tal z sådana att $z^r = 1$). Det vi

vill göra här är att anta en primitiv r:te enhetsrot $\zeta \in K$ och en minimal polynom h till ζ sådan att $h(x)$ blir en irreducibel faktor till $X^r - 1$. Detta leder till att

$$K = \mathbb{F}_p(\zeta) \simeq \mathbb{F}_p[x]/(h(x)),$$

vilket innebär att K är en homomorfisk bild av $\mathbb{F}_p[x]/(x^r - 1)$ (det vill säga mängden av avbildade element under en homomofri från ringen $\frac{\mathbb{F}_p[x]}{x^r-1}$ till ringen K) där x avbildas på ζ. Vi kallar homomorfin för ϕ, där $\phi(x) = \zeta$.

3. Med att G, J och K är definierade vill vi skapa en motsägelse som leder till vad vi vill visa i detta bevis. Vi definierar först mångden $\overline{G}$ som är bilden av G under ϕ i den tidigare noterade homomorfiska bilden K. Med andra ord de polynomer $g(x)$ där x är ersatt med r:te primitiva roten ζ. $\overline{G}$ definierar vi som följande:

$$\overline{G} = \{\gamma \in K : \gamma = g(\zeta) \text{ för någon } g(x) \in G\}.$$

Observera att $\overline{G}$ är en delmängd av K.

Vi betraktar nu den delgrupp H till den multiplikativa gruppen $\mathbb{Z}_r^*$ genererad av n och p, och beteckna storleken av H som d, det vill säga $d =\mid H \mid$. Vi definierar med detta delmängden G_d till G som följande:

$$G_d = \{g(x) \in G : g(x) = 0 \text{ eller } \deg g(x) < d\}.$$

Notera att kravet pekar på att graden av polynomet $g(x) \in G_d$ är strikt mindre än d. Vi vill nu visa att en injektion från G_d till $\overline{G}$ finns när homomorfin ϕ ovan är begränsad till G_d. Notera att ty d är ordningen till en delgrupp till $\mathbb{Z}_r^*$, så är $d \leq \varphi(r)$. Då $\varphi(r) < r$, är alla element i G_d distinkta mod $x^r - 1$. Med detta, anta två element $g_1(x), g_2(x) \in G_d$, där $g_1(\zeta) = g_2(\zeta)$. Vi vill nu visa att $g_1(x) = g_2(x)$, genom att först anta en exponent $j = n^a p^b$ där a och b är icke-negativa heltal, det vill säga $j \in J$. Vi kan då säga att

$$g_1(\zeta^j) = g_1(\zeta)^j = g_2(\zeta)^j = g_2(\zeta^j).$$

Vi kan konstatera att denna ekvation gäller för d olika värden av $j \mod r$ (notera att i kravet för J så kommer j att begränsas $\mod r$). Notera att när a och b i $j = n^a p^b$ varierar bland alla icke-negativa heltal, får vi hela delgruppen H (som genereras av n och p), och därmed får alla d möjliga element i H. Dock så är alla ζ^j distinkta om alla expontenter j är distinkta $\mod r$, då vi från tidigare har etablerat att ζ är en primitiv r :te enhetsrot, där $\zeta^r = 1$. Detta leder till att polynomet $g_1(x) - g_2(x)$ kommer att ha d stycken distinkta rötter i K. Dock då vi har etablerat att alla polynomer i G_d har en grad strikt mindre än d kan detta inte gälla enligt sats 4.1.12, vilket leder till att polynomet $g_1(x) - g_2(x)$ är ett nollpolynom, och att $g_1(x) = g_2(x)$ måste gälla. Därmed är ϕ injektiv från G_d till $\overline{G}$.

Med att vi har etablerat att en injektion från G_d till $\overline{G}$ sker, vill vi använda denna relation för att uppskatta storleken av G_d samt $\overline{G}$. Vi betraktar följande polynomer baserat på kravet till G_d:

$$g(x) = 0 \text{ eller } g(x) = \prod_{0 \leq a \leq \sqrt{d}\log_2(n)} (x + a)^{\epsilon_a}$$

där ϵ_a är 0 eller 1. Notera att dessa polynomer ingår i G då $d \leq \varphi(r)$, eftersom vi har etablerat att

$$\prod_{0 \leq a \leq \sqrt{\varphi(r)}\log_2 n} (x + a)^{\epsilon_a}$$

ingår i G. Notera att då d är åtminstone lika stor som ordningen av n i $\mathbb{Z}_r^*$, så gäller $d > \log_2^2 n$, det vill säga att $\sqrt{d} > \log_2 n$. Med detta kan vi säga att

$$d > \sqrt{d}\log_2(n).$$

Därmed ingår varje polynom

$$g(x) = 0 \text{ eller } g(x) = \prod_{0 \leq a \leq \sqrt{d}\log_2(n)} (x+a)^{\epsilon_a}$$

i G_d så länge alla ϵ_a inte är lika med 1, vilket eventuellt skulle kunna leda till en grad större än eller lika med d. Vi kommer därmed ha minst

$$1 + (2^{\lfloor \sqrt{d}\log_2(n) \rfloor + 1} - 1) = 2^{\lfloor \sqrt{d}\log_2(n) \rfloor + 1}$$

polynomer i G_d. Notera att

$$2^{\lfloor \sqrt{d}\log_2(n) \rfloor + 1} > 2^{\sqrt{d}\log_2(n)}$$

där

$$2^{\sqrt{d}\log_2(n)} = (2^{\log_2 n})^{\sqrt{d}} = n^{\sqrt{d}}.$$

Därmed vet vi att $|G_d| > n^{\sqrt{d}}$, och ty det finns en injektion från G_d till $\overline{G}$ kan vi därmed säga att

$$|\overline{G}| \geq |G_d| > n^{\sqrt{d}}.$$

4. Då vi har visat att $|\overline{G}| \geq |G_d| > n^{\sqrt{d}}$, går vi tillbaka till kroppen K. Vi hade etablerat att $K \simeq \mathbb{F}_p[x]/(h(x))$ där $h(x)$ var ett irreducibelt polynom. Om vi anger att graden av $h(x)$ är k, kan vi säga att $K \simeq \mathbb{F}_{p^k}$, där den multiplikativa gruppen $\mathbb{F}^*_{p^k}$ av nollskilda element har ordningen $p^k - 1$. Med detta, om vi ansätter att j, j_0 är positiva heltal, där de är kongruenta med varandra $\mod p^k - 1$ (det vill säga $j \equiv j_0 \pmod{p^k - 1}$), och ett element $\beta \in K$ får vi följande:

$$\beta^j = \beta^{j_0}.$$

Observera att om $\beta = 0$ är detta uppenbart. Om $\beta \neq 0$, det vill säga $\beta \in K^*$ ansätter vi att $j = j_0 + a(p^k - 1)$ då $j \equiv j_0 \pmod{p^k - 1}$, och $\beta \in K^*$. Vi kan då skriva om β^j som följande:

$$\beta^j = \beta^{j_0 + a(p^k - 1)} = \beta^j \cdot \beta^{a(p^k-1)} = \beta^j \cdot (\beta^{p^k-1})^a.$$

Då K^* har ordningen $p^k - 1$, innebär det per Lagranges sats (4.1.8) att varje element i gruppen $K^* \simeq \mathbb{F}^*_{p^k}$ har en ordning som är en delare till $p^k - 1$, och därmed gäller

$$\beta^{p^k-1} = 1 \text{ för alla } \beta \in K^*.$$

Med detta får vi att

$$\beta^j \cdot (\beta^{p^k-1})^a = \beta^j \cdot 1^a = \beta^j.$$

Från detta definierar vi mängden J' som följande:

$$J' = \{j \in \mathbb{Z} : j > 0, j \equiv j_0 \pmod{p^k - 1} \text{ för någon } j_0 \in J\}.$$

Notera att J' innehåller J som en delmängd, och därmed kan följande observationer göras på J':

(a) Då J är sluten inom multiplikation är J' också sluten inom multiplikation. För att visa detta, låt $j_1, j_2 \in J'$. Låt $j_i \equiv u_i \pmod{p^k - 1}$ för några $u_i \in J$. Observera att produkten $j_1 j_2$ blir som följande då $j_1 \equiv u_1 \pmod{p^k - 1}$ och $j_2 \equiv u_2 \pmod{p^k - 1}$:

$$j_1 j_2 \equiv u_1 u_2 \pmod{p^k - 1}.$$

Då J är sluten under multiplikation innebär det att $u_1 u_2 \in J$ när $u_1, u_2 \in J$. Då $j_1 j_2$ är kongruent med $u_1 u_2$ modulo $p^k - 1$ innebär det per definition för J' att $j_1 j_2 \in J'$.

(b) För $j \in J'$ med $j \equiv j_0 \pmod{p^k - 1}$, med $j_0 \in J$, och $g(x) \in G$, gäller $g(\zeta)^j = g(\zeta)^{j_0} = g(\zeta^{j_0}) = g(\zeta^j)$.

(c) Eftersom $np^{k-1} = \frac{n}{p} \cdot p^k \equiv \frac{n}{p} \cdot 1^k \equiv \frac{n}{p} \pmod{p^k - 1}$, och eftersom $n, p \in J$ och J är sluten under multiplikation, så är n/p också en element i J'.

Vi vill använda oss av dessa observationer för att visa vår sista motsägelse. Vi betraktar först heltalet $p^a(n/p)^b$ där a, b är heltal inom intervallet $[0, \sqrt{d}]$. Notera att antalet kombinationer av (a, b) för $p^a(n/p)^b$ är mer än d. För att visa detta, låt $m = \lfloor \sqrt{d} \rfloor$. Då är $m + 1 > \sqrt{d}$, och antalet par heltal (a, b) är därmed

$$(m + 1)^2 > (\sqrt{d})^2 = d.$$

Notera också att p och n/p är element i delgruppen H till $\mathbb{Z}_r^*$ som vi definierade tidigare, eftersom H innehåller den multiplikativa delgruppen som genereras av p och därmed innehåller p^{-1}. Då antalet val av (a, b) är större än ordningen av delgruppen kommer vi enligt postfacksprincipen kunna säga att det finns två distinkta par (a_1, b_1) och (a_2, b_2) sådan att två element $j_1, j_2 \in J'$ där $j_1 = p^{a_1}(n/p)^{b_1}$ och $j_2 = p^{a_2}(n/p)^{b_2}$ är kongruenta med varandra $\mod r$. Observera att $j_1, j_2 \in J'$ eftersom n/p och p ligger i J'. Detta medför, eftersom ζ är en r:te enhetsrot, att

$$\zeta^{j_1} = \zeta^{j_2}$$

och ty $j_1, j_2 \in J'$ så gäller för alla $g(x) \in G$

$$g(\zeta)^{j_1} = g(\zeta^{j_1}) = g(\zeta^{j_2}) = g(\zeta)^{j_2},$$

vilket betyder att $\gamma^{j_1} = \gamma^{j_2}$ för alla $\gamma \in \overline{G}$. Vi ansätter nu polynomet $f(x) = x^{j_1} - x^{j_2}$ där $f(x) \in K[x]$. Då vi har etablerat att $\gamma^{j_1} = \gamma^{j_2}$ för alla $\gamma \in \overline{G}$, betyder det att alla element i $\overline{G}$ är möjliga rötter till $f(x)$. Notera dock att den största kombinantionen av (a, b) som exponenterna j_1, j_2 kan ha är $(\sqrt{d}, \sqrt{d})$ då de är begränsade till heltal i intervallet $[0, \sqrt{d}]$. Vi får därmed att

$$j_1, j_2 \leq p^{\sqrt{d}}(n/p)^{\sqrt{d}} = n^{\sqrt{d}},$$

och då graden av ett polynom bestäms av största exponenten i polynomet, kan vi säga att

$$\deg f \leq n^{\sqrt{d}}.$$

Jämför detta med de möjliga rötterna, vilket är alla rötter i $\overline{G}$, där vi har etablerat att $|\overline{G}| > n^{\sqrt{d}}$, ser vi att antalet element i $\overline{G}$ är betydligt större än graden av $f(x)$. Vi får därmed en motsägelse enligt sats 4.1.12, vilket betyder att $f(x)$ är en nollpolynom. Då $f(x)$ är ett nollpolynom måste $j_1 = j_2$ gälla.

5. Vi har nu vårt sista steg. Vi har etablerat att $j_1 = j_2$ måste gälla enligt ovan, vilket betyder att

$$j_1 = j_2 \Leftrightarrow p^{a_1}(n/p)^{b_1} = p^{a_2}(n/p)^{b_2},$$

och då (a_1, b_1) och (a_2, b_2) är distinkta får vi följande utveckling:

$$p^{a_1}(n/p)^{b_1} = p^{a_2}(n/p)^{b_2} \Leftrightarrow p^{a_1 - b_1} n^{b_1} = p^{a_2 - b_2} n^{b_2} \Leftrightarrow n^{b_1 - b_2} = p^{a_2 - b_2 - (a_1 - b_1)} \Leftrightarrow n^{b_1 - b_2} = p^{b_1 - b_2 - a_1 + a_2}.$$

Observera att eftersom (a_1, b_1) och (a_2, b_2) är distinkta så är $b_1 \neq b_2$. Med unik primfaktorisering kan vi med detta säga att n är en potens till p. Därmed har vi visat att n är en potens av p.

□

Vi har en sats kvar för att visa AKS-primtalstestets korrekthet. Det är en sats från sida 122 i Dietzfelbinger [6] som binder algoritmen till sats 6.3.1.

Sats 6.3.2. *AKS-algoritmen ger att ett heltal $n \geq 2$ är ett primtal om och endast om n är ett primtal.*

Bevis. Vi vill visa att denna ekvivalensrelation är sann, det vill säga att både "⇒" och "⇐" av påståendet gäller. Vi börjar med "⇐".

Fall "⇐": Låt oss anta att $n \geq 2$ är ett primtal och observera algoritmen.

I steg 1 av algoritmen ser vi att då n inte är en potens $n = a^b$ för någon $b > 1$, kommer algoritmen inte ge outputen "Not prime", och vi kan fortsätta vidare till steg 2.

I steg 2 ser vi att i (a), då r alltid är mindre än primtalet n, kommer r aldrig att vara en delare till n. Om detta fortsätter tills $r = n$ ger algoritmen outputen "Prime", vilket betyder att n är ett primtal enligt algoritmen.

Om vi antar att något r uppnår kravet i del (b) och vi fortsätter vidare till steg 3, betyder det att r är större än ordningen av n i $\mathbb{Z}_r^*$, som är större än $\lceil \log_2^2 n \rceil \geq \log_2^2 n$. Därmed är $\sqrt{r} > \log_2 n$, och

$$n > r > \sqrt{r} \log_2 n.$$

Med att vi har en adekvat r, och n är ett primtal vet vi per lemma 6.1.1 att kongruensen i steg 3 kommer att hålla för alla a med $a \leq \sqrt{r} \log_2 n$, vilket slutligen ger outputen "Prime". Därmed ger algoritmen att n är ett primtal om n är ett primtal.

Fall "⇒": Vi vill nu visa att om algoritmen ger att $n \geq 2$ är ett primtal så är n ett primtal. Vi antar att algoritmen ger outputen "Prime" i antingen steg 2. (a) eller steg 3. Vi har då två fall.

Fall 1 **Output "Prime" ges i steg 2.(a)**

I detta fall har vi kontrollerat om alla r i $2 \leq r < n$ är en delare till n, vilket är samma som att direkträkna. Då alla heltal under $2 \leq r > n$ inte är en delare till n får vi per den klassiska definitionen för primtal att n är ett primtal.

Fall 2 **Output "Prime" ges i steg 3.**

Vi har gått över från steg 2. genom att hitta ett r som uppnår kravet för att testas i steg 3. Vi ansätter att $r < n$ är ett sådant heltal som uppnår kravet för att prövas i steg 3. Vi kontrollerar nu om n och r uppnår kraven i sats 6.3.1. Vi får då att

i. $n \geq 2$ uppnås då det är vår input.

ii. r är ett positiv heltal där $\gcd(r, n) = 1$ uppnås, då vi kontrollerade det i steg 2.(a).

iii. Ordningen av n i $\mathbb{Z}_r^*$ är större än $\log_2^2 n$, då vi kontrollerade det i steg 2.(b).

iv. $(X + a)^n \equiv X^n + a \pmod{X^r - 1, n}$ i $\mathbb{Z}_n[x]$ för alla $0 \leq a \leq \sqrt{\varphi(r)} \log_2 n$ uppnås då vi kontrollerade det i steg 3., och då algoritmen hade gett output "Prime" betyder det att detta krav uppnås.

Med att alla krav i sats 6.3.1 uppnås genom stegen, så betyder det att n antingen är en potens till ett primtalsfaktor p sådant att $n = P^b$ för $b > 1$, eller så är n ett primtal. Dock eftersom vi provade om n var ett potens i steg 1., vilket visade att n inte var det, betyder det att n är ett primtal. Därmed är n ett primtal om algoritmen ger att n är ett primtal.

Med att vi har kunnat visa att påståendet i sats 6.3.2 gäller åt båda håll, har vi bevisat satsen. □

Med detta vet vi att AKS-primtaltest är deterministiskt och säger att ett heltal n är ett primtal om och endast om n är ett primtal. Vi kan nu säga att vi har gått igenom det innehåll som vi ville presentera innan vi besvarar studiens frågeställningar.

7 Sammanfattning och slutdiskussion

Låt oss sammanfatta det vi har gått igenom i det här arbetet. Vi svarar på de fyra första frågeställningarna, och kort diskuterar svaret till vår femte frågeställning i en separat subsektion.

7.1 Svaren på våra fyra första frågeställningar

7.1.1 fråga 1: Vad har primtalstester för mening inom kryptografin?

Primtalstester har sin betydelse främst inom asymmetrisk kryptografi, där stora primtal är viktiga för att konstruera nyckeln. Ett typiskt exempel är RSA-kryptografin, som konstruerar sin nyckel från två primtal. Från perspektivet av primtalstester hade RSA-kryptografin en stor roll, där behovet för stora primtal gav upphov till utvecklandet av mer praktiska primtalstester. Resultatet av detta behov är bland annat de primtalstester vi har presenterat här, samt flera andra som kan hittas i andra verk.

7.1.2 fråga 2: Hur funkar primtalstester? Är det deterministiskt?

Primtalstester fungerar genom att ta en egenskap som är unik hos primtal, och pröva ett valt tal n med ett visst antal tal a från ett intervall. Om det ser ut att det prövade talet n inte uppnår kravet skapad av konceptet, blir det prövade talet a ett vittne till n:s sammansatthet. Beroende på hur man väljer talen a från intervallet finns det två stora grupper av primtalstester.

Den ena är probabilistiska primtalstester, där man på slumpen väljer ett bestämt antal a från intervallet och prövar n med det. Här bygger primtalstestet på att sannolikheten att använda en lögnare (ett tal a som ger att n uppnår kravet även om n är ett sammansatt tal) är låg nog för att ge ett rimligt resultat efter ett bestämt antal omgångar som n prövas.

Den andra kallas för deterministiska primtalstester och kan ses mer som ett sätt att bevisa att ett tal n är ett primtal. Deterministiska primtalstester bygger därmed på att algoritmen ger ett pålitligt bevis på att n är ett primtal om n visar egenskapen unikt för primtal, för alla prövade tal a i intervallet.

7.1.3 fråga 3: Vad är Miller-Rabin test? Hur fungerar det?

Miller-Rabin primtalstest är ett probabilistiskt primtalstest som baseras på kvadratroten av 1, där för ett tal n och ett relativt prima tal a att

$$n - 1 = 2^k u$$

ska antingen ge att $a^u \equiv 1 \pmod{n}$ eller $a^{2^i u} \equiv n - 1 \pmod{n}$, $i = 1, 2, ..., k - 1$ för ett visst antal värden a för att vi ska kunna säga att n är ett möjlig primtal. Primtalstestet är bevisat att ha mindre än en fjärdedels risk att beträffa en så kallad MR-lögnare, och sägs ge ett rimligt resultat efter 50 prövningar.

7.1.4 fråga 4: Vad är AKS-primtalstest? Hur fungerar testet?

AKS-primtalstest är ett deterministiskt primtalstest som bygger på en princip baserad på Fermats sats med polynomer, nämligen att för ett primtal p så ska

$$(X + a)^p \equiv X^p + a \pmod{X^r - 1, p}$$

gälla för alla heltal a, och $r \geq 1$. Primtalstestet är känt för att ge ett definitivt svar om ett valt tal n inom en polynomiell tid utan att använda sig av villkor som att anta att Riemannhypotesen är sann.

7.2 Praktikaliteten av AKS-primtalstest jämfört med Miller-Rabins primtaltest

Om vi reflekterar tillbaka till vår sektion om kryptografins historia, så betyder det inte att om en metod är bra så kommer det att användas. Exempelvis så kunde vi se att Vigeneré-chiffer som var svårare att knäcka

än en monoalfabetisk chiffer med koder, sällan användes av professionella under tidigmodern tid. Anledningen var att det ansågs vara för komplicerat, och tog tid. Det var först med uppfinningen av telegrafin och behovet att snabbt kunna skapa nycklar utan att behöva trycka ut kodböcker som Vigeneré-chiffret blev populärt inom offentlig anvädning, speciellt militären. Med andra ord så kan man säga att tiden som tar är en av de viktigaste aspekterna gällande kryptografisk användning.

Notera att vi har i tidigare sektioner räknat ut komplexiteten av Miller-Rabins primtalstest respektive AKS primtalstest med naiva metoder. För Miller-Rabins primtalstest är bit-komplexiteten $O(k \cdot (\log n)^3)$ där k beskriver antalet omgångar Miller-Rabins test utförs, medan för AKS-primtalstest räknade vi ut att bit-komplexiteten var $O((\log n)^{16.5})$. Här ser vi tydligt hur AKS-primtalstestet är betydligt komplicerat än Miller-Rabins primtastest. Vidare för AKS-primtalstest påstod tidigare verk att med förenklingar och mer sofistikerade metoder, kunde AKS-primtalstest förkortas ner till $\tilde{O}((\log n)^6)$ bit-operationer. Miller-Rabins primtaltest kan förkortas ner till $\tilde{O}(k \cdot (\log n)^2)$ med avancerade metoder. Kort sagt ser vi att AKS-algoritmen har en snabbare tillväxt än Miller-Rabins primtalstest ju större input n vi har. Vi kan därmed säga att Miller-Rabins primtalstest är snabbare, och därmed mer praktiskt än AKS-primtalstest. Utanför det matematiska analyset är Miller-Rabins primtalstest en av de mest populära primtalstester i bruk.

Dock kunde vi också se att teknologins utveckling gav tillgång till mer komplicerade kryptografiska metoder. Vi kunde se i [17] hur datorer användes för att skapa komplicerade polyalfabetiska chiffer som mekaniska apparater som Enigma inte klarade av. Detta kan nog appliceras på primtalstester också. Med detta får jag gissa att AKS-primtalstest, även om det inte är praktiskt nu, kanske kan bli praktiskt i framtiden med uppfinningen av snabb-beräknande datorer. Dock är det nog värt att notera att mer praktiska tester såsom Miller-Rabin, eller andra deterministiska tester (som är snabbare även om det bygger på obevisade hypoteser), troligen blir betydligt snabbare att utföra. På det sättet kan man nog säga att med teknologins utveckling i form av praktiska kvantdatorer som sägs utföra beräkningar mycket snabbt, kan AKS-primtalstest kanske uppnå en praktiskt nivå som vi har nu med exempelvis Miller-Rabin, eller andra snabbare deterministiska primtalstester, och kan därmed användas för kryptografiska ändamål. Dock kommer dessa andra primtalstester troligen att också bli snabbare vilket kan betyda att AKS-primtalstest fortfarande är slöare, och därmed opraktiskt att använda.

Om vi sammanfattar denna lilla essä till ett svar på frågeställningen kan man säga att AKS-testet inte är praktiskt jämfört med Miller-Rabins primtalstest. Det kräver betydligt mer operationer som algoritm, jämfört med Miller-Rabin som prövar talet n med ett fåtal möjliga vittnen från ett intervall. Inom kryptografin, där man kan säga att enkelhet och tid är de viktiga aspekterna som fokuseras, är AKS därmed opraktiskt. Det är dock viktigt att notera att AKS-primtalstest inte sätter fokus på att vara ett praktiskt primtalstest för nyckelskapande inom kryptografin. Istället var fokuset som Agrawal et. al. hade med primtalstestet att konstruera ett primtalstest som kunde bevisa att ett valt heltal n är ett primtal utan villkor (t.ex. att anta att Riemannhypotesen är sann) under en polynomisk tid.

7.3 Slutord

Med detta har vi svarat på alla frågeställningar till detta arbete. Som författare hoppas jag att detta arbete har fungerat som en ingång till både primtalstester och kryptografi, vilket arbetet syftade på. Dock kan en rad brister noteras i detta arbete. För det första är historiesektionen oerhört lång, med tanke på att aktuella området med primtalstester behandlas völdigt kort i slutet av sektionen. Samtidigt känns det onödigt långt för att presentera två primtalstester. Speciellt när man räknar med att många teorier kapades av från referenslitteraturen för att få plats. Gällande primtalstester finns det flera som inte presenterades här, som Solovay-Strassens primtalstest som tas upp av bland annat Dietzfelbinger [6] och Gallier [15]. Detta kan vara något att läsa för de intresserade. Jag som författare får härmed be om ursäkt för de oklarheter som har dykt upp i arbetet, trots att detta skulle fungera som introduktion. Med detta vill jag som författare tacka, och återigen hoppas att denna introduktion har väckt intresset för primtaltester, och kryptografi.

Referenser

[1] Norman L. Biggs. *Discrete Mathematics*. Oxford University Press, second edition, 2002.

[2] Keith Conrad. Carmichael numbers and korselt's criterion. Online. `https://kconrad.math.uconn.edu/blurbs/ugradnumthy/carmichaelkorselt.pdf` (tagen november 2025).

[3] Keith Conrad. The chinese remainder theorem. Online. `https://kconrad.math.uconn.edu/blurbs/ugradnumthy/crt.pdf` (tagen november 2025).

[4] Keith Conrad. The miller-rabin test. Online. `https://kconrad.math.uconn.edu/blurbs/ugradnumthy/millerrabin.pdf` (tagen november 2025).

[5] Harold Davenport. *THE HIGHER ARITHMETIC AN INTRODUCTION TO THE THEORY OF NUMBERS*. Cambridge University Press, eight edition, 2008.

[6] Martin Dietzfelbinger. *Primality Testing in Polynomial Time – From Randomized Algorithms to "PRIMES in P"*. Springer-Verlag Berlin, 2004.

[7] Whitfield Diffie & Martin Hellmann. New directions in cryptography. *IEEE*, September 1976. `https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1055638`.

[8] Jill Pipher & Joseph H. Silverman Jeffrey Hoffstein. *An introduction to Mathematical Cryptography*. Springer, New York, USA, second edition, 2014.

[9] David Kahn. *The Code Breakers*. Oxford University Press, 1967.

[10] Neeraj Kayal & Nitin Saxena Manindra Agrawal. Primes in p. *Annals of Mathematics*, September 2004. `https://doi.org/10.4007/annals.2004.160.781`.

[11] Neeraj Kayal & Nitin Saxena Manindra Agrawal. Errata:primes in p. *Annals of Mathematics*, January 2019. `https://doi.org/10.4007/annals.2019.189.1.6` (tagen augusti 2026).

[12] Richard Earl & James Nicholson. *Oxford CONCISE DICTIONARY OF Mathematics*. Oxford University Press, sixth edition, 2021.

[13] Rudold Lidl & Harald Niederreiter. *ENCYCLOPEDIA OF MATHEMATICS AND ITS APPLICATIONS 20: FINITE FIELDS*. Cambridge University Press, second edition, 1997.

[14] Richard Crandall & Carl Pomerance. *Prime Numbers A Computational Perspective*. Springer New York, second edition, 2006.

[15] Jean Gallier & Joselyn Quaintance. Notes on primality testing and public key cryptography part 1: Randomized algorithms miller-rabin and solovay-strassen tests. Online, 2025. `https://www.cis.upenn.edu/~jean/RSA-primality-testing.pdf`.

[16] Mary Radcliffe. Math 127: Chinese remainder theorem. Online, April 2019. `https://www.math.cmu.edu/~mradclif/teaching/127S19/Notes/ChineseRemainderTheorem.pdf` (tagen november 2025).

[17] Simon Singh. *The code book*. Delacorte Press, 2001.

[18] Ronald L. Rivest & Clifford Stein Thomas H. Cormen, Charles E. Leiserson. *Introduction to Algorithms*. MIT Press, third edition, 2009.

[19] Amin Witno. Won series in discrete mathematics and modern algebra volume 5 the primitive root theorem. Online, April 2012. `http://witno.com/philadelphia/notes/won5.pdf` (tagen augusti 2026).

[20] Doron Zeilberger. Dr. z.'s number theory lecture 13 handout: Wilson's theorem and fermat's little theorem. Online, November 2024. `https://sites.math.rutgers.edu/~zeilberg/numtheory/L13.pdf` (tagen februari 2026).

Rättelseblad till arbete L13

Fredrik Huss

Augusti 2026

Följande misstag har noterats i den inlämnade uppsatsen:

Sida 16. I definition 4.1.15 används ”splittringsfält” istället för ”splittringskropp” fast det står ”splittringskropp” i definitionstiteln. För att vara konsekvent med vad som har skrivits i definitionstiteln och senare delen av arbetet bör detta ändras till splittringskropp.

Sida 18-19. I beviset till sats 4.1.18 har vi intervallet $1 \leq a < p$ i första meningen av beviset och $1 \leq a < 1$ i senare del av beviset. Här bör intervallet istället vara $1 \leq a < p^m$.