SJÄLVSTÄNDIGA ARBETEN I MATEMATIK

MATEMATISKA INSTITUTIONEN, STOCKHOLMS UNIVERSITET

RSA metoden för kryptering

av

Kaveh Baymani

2022 - No K10

RSA metoden för kryptering

Kaveh Baymani

Självständigt arbete i matematik 15 högskolepoäng, grundnivå

Handledare: Torbjörn Tambour

2022

Innehåll

RSA metoden för kryptering

Sammanfattning

RSA som var den första metoden som implementerade asymmetrisk kryptering, har lyckats bibehålla sin plats under många år trots enorma utvecklingen som datorer och technologin har gått genom. I detta arbete ska vi undersöka RSA både i praktik och teori för att ta reda på vad det är som ligger bakom den långvariga succén. Detta kräver att vi lär oss mer om primalitet och primtalstester som har en stor tillämpning inom säkerheten som erbjuds av RSA algoritmen. Det är omöjligt att täcka allt inom ett enda arbete men Fermats och Miller-Rabins primalitetstester kommer ligga under noggrann granskning. Utöver det kommer vi läsa lite om kvantdatorer och ta reda på vilken påverkan de har på de existerande krypteringsmetoder. Slutligen erbjuder arbetet en modifierad version av RSA som kan möjligtvis resultera i ökad säkerhet när det kommer till vissa typer av angrepp som RSA har fått stå emot.

Nyckelord: RSA, asymmetrisk kryptering, primalitet, primtalstest, kvantdatorer, säkerhet

Förord

Jag vill rikta ett stort och hjärtligt tack till min handledare, Torbjörn Tambour, för all hjälp och stöd under arbetets gång.

1

Introduktion

Kryptering från ett historiskt perspektiv

Behovet av att kryptera meddelanden framkom redan efter att människorna började för första gången anteckna det som skedde dagligen runt omkring sig. Man kan anta att tidigare i historien fördes meddelanden vidare med hjälp av mimik eller teckenspråk. Utvecklingen av skriftspråket la grunden till skapandet av stenografi och kryptologi som anses vara de två främsta metoder för att hemlighålla meddelanden idag [5, 5].
Kryptografi innebär för avsändaren att göra sitt budskap obegripligt och förhindra fienden under krigstiden, eller vem som helst som försöker avlyssna samtalet och komma åt informationen som riktar sig endast åt en tilltänkt mottagare, ifrån att uppfatta det som finns gömd i meddelandet. Ordet kryptografi har sina rötter i det grekiska språket och betyder gömd [5, 5].

Kryptering av skrift och meddelanden görs på ett sådant vis som håller utomstående mottagaren borta ifrån att förstå koden och kunna dekryptera texten. Meddelandet går då endast att dekryptera av personen som sändaren önskar sig och har redan utrustat med dekrypteringskoden [1, 1]. På grund av att sekretessen alltid varit oerhört viktigt för både avsändaren och mottagaren genom historien och framför allt vid förekommandet av krig och konflikter så har det alltid funnits enorma varningsflaggor som ställt frågor kring säkerheten av alla tidigare krypteringsmetoder.

Alla metoder som människorna använde sig av innan RSA, refererades till som symmetrisk kryptering eller delad-nyckel kryptering. Precis som ordet symmetrisk i titeln antyder så definieras denna typ av kryptering som användning av samma kryptonyckel vid kryptering såväl som dekryptering. Med andra ord, ansågs kryptering och dekryptering som två

symmetriska process. Denna typ av kryptering baserades på att både avsändaren och tilltänkta mottagaren behövde ha tagit del av information gällande vilken metod som ska användas och dessutom kände till nyckeln vid dekryptering [1, 1]. Detta innebär att avsändaren behövde distribuera nyckeln i ett separat meddelande eller så hade de redan kommit överens om dekrypteringsmetoden sedan tidigare.

Säkerhetsbristen som finns hos de flesta symmetriska modeller förblev ett grundläggande problem. Det var relativt enkelt att dekryptera meddelanden där nyckeln som användes till att kryptera informationen med var samma som dekrypteringsnyckeln. En tredje person kunde då lika enkelt ta reda på vad nyckeln är och göra om processen baklänges för att bryta ner den krypterade koden. Lösningen på säkerhetsbristen förblev då ett problem framtill 1976 när Whitfield Diffie och Martin Hellman upptäckte till slut en metod baserat på asymmetriska nycklar och tog första steget mot en förbättrad säkerhetsnivå. [1, 2]

Diffie-Hellman metoden baserades på asymmetrisk kryptering till skillnad från alla andra tidigare metoder som använde sig av symmetriska nycklar. Detta innebär att mottagaren hade sin egen privata nyckel utöver de offentliga nycklarna och fick använda sig av den för att dekryptera meddelandet. Dagens kryptosystem bygger på ett sådant system där sändaren som krypterar använder en särskild krypteringsnyckel och mottagaren använder en annan som kallas dekrypteringsnyckel [5, 6].

Diffie och Hellman var de första som kom fram till idén att använda två distinkta nycklar där ena nyckeln var privat och den andra var offentlig. I denna metod är den privata nyckeln dekrypteringsnyckeln och den offentliga nyckeln är krypteringsnyckeln [1, 2]. Om en godtycklig person ”Bob” ska skicka ett meddelande till en mottagare ”Alice” behöver han endast använda sig av den offentliga nyckeln för att kunna kryptera meddelandet och då är mottagaren ”Alice” den enda individen i hela världen som kan dekryptera meddelandet med hjälp av den privata nyckeln som hon och endast hon har tillgång till [1, 2].

Trots att Diffie och Hellman lyckades komma upp med idén om distinkta nycklar och var de första som upptäckte behovet av en asymmetrisk chiffernyckel så misslyckades de med att faktiskt hitta ett välfungerande koncept till att generera nycklar. Däremot ledde idén som de la fram till ökat intresse hos andra forskare som arbetade inom samma krets och la grunden till utvecklingen som gjordes senare av framför allt Rivest, Shamir och Adleman [1,2].

RSA-metoden för kryptering togs fram av Ron Rivest, Adi Shamir och Len Adleman som var en grupp på tre personer som bestod av två dataloger och en matematiker som arbetade tillsammans under en längre period där datorvetenskapsmännen kom upp med en hel del idéer och Adleman som var gruppens matematiker fortsatte vederlägga dem. Adlemans tålamod gynnade hela gruppen i långa loppet trots att processen var tidskrävande och även nedslående för Rivest och Shamir. Varje misslyckande teorin pushade dem närmare och närmare och till slut lyckades Rivest och fick ett genombrott i april 1977 och fick svar på frågan som de hade jobbat med hela året.

Är det möjligt att hitta ett välfungerande sätt som möjliggör och underlättar dekryptering med asymmetriska nycklar om man utrustar mottagaren med någon typ av speciell information? De besvarade frågan med hjälp av modulär aritmetik och lyckades komma upp med en metod att generera distinkta nycklar som ska nu beskrivas i detaljer under kommande avsnitt. Därefter döptes metoden till RSA som tog sin förkortning efter Rivest, Shamir och Adleman [1,3]

2

Matematiska begrepp – kryptering

2.1 Kongruenser

RSA metoden för kryptering bygger på modulär aritmetik och Säkerheten i systemet ligger i att det är relativt enkelt att multiplicera två stora primtal men det är praktiskt taget omöjligt att hitta primtalsfaktorer i ett stort heltal med hundratalssiffror [7, 4].

För att begripa innebörden av moduloräkningar som kan även refereras till som kongruensräkningar måste man först studera kongruenser.

Låt n vara ett positivt heltal.

Två godtyckliga heltal a och b är kongruenta modulo n om $a - b$ är delbart med n. Med andra ord, om $a\ och\ b$ ger samma rest vid division med n. Kongruensen betecknas då på följande sätt:

$$a \equiv b \mod n$$

Detta kan även tolkas som:

$$a = b + kn \text{ för något heltal } k$$

Om a vid division med n ger resten r, så är $a \equiv r \mod n$. Givet ett tal a så finns precis ett tal r sådant att: $0 \leq r \leq n - 1$, sådant att $a \equiv r \mod n$, nämligen resten vid division med n [7, 2].

Sats: följande gäller för kongruenser:

a) $a \equiv a \mod n$ för alla a

b) Om $a \equiv b \mod n$ så är $b \equiv a \mod n$

c) Om $a \equiv b\ och\ b \equiv c \mod n$, så är $a \equiv c \mod n$

d) Om $a \equiv b \ och\, c \equiv d \ \ mod\, n$, så gäller $a \pm c \ \equiv b \pm d \ och\, ac \equiv bd \ \ mod\, n$

e) Om $a \equiv b$ så är $a^k \equiv b^k \ \ mod\, n$ för alla heltal $k \geq 0$

f) Om $sgd\ (a\ , n) = 1$, så finns ett heltal x sådant att:

$$ax \equiv 1 \ mod\, n$$

Bevis:

De tre första påståenden är mer eller mindre självklara. Om $a \equiv b, c \equiv d$ så kan vi skriva:

$$a = b + kn \ \text{för några heltal } k$$

$$c = d + In \ \text{för några heltal I}$$

Alltså är $a \pm c = b \pm d + (k \pm I)n$ och första delen av d) följer. Multiplikation ger i stället $ac = bd + (bI + dk + kIn)n$ varav $ac \equiv bd$. Upprepad användning av d) ger e). [7, 2]

I del f) har vi enligt Euklides algoritmen som beskrivs i mindre detaljer senare i kapitel 2.2, att det finns heltal $x\ och\ y$ sådant att $ax + ny = 1$ .Modulo n betyder detta att $ax \equiv 1$

Talet x brukar kallas för inversen $a\ modulo\ n$ och betecknas $x = a^{-1}$ (se 2.2). När k är ett positivt heltal gäller de vanliga potensreglerna och vi har: [7, 3]

$$a^{-k} = (a^{-1})^{k}$$

2.2 Euklides algoritm och dess användning

Euklides algoritmen som är nästan 2400 år gammal används framtill idag till för att räkna ut största gemensamma delaren mellan två tal. Med hjälp av Euklides algoritmen kan man ta reda på ifall två tal är relativt prima. Algoritmen är grundläggande inom RSA och framför allt vid inversberäkningar som är avgörande vid dekryptering av ett meddelande.

Sats:

Låt $a \geq b$ vara positiva heltal. Successiva divisioner leder till följande situation:

$$a = kb + r$$
$$b = k_1 r + r_1$$
$$r = k_2 r_1 + r_2$$
$$.$$
$$.$$
$$.$$
$$r_{m-2} = k_m r_{m-1} + r_m$$
$$r_{m-1} = k_{m+1} r_m$$

Då är SGD$(a,\ b) = r_m$ den sista icke-försvinnande resten. **[20, 76]**

Bevis: [20, 76]

Enligt division algoritmen har vi:

$$b > r > r_1 > r_2 > \cdots \geq 0.$$

Resten krymper i varje steg. Detta visar att algoritmen måste någon gång ta slut och består av maximalt b steg. Vi har då:

$$SGD(a,b) = SGD(b,r) = SGD(r,r_1) = \cdots = SGD(r_{m-1}, r_m) = r_m$$

Exempel:

Beräkna $27^{-1}\ (mod\ 392)$.

Observera att $a^{-1}\ mod\ n$ betyder inversen av $a\ mod\ n$. (se avsnitt 2.1)

Lösning:

Vid beräkningar $mod\ 392$ arbetar vi helt enkelt med faktorer som ligger i intervallet $[0\ , n - 1]$ där $n = 392$

Vi söker efter ett värde på x sådant att:

$$27.x\ (mod\ 392) \equiv 1$$

Vi börjar då med Euklides algoritmen:

$$392 = 27.(\ 14\ ) +\ 14$$

$$27 = 14\ .(\ 1\ ) + 13$$

$$14 = 13\ .(\ 1\ ) + 1$$

Om vi gör processen baklänges får vi:

$$14 + 13\ (\ -1) = 1 \qquad (1)$$

$$27 + 14\ (\ -1\ ) = 13 \qquad (2)$$

$$392 + 27\ (\ -14\ ) = 14 \quad (3)$$

Substitutionsmetoden och insättning av värdena från nästkommande ekvationen ger oss:

$$14 + [\ 27 + 14\ (-1)](-1) = 1 \quad (1)$$

$$14 + 27\ (-1) + 14 = 1$$

$$2\ (14) + 27(-1) = 1$$

Ersätter nu 14 med vad 14 stod för i ekvation 3 och får:

$$2[392 + 27(-14)] + 27(-1) = 1$$

$$2\ .392 + 27(-28) + 27(-1) = 1$$

$$2\ .392 + 27\ .(-29) = 1\ (mod\ 392)$$

Vad vi behöver göra är att konvertera -29 till ett tal som ligger i intervallet 0 till 391.

Om vi subtraherar 29 ifrån 392 får vi ett värde som är kongruent med $-29\ mod\ 392$.

$$392 - 29 = 363$$

Då kan vi ersätta -29 i ekvationen med 363.

$$2\ .392 + 27\ .363 = 1\ (mod\ 392)$$

$$27\ .363 = 1\ (mod\ 392)$$

$$27^{-1} = 363\ (mod\ 392)$$

Nu har vi beräknat inversen till 27 mod 392.

Egenskapen att kunna beräkna inversen till ett givet tal hos Euklides algoritmen är viktigt inom RSA krypteringsmetoden och framför allt när man försöker hitta den privata nyckeln för att kunna dekryptera meddelanden (se avsnitt 4.2)

2.3 Eulers sats och bevis

En annan sats som RSA kryptering grundar sig på är Eulers satsen. Satsen har fått sitt namn efter Leonhard Euler och är en generalisering av Fermats lilla sats (se 3.1). Satsen lyder på följande sätt:

Sats:

Låt p och q vara två distinkta primtal och $g = \text{sgd}\ (p-1, q-1)$

Då har vi:

$$a^{\frac{(p-1)(q-1)}{g}} \equiv 1\ mod\ pq$$ för alla a som uppfyller $sgd(a, pq) = 1$

Särskilt om p och q är udda primtal så har vi:

$$a^{\frac{(p-1)(q-1)}{2}} \equiv 1\ mod\ pq$$ för alla a som uppfyller $sgd(a, pq) = 1$. [4, 118]

Bevis: (Eulers sats)

genom antagandet vet vi att a inte är delbar med p och dessutom att g är ett tal som delar $(q-1)$. Då kan vi beräkna:

$$a^{\frac{(p-1)(q-1)}{g}} = \left(a^{(p-1)}\right)^{\frac{(q-1)}{g}}$$ eftersom $^{(q-1)}/_{g}$ är ett heltal

$$\equiv 1^{\frac{(q-1)}{g}}\ (mod\ p)$$ eftersom $a^{p-1} \equiv 1\ (mod\ p)$ enligt Fermats lilla sats.

$$\equiv 1\ \ (mod\ p)$$ eftersom 1 upphöjt vilket tal som helst blir lika med 1.

Med hjälp av precis likadana beräkningar med omvända roller hos p och q får vi att:

$$a^{\frac{(p-1)(q-1)}{g}} \equiv 1\ \ (mod\ q)$$

Detta bevisar i sin tur att $a^{\frac{(p-1)q-1)}{q}} - 1$ är delbart med både p och q; och därav även delbart med pq med tanke på att pq är produkten. Detta bevisar då satsen [4, 119].

3

Primtal och primtalstest

3.1 Primtalstest med Fermats lilla sats

Inom RSA kryptering har primtalen en viktig tillämpning och används som offentliga nycklar i kommunikationen mellan avsändaren och mottagaren. Primtalstest som är en process som avgör huruvida ett givet positiv heltal är dessutom primt, blir då ett centralt begrepp inom kryptering med tanke på att processen bekräftar primaliteten hos talen och detta är ett villkor som avgör huruvida talen får senare användas som offentliga nycklar. De utvalda primtalen kan vara upp till flera hundra siffror långa och detta innebär att primaliteten inte går att reda ut utan utförliga beräkningar och primtalstester.

Svårighetsnivån när man arbetar med primtalen beror helt och hållet på storleken av de givna talen och kan variera från relativt enkelt till helt omöjligt att avgöra utan hjälp av moderna datorn och det är inte heller säkert att datorn klarar av det [6,2].

Om man får i uppgift att avgöra huruvida ett givet tal n är primtal så räcker det med att avgöra ifall n är delbar med något primtal x som ligger i intervallet $2 < x < \sqrt{n}$.
Anledningen är att varje sammansatt tal n har en primtalsdelare $p \leq \sqrt{n}$. Detta gör det möjligt att avgöra till exempel ifall ett små tal som 127 är ett primtal genom att avgöra närliggande kvadratrötterna [6,1]

$$11^2 < 127 < 12^2$$

Då kan vi dra för slutsats att det räcker att testa delbarheten med primtalen $p < 11$ för att kunna inse att 127 faktiskt är ett primtal.
Däremot har ovanstående metod såklart ingen användning och är inte tidssparande på något sätt när man arbetar med tal som består av flera hundra siffror och det är just sådant stora tal

som används inom kryptering för att öka säkerheten. Då är Fermats lilla sats en stor hjälp om man arbetar med att generera nycklar inom RSA kryptering och formuleras enligt följande:

Sats:

Låt p vara ett primtal och a ett tal som inte är delbart med p. Då gäller att:

$$a^{p-1} \equiv 1 \ mod\ p$$

Bevis för Fermats lilla sats: [7, 5]

Vi låter a vara ett tal som inte är delbart med p. Vi skall betrakta de $p-1$ talen.

$$a \cdot 1, a \cdot 2, \dots, a \cdot (p-1)$$

Låt deras rester vid division med p vara $b_1, b_2, \dots, b_{p-1}$. Notera att:

$$1 \leq b_i \leq p-1$$

Och dessutom att:

$$ai \equiv b_i \ mod\ p$$

Vi skall först visa att alla $b_i : na$ är olika. För antag att:

$$b_i = b_j$$

Då är $ai \equiv aj \ mod\ p$ och vi har alltså: $p|a(i-j)$. Men p delar inte a, så $p|(i-j)$. varav följer $i = j$ eftersom både ligger i mellan 1 och $p-1$.

Då talen b_i är $p-1$ stycken till antalet och är olika, så måste de utgöra alla tal $1,2, \dots, p-1$ fast i en annan ordning.

Vi har att:

$$b_1 \cdot b_2 \cdot \dots \cdot b_{p-1} = 1 \cdot 2 \cdot \dots \cdot (p-1) = (p-1)!$$

Produkten av alla ai är $a^{p-1}(p-1)!$, så av $ai \equiv b_i \ mod\ p$ följer:

$$a^{p-1}(p-1)! \equiv b_1 \cdot \dots \cdot b_{p-1} \equiv (p-1)! \quad mod\ p$$

Det gäller alltså att $p|(a^{p-1}-1)(p-1)!$. Men p kan inte dela $(p-1)!$ Eftersom det skulle innebära att den delar någon av faktorerna vilka är alla $< p$.

Det följer då att:

$$\mathrm{p}|a^{p-1}-1$$

Med andra ord att $a^{p-1} \equiv 1 \quad mod\ p$

För att göra processen lite mer begripligt ska vi se ett exempel.

Exempel: Talet 15485863 är ett primtal. Fermats lilla sats säger då att:

$$2^{15485862} \equiv 1 \ (mod\ 15485863)$$

Då vet vi utan att behöva beräkna något att talet $2^{15485862} - 1$ som innehåller över två miljoner siffror är en multipel av 15485863. [4, 129]

Observation:

Om $a^{n-1} \not\equiv 1\ mod\ n$ kan vi dra slutsatsen att n måste vara ett sammansatt tal. Däremot räcker det inte $a^{n-1} \equiv 1 \ \ mod\ n$ som bevis på att ett tal är primtal [6, 2]. Detta beror på att relationen som Fermats lilla sats byggs på är ensidig. Trots att det inte finns något som förhindrar en ekvivalensrelation i praktik, så har man hittat exempel på tal som uppstår som motbevis i speciella fall.

Exempel:

$$2^{341-1} \equiv 1\ mod\ 341$$

Vi vet att $341 = 11 \cdot 31$ och är ett sammansatt tal, men Fermats sats gäller trots att 341 är icke-primt. Vi kan då identifiera 341 som ett tal som lurar Fermats test. Sådana tal kallas för ”Pseudoprimtal” [6, 2]. Det är inte tillräckligt att utföra Fermats test på liknande tal och detta leder till följande definition.

Definition:

låt n vara ett heltal. Vi säger att heltalen a tyder på att n är sammansatt om:

$$a^n \not\equiv a \ \ mod\ n$$

a kallas i så fall ett ”vittne” till sammansattheten” av n. Ett enda vittne till n tillsammans med kännedom till att:

$$a^p \equiv a \ \ (mod\ p) \text{ för alla heltal } a$$

Bevisar att n är sammansatt [4, 130].

Observation:

Sammansatta tal som saknar vittne kallas för Carmichaeltal. R. D. Carmichael var den första som publicerade ett papper kring Carmichaeltalen vid 1910, men lyckades endast lista ut 15 Carmichaeltal. Idag vet vi att sådana finns oändligt många av [4, 130].

Talet 561 är det minsta Carmichaeltal och är det första och minsta talet som leder till frågetecken kring effektiviteten hos fermats test [1, 4]. Miller-Rabins primtalstest baseras däremot på att alla sammansatta tal har ett stort antal vittnen och anses vara mer effektivt i jämförelse med fermats test [4, 131]. I det nästkommande avsnitt ska Miller-Rabin primtalstest beskrivas.

3.2 Miller-Rabins primtalstest

I ett försök för att hitta en lösning till problemet som orsakas av pseudoprimtalen när man använder fermats sats för primtalstestning, lyckades Gary Miller och Michael Rabins med att modifiera fermats test på ett sätt som kan till skillnad från Fermats test faktiskt bevisa att ett godtyckligt heltal som är positivt är sammansatt [8, 5]. Testet har egenskapen att göra det mycket osannolikt att ett icke-primtal skulle bedömas vara primt när det utförs med flera olika baser [8, 5].

Proposition 1:
Låt p vara ett udda primtal och att:

$$p - 1 = 2^k q \text{ där } q \text{ är udda}$$

Låt a vara ett godtyckligt tal som ej är delbart med p. Då är ett utav följande två villkoren uppfyllt:

(i) $a^q \equiv 1 \mod p$

(ii) Ett utav $a^q, a^{2q}, a^{4q}, \dots, a^{2^{k-1}q}$ är kongruent med -1 modulo p.

Bevis:
Vi vet redan enligt fermats lilla sats att:

$$a^{p-1} \equiv 1 \ (mod\ p)$$

Detta innebär att det sista talet i nedanstående serie som är lika med a^{p-1} är kongruent med 1 modulo p.

$$a^q, a^{2q}, a^{4q}, \dots, a^{2^{k-1}q}, a^{2^k q}$$

Dessutom är varje tal i listan kvadraten av det föregående talet och då måste en utav följande två möjligheter inträffa:

(i) första talet i listan är kongruent med 1 modulo p

(ii) något tal i listan är inte kongruent med 1 modulo p. Men det blir kongruent med 1 modulo p när den kvadreras.

Det enda talet som uppfyller både villkoren:

$$b \not\equiv 1 \ (mod\ p) \quad \text{och } b^2 \equiv 1 \ (mod\ p)$$

Är -1.

Då måste ett utav talen i listan vara kongruent med -1 modulo p. Vilket skulle bevisas [4, 130].

Miller-Rabin algoritm:

För att kunna begripa strukturen bakom testet och implementera algoritmen måste vi förstå innebörden av gruppen Z_n^*. Beteckningen står för ett antal element $|a|_n \in z_n$, där a är ett heltal relativt primt till n [10, 239].

Antag att $n = {p_1}^{e_1} \dots {p_r}^{e_r}$ betecknar primfaktoriseringen av n. Då kan vi med hjälp av den kinesiska restsatsen få följande ring isomorfin:

$$\mathbb{Z}_n \cong \mathbb{Z}_{{p_1}^{e_1}} \times \dots \times \mathbb{Z}_{{p_r}^{e_r}}$$

Detta framkallar då följande gruppisomorfin:

$$\mathbb{Z}_n^* \cong \mathbb{Z}_{{p_1}^{e_1}}^* \times \dots \times \mathbb{Z}_{{p_r}^{e_r}}^*$$

Således, för att bestämma strukturen av gruppen $\mathbb{Z}_n^*$, räcker det att avgöra hur strukturen på $n = p^e$ ser ut, när p är primt.

Dessutom har vi att gruppen $\mathbb{Z}_{p^e}^*$ har en ordning som beräknas på följande sätt:

$$\phi(p^e) = p^{e-1}(p-1)$$

Detta stämmer i allmänhet för ett primtal p och ett positivt heltal e och kan motiveras med hjälp av följande bevis:

Bevis:

Vi har i intervallet $[0, p^e - 1]$, att multiplar av p är följande:

$$0.p, 1.p, \dots (p^{e-1} - 1).p,$$

Och sådana existerar precis p^{e-1} många av.

Således har vi:

$$\phi(p^e) = p^e - p^{e-1} = p^{e-1}(p-1)$$

Vilket skulle bevisas [10, 27].

Antag att heltalen som ska genomgå Miller-Rabins primalitetstest är udda. Då kan den allmänna strukturen hos testet se ut på följande sätt:
Låt $z_n^{\neq}$ vara en mängd nollskilda element z_n. Då är $|z_n^{\neq}| = n - 1$. Det gäller även att $z_n^{\neq} = z_n^*$ om n är primtal.

Antag att vi har en mängd $l_n \subseteq z_n^{\neq}$. Då gäller följande:

- Det existerar en effektiv algoritm som har förmågan att endast baserad på input n, och kunskap om att $\alpha \in z_n^{\neq}$, avgöra huruvida $\alpha \in l_n$
- Ifall n är primtal har vi: $l_n = z_n^*$
- Ifall n är sammansatt har vi: $|l_n| \leq c\,(n-1)$ för en del konstant $c < 1$.

För att avgöra primaliteten av n med hjälp av algoritmen, använder man sig av ett system som producerar sant/falskt som output. Algoritmen går ut på att bestämma en ”error parameter” t och välja därefter olika värde på α från $\alpha_1, \dots, a_t \in z_n^{\neq}$. Om $\alpha_i \in l_n$ för $i = 1, \dots, t$, får man då sant som output. Annars får man tillbaka falskt [10, 241].

Om n är primtal kommer algoritmen alltid producera Sant som output. Detta tyder då på att $l_n = z_n^*$. Däremot förväntar man sig att få falskt som output för icke-prima n. Det är värt att notera att algoritmen inte är felfritt. Sannolikheten att få sant som är felaktigt för en sammansatt n, är nästan försumbart dock inte obefintligt.
Detta kan motiveras enkelt på följande sätt:
Om man har två möjliga output (falskt och sant), och utför processen tillräckligt många gånger (exempelvis 100 försök), minskar sannolikheten att få felaktig output efter varje försök. Detta beror på att var och en utav alternativen har en sannolikhet på 50% till att börja med som sjunker ner till 25% redan efter nästkommande försöket och minskar därefter med samma fart. man har efter 100 försök enligt följande:
Högsta sannolikheten: c^t där t betecknar antal försök

$c = \frac{1}{2}, t = 100$

$$\frac{1}{2}^{100} = \frac{1}{2^{100}}$$

Utförandet av bråkmultiplikationer ger oss slutligen ett värde som blir nästan lika med noll men går inte att bortse ifrån helt och hållet. Detta motsvarar då sannolikheten av att få ett felaktigt output efter 100 försök [10, 242].

Miller-Rabins idé byggs på antagandet att för alla sammansatta tal existerar ett stort antal vittnen [4]. Detta definieras på följande sätt.

Definition:

Låt n vara ett udda tal och att:

$$n - 1 = 2^k q \quad \text{där q är udda}$$

Ett heltal a som uppfyller villkoret $\text{sgd}(a, n) = 1$ kallas då för ett Miller-Rabins vittne till sammansattheten av n om följande två villkoren är uppfyllda:

(i) $a^q \not\equiv 1 \pmod{n}$

(ii) $a^{2^i q} \not\equiv -1 \pmod{n}$ för alla $i = 0,1,2, \dots, k-1$.

Det följer då från proposition 1 att om det existerar ett a sådant att a är en Miller-Rabins vittne till n, då är n sammansatt [4, 131].

Proposition 2:

Låt n vara ett udda sammansatt tal. Då är minst 75% av alla tal a i intervallet 1 till $n-1$ Miller-Rabins vittne till n. [4, 131]

Om Bob får i uppgift att identifiera stora primtal, tar han då ett misstänkt primtal n och utför Miller-Rabins testet med tio olika värden på a. Om det visar sig att något a är ett Miller-Rabins vittne, är n sammansatt [4, 132]. Antag nu att inga värden på a visar sig vara ett vittne till n efter 10 försök trots det som sägs i proposition 2. Då kan man dra slutsatsen om att sannolikheten för sammansattheten av n är så litet som $(\frac{1}{4})^{10}$. Detta är lika med $\frac{1}{1048576}$ som motsvarar ungefär $0{,}0000953674$ %.

Exempel: vi ska nu illustrera Miller-Rabins test i praktik.

Vi har: $a = 2,\ n = 561$

Vi får följande genom att faktorisera:

$$n - 1 = 560 = 2^4 \cdot 35$$

Och beräknar därefter:

$$2^{35} \equiv 263 \pmod{561}$$

$$2^{2.35} \equiv 263^2 \equiv 166 \pmod{561}$$
$$2^{4.35} \equiv 166^2 \equiv 67 \pmod{561}$$
$$2^{8.35} \equiv 67^2 \equiv 1 \pmod{561}$$

Det första talet $2^{35} \mod 561$ är varken 1 eller -1. Resterande talen är inte heller lika med -1. Då är 2 ett Miller-Rabins vittne till att 561 är sammansatt. [4, 132]

Bevis till proposition 2: [hämtad ifrån 11]

Tidigare under proposition 2 fick vi acceptera att minst 75% av alla tal a i intervallet 1 till $n-1$ är Miller-Rabins vittne till n. Med andra ord, de tyder på sammansattheten av n. Detta påstående ska nu bevisas.

Detta anses vara svårt att bevisa med tanken på att gruppen av Miller-Rabins icke-vittne inte är sluten under multiplikation. Med andra ord, multiplikation mellan två delmängder garanterar inte ännu ett vittne till [11, 8].

Propositionen kan även tolkas som att antalet heltal som inte är ett Miller-Rabins vittne inom samma intervall kan inte överstiga 25%.

Detta kan vi göra genom att bevisa att 25% är faktiskt en övre begränsning hos gruppen som består av Miller-Rabins icke-vittnen för alla n som ligger i intervallet $\{1, \dots, n-1\}$.

Då har vi för W som betecknar antalet Miller-Rabin vittne för n i intervallet $\{1, n-1\}$:

$$\frac{W}{n-1} \geq \frac{3}{4}$$

Vi har tre olika fall som behöver undersökas. Denna indelning görs baserad på antal primfaktorer som ingår i uppbyggnaden av heltalen.

Fall 1:

$n = p^{\alpha}$ är primpotensen till ett udda primtal och $\alpha \geq 2$.

För att göra första fallet mer begripligt, erbjuds nu följande sats:

Sats:

Om $n = p^{\alpha}$ för ett primtal p och $\alpha \geq 1$, har vi att samtliga lösningar till $\alpha^{p-1} \equiv 1 \, mod \, p^{\alpha}$ är Miller-Rabins icke-vittne till n. Dessa lösningar etablerar då en sluten grupp vid multiplikation mod n [11, 6].

Bevis:

Låt $a \in \{1, \dots, n-1\}$ vara en Miller-Rabin icke-vittne. Med tanke på att a är relativt primt till $n = p^{\alpha}$, har vi enligt Eulers satsen att:

$$a^{\varphi(n)} \equiv 1 \ mod\, n$$

Dessutom har vi för en icke-vittne att en utav följande stämmer:

$$a^{k} \equiv 1 \ mod \ n$$

Eller

$$a^{2_i k} \ \equiv \ -1 \ mod\, n \ \text{ för } i \leq e-1$$

Bägge fallen medför att:

$$a^{2^e k} \equiv 1 \ mod\, n \ \Leftrightarrow \ a^{n-1} \equiv 1 \ mod\, n$$

Således delas $(\varphi(n), n-1) = (p^{\alpha-1}(p-1), p^{\alpha}-1)$ av ordningen på $a \ mod\, n$.

Med tanke på att p är relativt primt med $p^{\alpha}-1$, och $p-1$ delar $p^{\alpha}-1$ har vi:

$$(p^{\alpha-1}(p-1), p^{\alpha}-1) = p-1$$

Då måste $a^{p-1} \equiv 1 \ mod\, p^{\alpha}$ stämma.

Ovanstående relation gäller även omvänt.

Antag att:

$$a^{p-1} \equiv 1 \ mod\, p^{\alpha}$$

Låt:

$p-1 = 2^{f}\tau$,där $f \geq 1$ och τ är udda.

Med tanke på att $p-1$ är en faktor till $p^{\alpha}-1 = 2^{e}k$ har vi:

$$f \leq e, \qquad \tau \mid \mathrm{k}$$

Då $(a^{\tau})^{2^{f}} \equiv 1 \ mod\, p^{\alpha}$, har vi att ordningen på $a^{\tau} \ mod\, p^{\alpha}$ blir då 2^{j} för $j \in \{0, \dots, f\}$.

Om $j = 0$, har vi att $a^{\tau} \equiv 1 \ mod\, p^{\alpha}$. Då är även $a^{k} \equiv 1 \ mod \ p^{\alpha}$ då vi redan vet att $\tau \mid$ k sedan tidigare.

Om $j \geq 1$, har vi att:

$x := (a^{\tau})^{2^{j-1}}$ uppfyller $x \not\equiv 1 \ mod\, p^{\tau}$ medan $x^2 \equiv 1 mod\, p^{\tau}$.

Således har vi:

$p^{\tau} \mid (x-1)(x+1)$ och termerna $(x-1)$ och $(x+1)$ kan inte vara näst intill varandra. Detta kan tolkas som att endast en utav dessa två är delbar med p och det talet måste då innehålla alla faktorer i p^{τ}.

Vi har då med andra ord att $p^\tau|(x+1)$, eller $p^\tau|(x-1)$. Detta ger då:

$$x \equiv \pm 1 \ mod \ p^\tau$$

Vi visade redan tidigare att $x \not\equiv 1 \ mod \ p^\tau$. Då gäller det endast att $x \equiv -1 \ mod \ p^\tau$.
Om vi då ersätter x med vad den representerar får vi:

$$a^{2^{j-1}e} \equiv -1 \ mod \ p^\alpha$$

Då vi har τ | k och k är udda, kan vi genom att utveckla både sidorna upphöjt till k/τ få enligt följande:

$a^{2^i k} \equiv -1 \ mod \ p^\alpha$ där $i = j-1 \in \{0, \dots, f-1\} \subset \{0, \dots e-1\}$. □

Vi återvänder till fallet nu.

Vi hade att alla lösningar till $a^{p-1} \equiv 1 \ \ (mod \ p^\alpha)$ var icke-vittne till p^α. Frågan som ställs nu efter att detta har bevisats, är då hur många sådana a finns det i intervallet $[1, p^\alpha - 1]$.
I samband med att besvara ovanstående frågan ska vi undersöka fallet med ett exempel.

Om vi har: $p = 5, 7$, och små värde på α,
Så ser lösningar till $a^{p-1} \equiv 1 \ \ (mod \ p^\alpha)$ enligt följande tabellen:

α	Lösningar till $a^4 \equiv 1 \ mod \ 5^\alpha$	Lösningar till $a^6 \equiv 1 \ mod \ 7^\alpha$
2	1,7,18,24	1,18,19,30,31,48
3	1,57,68,124	1,18,19,324,325,342
4	1,182,443,624	1,1047,1048,1353,1354,2400

Genom att undersöka tabellen kan man misstänka att $a^{p-1} \equiv 1 \ \ mod \ p^\alpha$ har $p-1$ antal lösningar mod p^α för alla värden på α. Detta kan bekräftas även för $\alpha = 1$ med hjälp av Fermats lilla sats (se 3.1). För större värde på α kan man använda induktion på följande sätt:
Om $a^{p-1} \equiv 1 \ \ (mod \ p^\alpha)$, finns det då ett unikt $a' \ \ mod \ (p^{\alpha+1})$ sådant att:

$$a'^{\,p-1} \equiv 1 \text{ mod } \ p^{\alpha+1} \text{ ,och } \ a' \equiv a \ \ \text{mod } p^\alpha$$

Att påstå $a' \equiv a \ \ \text{mod } p^\alpha$ är samma som att påstå $a' \equiv a + cp^\alpha \ \ \text{mod } p^{\alpha+1}$. Med det sagt kan vi då påstå att vi ska bevisa att ett unikt val av $c \ \ mod \ p$, måste finnas så att:

$$(a + cp^\alpha)^{\,p-1} \equiv 1 \ \ (mod \ p^{\alpha+1})$$

Med hjälp av binomialsatsen får vi:

$$(a + cp^{\alpha})^{\,p-1} \equiv a^{p-1} + (p-1)a^{p-2}cp^{\alpha} \pmod{p^{\alpha+1}}$$

Där termer med högre grad försvinner med tanke på att $p^{r\alpha} \equiv 0 \mod p^{\alpha+1}$ för alla $r \geq 2$.

Vi har för något heltal $M \in \mathbb{Z}$ att $a^{p-1} \equiv 1 + p^{\alpha}M$. Med andra ord. så söker vi C som uppfyller följande kongruensen:

$$(1 + p^{\alpha}M) + (p-1)a^{p-2}cp^{\alpha} \equiv 1 \bmod p^{\alpha+1},$$

Som är i sin tur ekvivalent med:

$$M - a^{p-2}c \equiv 0 \bmod p$$

Och detta har då en unik lösning för c modulo p med tanke på att a är inverterbar modulo p.

Tidigare visade vi att det finns $p-1$ Miller-Rabins icke-vittne till p^{α} i intervallet $\{1, \dots, p^{\alpha} - 1\}$, då har sådana tal följande andel inom intervallet:

$$\frac{p-1}{p^{\alpha}-1} = \frac{1}{1 + p + \cdots + p^{\alpha-1}}$$

Med tanke på att $\alpha \geq 2$, ovanstående förhållande kan högst vara $\frac{1}{(1+p)}$. Detta kan som högst vara $\frac{1}{1+3} = \frac{1}{4}$ och denna lösning erhålls endast om $\alpha = 2 \; och \; p = 3$. Förhållandet för alla andra möjliga värden på p^{α} blir mindre än $\frac{1}{4}$.

$p^{\alpha} = 9$ ger då största förhållandet i intervallet och även detta följer tumregeln som vi försöker bevisa. Vi har nu bevisat att 25% faktiskt är en övre begränsning för element som kategoriseras inom första fallet.

Fall 2:

Fokuset ska nu skiftas mot alla n som har i stället två olika primfaktorer i sin uppbyggnad.

Vi har

$n - 1 = 2^{e}k, e \geq 1$ och k är udda.

Vi låter i_0 vara största heltalen i intervallet $\{0,1, \dots, e-1\}$ så att vissa a_0 uppfyller $(a_0, n) = 1$.

Vi har dessutom att $i_0 > 0$, och att följande är en grupp som innehåller alla Miller-Rabins icke-vittnen under multiplikation med n.

$$G_n = \left\{1 \leq a \leq n-1 \colon \; a^{2^{i_0}K} \equiv \pm 1 \bmod n\right\}$$

Gruppen är dessutom en delmängd av alla inverterbara heltalen modulo n.

Bevis: [11, 7]

$$G_n = \left\{1 \le a \le n-1:\ a^{2^{i_0}K} \equiv \pm 1\ mod\ n\right\}$$

Är en grupp under multiplikation modulo n och vi ska visa att mängden innehåller alla a i intervallet $\{1, \dots, n-1\}$, som uppfyller ett av följande kraven:

1) $a^k \equiv 1\ mod\ n$
2) $a^{2^i k} \equiv -1\ mod\ n$ för något $i \in \{0, \dots, e-1\}$

Om 1 är uppfylld för a har vi att:

$$a^{2^{i_0}\,k} \equiv 1\ mod\ n$$

Om 2 är uppfylld för a har vi att:

$$(a^k)^{\,2^i} \equiv -1\ mod\ n$$

Och då är $i \le i_0$ med tanke på maxvärdet som i_0 kan anta. (betrakta $a_0 = a^k$)
Alltså om $i = i_0$ får vi:

$$a^{2^{i_0}k} \equiv -1\ \ mod\ n$$

Dessutom får vi om $i < i_0$,genom kvadrering av bägge sidor av$(a^k)^{\,2^i} \equiv -1\ \ mod\ n$ tillräckligt många gånger att:

$$a^{2^{i_0}k} \equiv 1\ \ mod\ \ n$$

Både fallen tyder dock på att: $a \in G_n$

Vi ska nu visa att G_n är en delgrupp av inverterbara tal modulo n.
Låt p vara en primfaktor av n
Och låt:

$$n = p^\alpha n' \text{ där } \alpha \ge 1 \text{ och } n' \nmid p$$

Både p^α och n' är då udda och $\neq 1$. Detta beror på att n inte är en primpotens. Då kan p^α och n' som minst vara 3.
Enligt den kinesiska restsatsen, existerar några $a \in \{1, \dots, n-1\}$ som uppfyller följande två kongruenser:

$$a \equiv a_0\ mod\ p^\alpha\ \ ,\quad a \equiv 1\ mod\ n'$$

Med tanke på att $(a_0, n) = 1$ får vi:

$$(a, n) = 1$$

Om vi då utför beräkningarna modulo p^α och därefter n' har vi:

$$a^{2^{i_0}k} \equiv a_0^{2^{i_0}k} \equiv -1^k \equiv -1 \ mod\, p^\alpha \Longrightarrow a^{2^{i_0}k} \not\equiv 1\, mod\, n$$

Detta beror på att:

$$-1 \not\equiv 1\, mod\, p^\alpha$$

Och dessutom:

$$a^{2^{i_0}k} \equiv 1\, mod\, n' \Longrightarrow a^{2^{i_0}k} \not\equiv -1\, mod\ \ n'$$

Beroende på att:

$$-1 \not\equiv 1\, mod\, n'$$

Därmed kan följande slutsats dras:

$$a^{2^{i_0}k} \not\equiv \pm 1\, mod\, n.$$

Då har vi att: $(a, n) = 1$ och $a \notin G_n$. □

Det gäller då att förhållandet $\frac{\varphi(n)}{|G_n|}$ är ett heltal och att $\varphi(n) < n - 1$ då n är icke-primt.

Vi ska visa om n inte är en primpotens, att $\frac{\varphi(n)}{|G_n|} \leq 4$. Då har vi:

$$\frac{antal\ icke-vittnen\ i\ intervallet}{n-1} < \frac{\varphi(n)}{|G_n|} \leq \frac{1}{4}$$

vi vill visa att alla $a \in G_n$ uppfyller $a^{n-1} \equiv 1 \ \ mod\, n$. Med tanke på att $i_0 \leq e - 1$, måste produkten $2^{i_0+1}k$ dividera $2^e k = n - 1$. Alla $a \in G_n$ uppfyller då $a^{2i_0k} \equiv \pm 1 \ \ mod\, n$. Kvadrering av detta ger oss:

$$a^{2^{i_0+1}k} \equiv 1 \ \ mod\, n$$

Och därmed har vi även att:

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

Fall 3:

n har minst 3 primfaktorer.

Observera att bland alla n som har 3 primfaktorer har vi även Carmichaeltalen. Då vi redan vet att sådana tal inte kan vara Miller-Rabins vittne, är det viktigt att dela upp fallet i två delar.

Del 1: n som inte är Carmichaeltal.

Låt:

$$F_n = \{1 \leq a \leq n - 1 \colon a^{n-1} \equiv 1 \ \ mod\, n\}$$

Då gäller följande relation:

$$\{1 \leq a \leq n-1 : (a,n) = 1\} \supset F_n \supset G_n$$

Alla tre mängder är grupper under multiplikation modulo n. Vi ska visa att relationen mellan givna mängder stämmer.

Genom gruppteorin har vi:

$$\frac{\varphi(n)}{F_n} \geq 2 \text{ och } \frac{F_n}{G_n} \geq 2.$$

Därmed har vi att:

$$\frac{\varphi(n)}{G_n} = \frac{\varphi(n)}{F_n}\,\frac{F_n}{G_n} \geq 4$$

Om n inte är ett Carmichaeltal har vi då vissa heltal som är relativt prima till n, inte är delmängder i F_n. Då stämmer första relationen.

För att visa att $F_n \neq G_n$, det vill säga den andra relationen, måste vi välja en primfaktor av n som betecknas med p, och påbörja processen genom att låta $n = p^{\alpha}n'$ där $\alpha \geq 1$ och p delar inte n'. Då är $n' > 1$.

Heltalen $a \in \{1, \dots, n-1\}$ är då som vi visade tidigare, inte en delmängd i G_n. Beviset ger oss även att $a \in F_n$.

Utifrån följande 2 kongruenser:

$$a^{2^{i_0}k} \equiv -1 \ mod\, p^{\alpha}$$

$$a^{2^{i_0}k} \equiv 1 \ mod\, n'$$

Får vi då eftersom kongruensen uppfylls modulo p^{α} och n' att:

$$a^{2^{i_0+1}k} \equiv 1 \ mod\, n$$

Således har vi då med tanke på att $a^{2^{i_0+1}k}$ är en faktor till $n-1$ att:

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

Del 2: n har minst 3 olika primfaktorer.

Vi börjar med att beteckna nedbrytningen av primfaktorer av n som $p^1\alpha^1, \dots, p^r\alpha^r$ för distinkta primtalen p_ι, där exponenter $\alpha_\iota \geq 1$ och $r \geq 3$.

Låt:

$$H_n = \left\{1 \leq a \leq n-1 : a^{2i_0k} \equiv \pm 1 \ mod\, {p_\iota}^{\alpha_\iota} \ för\ \iota = 1, \dots r\right\}$$

Då gäller följande:

$$\{1 \leq a \leq n-1 : (a,n) = 1\} \supset H_n \supset G_n$$

Det som ska bevisas är då följande:

$$\frac{|H_n|}{|G_n|} \geq 4$$

Så att:

$$\frac{(\varphi_n)}{|G_n|} = \frac{(\varphi_n)}{|H_n|}\frac{|H_n|}{|G_n|} \geq \frac{|H_n|}{|G_n|} \geq 4$$

För heltalen x och y gäller det att:

$$x \equiv y \mod n \Leftrightarrow x \equiv y \mod {p_\iota}^{\alpha_\iota} \text{ för } \iota = 1, \dots r$$

Avbildningen mellan grupperna $f = H_n \to \prod_{e=1}^{r}\{ \pm 1 \mod {p_\iota}^{\alpha_\iota} \}$ som är definierad med $f(a \mod n) = (\dots, a^{2^{i_0}k} \mod {p_\iota}^{\alpha_\iota}, \dots)_{e=1}^{r}$ är en homomorfi. Det vill säga bevarar sin struktur.

Om vi låter $K_n = \ker f$, har vi: $H_n \supset G_n \supset K_n$. Och målgruppen av f har ordningen 2^r, ska vi bevisa att f är surjektiv. Med tanke på att f är en homomorfi, räcker det att visa för alla r-tupler som innehåller -1 i en utav sina komponenter i alla andra komponenter att de finns med i avbildningen av f. Beroende på symmetrin, räcker det att endast visa att $(-1,1,1, \dots 1)$ finns med i avbildningen.

Med det sagt, så är vi ute efter ett $a \in H_n$ sådant att:

$$a^{2^{i_0}k} \equiv \begin{cases} -1 \mod {p_1}^{\alpha_1} \\ 1 \mod {p_\iota}^{\alpha_\iota} \text{ där } \iota \geq 2 \end{cases}$$

Definitionen av ι_0 föreslår att det finns ett heltal a_0 sådant att ${a_0}^{2^{i_0}} \equiv -1 \mod n$. Baserad på kinesiska restsatsen vet vi även att det finns ett $a \in \{1, \dots, n-1\}$ sådant att:

$$a \equiv a_0 \mod {p_1}^{\alpha_1}, \qquad a \equiv 1 \mod {p_\iota}^{\alpha_\iota} \text{ för alla } \iota \geq 2$$

Då har vi:

$$a^{2^{i_0}k} \equiv {a_0}^{2^{i_0}k} \equiv (-1)^k \equiv -1 \mod {p_1}^{\alpha_1}$$

Och:

$$a^{2^{i_0}k} \equiv 1 \mod {p_\iota}^{\alpha_\iota} \text{ för alla } \iota \geq 2$$

Avbildningen $f(H_n)$ har ordningen 2^r och $f(G_n) = \{(1,1, \dots,1), (-1,-1, \dots, -1)\}$ har ordning 2. Därmed är $\frac{|H_n|}{|K_n|} = 2^r$ och är $\frac{|G_n|}{|K_n|} = 2$. Utifrån detta kan vi beräkna följande:

$$\frac{|H_n|}{|G_n|} = 2^{r-1}$$

Och detta blir minst 4 med tanke på att vi har $r \geq 3$.

Detta kompletterar beviset för alla heltal n oberoende av antalet primfaktorer de har i sin uppbyggnad att antalet Miller-Rabins icke-vittne inom intervallet $\{1, n-1\}$, kan aldrig överstiga 25%. Motsatsen ger då att minst 75% av alla tal inom samma intervall är Miller-Rabins vittne. Vilket skulle bevisas. [11,10] □

4

RSA i teori

4.1 Bakgrund

RSA metoden är känd för att vara den första algoritmen som använder asymmetrisk kryptering och detta innebär att metoden erbjuder möjligheten att använda två separata nycklar till att kryptera och dekryptera meddelanden. RSA grundar sig i princip på att det är lätt att multiplicera två primtal men däremot extrem tidskrävande och svårt att faktorisera ett godtyckligt givet tal i sina primfaktorer om de ingående primtalen är tillräckligt stora [3, 1]. Till och med idag saknar vi effektiva metoder som kan möjliggöra faktorisering av heltalen i sina primfaktorer.
Om man tar två godtyckliga relativt stora primtal som till exempel 7907 och 7919, är det enkelt att beräkna produkten. Däremot skulle faktorisering av produkten vara en utmaning och det kräver i princip att man går igenom alla tänkbara primtalsfaktorer tills man hamnar på talet 7907 [3, 1]. Om man nu valde att faktorisera ett tal av storleken 10^{130} , så skulle det kräva ungefär 5år för en modern pc att lösa problemet [3, 1]. Detta kan tas som ett tecken på säkerheten som RSA byggs på och förklarar skillnaden mellan RSA och alla symmetriska krypteringsmetoder som tidigare i historien användes för att upprätthålla säkerheten.

Sedan RSA-metoden för första gången publicerades i 1977 så har en hel del forskare ställt frågor kring säkerhetsnivån och analyserat huruvida systemet är sårbar när det befinner sig under attack. Trots att metoden har fått stå emot en hel del under senaste 20 åren, kan man påstå att inget har haft en förödande påverkan och angreppen har endast lyckats illustrera farorna med felaktig användning av RSA [2, 212]. Detta innebär att det är avgörande att implementera systemet på ett säkert sätt för att kunna leva upp till maximala säkerhetsnivån som erbjuds [2, 212].

RSA-systemet används även idag inom stora områden och bland annat kommersiella system. Metoden används av webbservrar och webbläsare för att säkra webtrafik, för att säkerställa integriteten av e-posten, för att säkra fjärrinloggningssessioner och är dessutom avgörande inom elektroniska kreditkortsbetalningssystem. RSA används framför allt inom applikationer där säkerheten för digitalt data kan vara en utmaning [2, 203].

Sammanfattningsvis kan man påstå att krypteringsmetoder som använder sig av distinkta nycklar, är i teorin baserat på svårlösta matematiska frågor såsom faktorisering av stora heltal, som är i och för sig nästan omöjliga att lösa. Exempelvis har vi RSA metoden som är baserad på faktorisering av stora heltal med upptill flera hundra siffror som kräver i princip flera år för en dator att beräkna. Men mottagaren inom RSA metoden har förmågan att lösa denna uppgift snabbare och smidigare med hjälp av sin privata nyckel. Med andra ord, använder mottagaren sin privata nyckel som ett verktyg som möjliggör utförandet av en uppgift som ska mer eller mindre vara omöjlig även för moderna datorer att hantera.

I detta kapitel ska vi undersöka grundläggande teorin som ligger bakom RSA metoden innan vi sätter i gång och undersöker hur metoden fungerar i praktik och med hjälp av datorer.

4.2 Kryptering och dekryptering med RSA

RSA algoritmen går ut på följande:
Om vi väljer två stora primtal p och q kan vi kalla produkten av dessa två för $p.q = N$ och produkten N används därefter av sändaren som en utav de offentliga nycklarna för att skicka meddelandet till en tilltänkt mottagare. Den andra offentliga nyckeln som kallas för e beräknas med hjälp av Eulers φ-funktion. Innan vi förklarar hur dessa nycklar används inom RSA, ska vi försöka definiera φ-funktionen:

Definition:

För ett naturligt tal n låter vi $\varphi(n)$ beteckna antalet heltal k sådana att $1 \leq k \leq n$ och $\gcd(k,n) = 1$. Funktionen φ kallas Eulers $\varphi -$ funktion. Om p är ett primtal så är $\varphi(p) = (p-1)$ och vi kan då bestämma $\varphi(p^m)$ [9, 48].

Detta bestäms genom att ta hänsyn till följande:

$$sgd\ (k, p^m) = 1 \Leftrightarrow p \text{ delar inte } k$$

Det gäller då för de tal k som har ett största gemensamma delare med p^m som är större än 1 att de är delbara med p. Med tanke på att vart p:te tal är delbart med p, kan vi bestämma antalet tal som är delbara med p på följande sätt:

$$\frac{p^m}{p} = p^{m-1}$$

Varav i sin tur följer att:

$$\varphi(p^m) = p^m - p^{m-1} = p^{m-1}(p-1)$$

Eulers $\varphi -$funktion har följande egenskaper:

- Om p är primtal, då är $\gcd(p,q) = 1$ för alla $1 \leq q < p$. Då har vi att:
$$\varphi(p) = p - 1$$
- Om p är primtal, och $k \geq 1$, då finns det precis $\frac{p^k}{p}$ tal mellan 1 och p^k som är delbara med p. Detta ger oss då:
$$\varphi(p^k) = p^k - p^{k-1}$$
- Om a och b är relativt prima, har vi att:
$$\varphi(ab) = \varphi(a) \cdot \varphi(b)$$
- $a\ och\ b$ som inte är prima, har alltid ett största gemensamma delare c enligt följande:
$$\varphi(ab) = \varphi(a).\varphi(b).\frac{c}{\varphi(c)}$$

Nyckel e har 2 kriterier som måste uppfyllas;

e ska ligga i intervallet $0 < e < \varphi(n)$ och $\varphi(n)$ räknas då ut med formeln:

$$\varphi(pq) = \varphi(n) = (p-1)(q-1)$$

Likheten ovan säger att $\varphi(pq) = \varphi(p) \cdot \varphi(q)$ och detta kan förklaras med hjälp av följande lemma:

Lemma:

Om $\text{sgd}(p, q) = 1$, så gäller $\varphi(pq) = \varphi(p) \times \varphi(q) = (p-1)(q-1)$ om p och q är relativt prima.

Bevis: [12, 66]

För att kunna se att $\varphi(pq) = \varphi(p) \times \varphi(q)$

Antag att uppsättningen av positiva heltalen mindre än n är: $\{1, \dots, (pq-1)\}$

Heltalen som tillhör ovanstående uppsättning och är relativt prima till n tillhör antingen $\{p, 2p, \dots, (q-1)p\}$ eller $\{q, 2q, \dots, (p-1)q\}$. Detta beror på att alla heltal som är relativt prima med n, måste dela antingen p eller q. Med det sagt, kan man påstå att alla heltal som innehåller antingen p eller q som en faktor i sig, är relativt prima med n [12, 66].

Notera att ovanstående uppsättningar inte är överlappande. Med andra ord, inga faktorer som tillhör den första mängden kan vara multiplar av q och samma gäller även för element hos den andra mängden i relation till p. Detta beror på att p och q är prima. Då kan antalet unika heltal i var och en av mängderna vara $(p-1) + (q-1)$

Då har vi:

$$\begin{aligned}\varphi(n) &= (pq-1) - [(q-1) + (p-1)] \\ &= pq - (p+q) + 1 \\ &= (p-1) \times (q-1) \\ &= \varphi(p) \times \varphi(q)\end{aligned}$$

Exempel 1: Beräkna $\varphi(21)$.

Lösning: $\varphi(21) = \varphi(3) \times \varphi(7) = (3-1) \times (7-1) = 2 \times 6 = 12$

Sats:

För små heltal är det möjligt att använda ovanstående beräkningar. Däremot kan $\varphi(n)$ ibland vara betydligt större och då är det mer lämpligt att använda följande formel:

$\varphi(n) = n \prod(1 - \frac{1}{p_j})$,

Där p_j står för primfaktorer av n [hänvisas till 10, 26 för bevis].

Exempel:

Beräkna $\varphi(2100)$.

Lösning:

Vi väljer talen 75 och 28. Vi har:

$$\varphi(2100) = \varphi(75 \times 28) = \varphi(75) \times \varphi(28)$$
$$= 75 \prod(1 - \frac{1}{p_j}) \times 28 \prod(1 - \frac{1}{p_j})$$
$$= \left[75\left(1 - \tfrac{1}{3}\right)\left(1 - \tfrac{1}{5}\right)\right] \times \left[28\left(1 - \tfrac{1}{2}\right)\left(1 - \tfrac{1}{7}\right)\right] = 40 \times 12 = 480$$

Nu ska vi återgå till krypteringsprocessen med en betydligt djupare förståelse av φ-funktionen.

Tillverkning av nycklar:

Som tidigare nämndes, är RSA metoden för kryptering baserad på användning av två distinkta nycklar, Den första nyckeln är produkten av två godtyckliga primtal p och q och kallas för n. Storleken av produkten motsvarar storleken på nyckeln n och uttrycks i bitar hos datorer (se 5,1). Den andra nyckeln, som kallas för e har 2 kriterier som måste uppfyllas:
e ska ligga i intervallet $1 < e < \varphi(n)$

Nästa villkor är för e att vara primt till $\varphi(n)$. Detta innebär att den största gemensamma delaren mellan e och $\varphi(n)$ ska vara 1.

$$sgd(e, \varphi(n)) = 1$$

Utöver det måste e ha en kort bitlängd. Begreppet bitlängd innebär antalet binära siffror som ingår i uppbyggnaden av ett heltal som en binär representation av sig själv. e ska dessutom inte ha särskilt många nollskilda element i sin binära representation [13, 1] (Vidare förklaring kring det binära systemet ges i kapitel 5).

När sändaren räknar klart ovanstående uppgifter kan hen då kryptera ett meddelande m på följande sätt:

$$C = m^e \pmod{N}$$

Där C är chiffertexten som genereras efter kryptering. Detta görs med hjälp av de offentliga nycklarna e och N och genom att beräkna inversen till meddelandet m modulo N. Svaret som vi får fram är då det krypterade meddelandet C som ska skickas vidare till mottagaren för dekryptering.

För att dekryptera ett RSA meddelande behöver mottagaren utöver de offentliga nycklarna skaffa även en personlig nyckel. Denna nyckel betecknas med d och kan räknas ut av kryptoskaparen med hjälp av följande formeln:

$$e \,.\, d \equiv 1 \,(\, mod\ \varphi(n))$$

Med andra ord söker man i det här läget ett d som bildar resten 1 vid division med $\varphi(n)$ när man multiplicerar den med e. Detta kan beräknas med hjälp av Euklides algoritmen.
Vi har då:

$$d = e^{-1} \,(mod\ \varphi(n))$$

Mottagaren använder sig därefter av följande formeln för att dekryptera meddelandet C:

Det ursprungliga meddelandet $M = C^d \ mod\ n$

Ovanstående processen görs i verkligheten med hjälp av kvantdatorer som möjliggör beräkningar som annars är omöjliga att utföra. Det är värt att notera att stora beräkningar med flera hundratals siffror kräver kvantdatorer med enorma kapacitet som även idag inte finns tillgänglig (se kapitel 5). Däremot kan man försöka få en djupare förståelse med hjälp av ett lite enklare exempel.

Exempel:

Om Vi exempelvis väljer $p = 5$ och $q = 7$, måste vi beräkna produkten av dessa två för att ta reda på den första offentliga nyckeln.

$$n = p \times q = 5 \times 7 = 35$$

$$\varphi(n) = (p - 1)(q - 1) = 4 \times 6 = 24$$

Den andra nyckeln e kan beräknas med hjälp av Eulers φ-funktion. Denna nyckel ska som sagt vara ett heltal sådant att:

$$sgd\left(e\,, \varphi(n)\right) = 1$$

$$1 < e < 35$$

Ett tal som uppfyller villkoren är exempelvis 5 som har inga gemensamma delare med $\varphi(n)$ och ligger dessutom inom intervallet.

Då har vi 2 offentliga nycklar i $e = 5\,,\ n = 35$

De offentliga nycklarna är kända för avsändaren och han kan nu kryptera ett meddelande M med hjälp av de och skicka över den till mottagaren. Dessa två nycklar kan möjligtvis vara kända av någon tredje person men den personen kommer ändå inte kunna dekryptera meddelandet utan den privata nyckeln som endast mottagaren har tillgång till.

Om avsändaren vill skicka ett meddelande $M = 3$ till mottagaren behöver han kryptera meddelandet innan han skickar iväg den för att undvika risken med att kommunikationen är avlyssnad. Det ska göras på följande sätt:

$$C = 3^5 \ mod\ 35$$

$$C = 243 \ mod\ 35$$

$$C = 33$$

Avsändaren har nu krypterats det ursprungliga meddelandet $M = 3$ med hjälp av de två offentliga nycklar han har tillgång till. Meddelandet som skickas vidare är då 33.

Därefter är det mottagarens tur att dekryptera meddelandet med hjälp av den privata nyckeln d som beräknas på följande sätt:

$$d = \ e^{-1}\ (mod\ \varphi(n))$$

Vi har då:

$$5 \times d \ \ (mod\ 24) = 1$$

$$d = 5$$

Mottagaren kan därefter få fram det ursprungliga meddelandet M på följande sätt:

$$M = \ C^d\ mod\ (n)$$

$$M = 33^5\ mod\ (35)$$

$$M = 39135393\ mod\ (35)$$

$$M = 3 \ mod\ (35)$$

Med hjälp av detta exempel har vi nu sett hur RSA fungerar i teorin. Trots att alla beräkningar som görs i verkligheten utförs av kvantdatorer och siffrorna som används är betydligt mycket större än vad vi har sett, så kan det ändå vara fördelaktigt att studera teorin med tanke på att teorin erbjuder ett djupare förståelse av vad säkerheten hos RSA baseras på. Om vi antar att en tredje part som avlyssnar kommunikationen försöker ta reda på vad det ursprungliga meddelandet var för något, inser vi hur svårt det är då tredje parten inte har någon aning om vad $\varphi(n)$ är och inte heller känner till primfaktoriseringen av n. Detta innebär att tredje parten har inte tillgång till den privata nyckeln d och kan då inte knäcka koden.

Trots att säkerheten hos RSA anses vara hög kan metoden befinna sig i fara om någon utomstående lyckas gissa primfaktoriseringen av n [13, 3]. I nästa kapitel studerar vi RSA i praktik och erbjuder även en modifiering av metoden som minskar säkerhetsrisken.

5

RSA i praktik

I föregående kapitlet studerade vi teorin som RSA grundar sig på och undersökte kryptering och dekrypteringsprocesessen i detaljer. Det är dock viktigt att komma ihåg att beräkningar som utgör RSA idag innehåller betydligt större siffror som i sin tur kräver att hela processen ska utföras med hjälp av kvantdatorer. Sådana enheter används till för att förenkla stora beräkningar och har dessutom förmågan att utföra flera beräkningar samtidigt. Säkerheten hos RSA kan öka kraftigt beroende på storleken hos utvalda nycklar då detta gör avslöjandet av primfaktorerna ganska osannolikt. Rekommenderade storleken på RSA nycklar är 2048 bitar som motsvarar tal med 617 signifikanta siffror. Detta minimerar förödande effekten av matematiska angrepp [14, 509].
I detta kapitel undersöker vi rollen som kvantdatorer spelar i relation till RSA, utvecklar vidare kring vikten av primtalen och dess användning i praktiken, studerar olika typer av angrepp som ett RSA kryptosystem kan möta och erbjuder slutligen en modifierade modell som kan göra angreppen mindre förödande.

5.1 Kvantdatorn

Kvantdatorer har förmågan att förse den digitala världen som vi lever i idag, med efterlängtade utvecklingen som krävdes för att hemlighålla information från obehöriga. Denna utveckling grundar sig på beräkningskraften och exponentiella skalbarheten som är inbyggd hos kvantdatorer [15, 1]. I detta kapitel ska vi nu undersöka kvantdatorns

fundamentala roll inom dagens kryptering, erbjuda ett grundläggande inblick på konceptet och förklara hur tekniken har möjliggjort nya alternativa metoder inom RSA och krypteringsvärlden.

För att kunna begripa kvantdatorns roll måste man förstå vad en kvantbit är. Generellt förlitar sig alla datorer på manipulering av data för att bearbeta information på kortaste möjliga tiden. Dagens datorer uppnår detta genom att manipulera enskilda bitar binärt. Sådana bitar kan då endast ha ett värde av 0 eller 1. Kvantdatorer har däremot förmågan att använda kvantmekanik och förlitar sig i stället på kvantbitar. Till skillnad från vanliga binära bitar, har kvantbitarna förmågan att kunna befinna sig parallellt i en kombination av både komponenttillstånden $|0\rangle$ och $|1\rangle$. [15, 11]

Om tillståndet som en kvantbit finner sig i ska mätas, förväntar man sig att biten ska falla tillbaka till en utav två bastillstånden. Dessa bastillstånd kan man representera med hjälp av vektorer på följande sätt:

$$|0\rangle := \begin{pmatrix}1\\0\end{pmatrix}; \qquad |1\rangle := \begin{pmatrix}0\\1\end{pmatrix}$$

När en kvantbit antar en komplex kombination av $|0\rangle$ och $|1\rangle$ kallas det för en kvantmekanisk superposition. Till skillnad från en klassisk bit som bara kan vara i tillståndet som motsvarar 0 eller tillståndet som motsvarar 1, kan en kvantbit vara i en superposition av båda tillstånden. Detta innebär att sannolikheten för att mäta 0 eller 1 för en kvantbit är i allmänhet varken 0,0 eller 1,0, och flera mätningar gjorda på kvantbitar i identiska tillstånd kommer inte heller alltid att ge samma resultat.

En kvantbit som befinner sig i superposition kan beskrivas med hjälp av en linjär kombination av $|0\rangle$ och $|1\rangle$ på följande sätt:

$$|\psi\rangle := \alpha|0\rangle + \beta|1\rangle = \begin{pmatrix}\alpha\\\beta\end{pmatrix}; \qquad |\alpha|^2 + |\beta|^2 = 1$$

Där α och β som är i grunden två komplexa tal och används till för att beskriva systemens beteende och definieras som sannolikhetsamplituder [15, 11].

Beräkningskraften hos en kvantdator ökar exponentiell med antalet kvantbitar och skiljer sig på det sättet ifrån konventionella datorer där beräkningskraften ökar linjärt med antal bitar som används. Detta beror på superposition och beräkningskraften som kvantbitar erbjuder vid sammankoppling. En exponentiell ökning i kraften hos kvantbitar sker när en kvantbit som kan enligt kvantfysiken anta både 0 och 1, sammankopplas till en annan kvantbit som ökar i sin tur kraften till att kunna anta 2^2 tillstånd samtidigt. Detta ger:

$$|00\rangle := \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}; \quad |01\rangle := \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}; \quad |10\rangle := \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}; \quad |11\rangle := \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Ovannämnde egenskapen hos kvantbitar erbjuder enorm beräkningskapacitet hos kvantdatorer. En kvantdator på 288 kvantbitar har förmågan att representera flera värden än vad det finns partiklar i universum [15, 12].
Kvantbitar och superposition är ett enormt område som är omöjligt att täcka i detta kapitel. Därmed ska vi fokusera på rollen som kvantdatorer spelar inom RSA krypteringsmetoden för att få en djupare praktisk förståelse av metoden.

Primfaktorisering av heltal på kvantdatorer

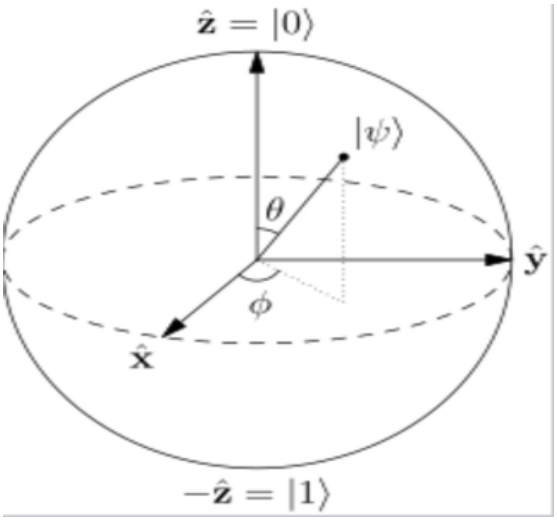
År 1994 kom matematikern Peter Shor upp med en algoritm för att primtalsfaktorisera heltalen med hjälp av kvantdatorer. Shor's algoritm går ut på att för ett givet tal N försöka hitta ett heltal p som befinner sig i intervallet {1, N} och delar dessutom N.
Algoritmen grundar sig på att utnyttja den exponentiella kraften hos kvantdatorer som används för utförandet av stora beräkningar för att genomföra faktoriseringsalgoritmen som kan i sin tur hota RSA och andra krypteringsmetoder [15, 23].

Algoritmen kan kortfattat beskrivas i följande 5 steg: [22]

Steg 1:
Antag N är heltalen som ska undersökas.
Låt m vara ett godtyckligt nollskilt heltal. Tillämpa algoritmen för största gemensamma delaren för att ta reda på $sgd(m, N)$.
Ifall vi har att $sgd\ (m, N) = 1$, går vi vidare till nästa steg. Annars har vi redan en lösning.

Steg 2:
Ta reda på vilken period $m\ mod\ N, m^2\ mod\ N,\ m^3 mod\ N,\ m^4\ mod\ n, \ldots$ har. Med andra ord, ska vi undersöka hur många divisioner det krävs för resten att börja hamna i en loop och repeteras.
Observera att kvantdatorer har förmågan att avgöra detta genom utförandet av kvant Fourier transformen som är en linjär transformation hos kvantbitar. Detta beror återigen på att

kvantdatorer är tillåtna att anta flera tillstånd samtidigt beroende på superpositionen. Då kan beräkningar för flera värden ske samtidigt (se figuren).



Figur 1. ovanstående är en representation av en kvantbit i superposition. Kvantbiten kan anta vilken position som helst på ytan av sfären då den befinner sig i superposition. När kvantbiten mäts faller den tillbaka till någon av polpunkterna |0⟩ eller |1⟩. [hämtad ifrån 15, 12]

Steg 3:

Ett jämnt resultat på perioden fungerar som grönt ljus för att gå vidare till nästa steg. Däremot kräver udda resultat en omstart från steg 1 med ett nytt godtyckligt heltal.

Steg 4:

Avgör huruvida $m^{(\frac{p}{2})} + 1 \neq 0 \, mod \, N$ (om p representerar perioden). Om detta stämmer, gå vidare till nästa steg. Annars gäller omstart.

Steg 5:

Slutligen ska datorn hitta största gemensamma delaren mellan (N, $m^{(\frac{p}{2})} - 1$)
Heltalet som fås fram i slutändan är en primfaktor till N [bevis och vidare förklaring hänvisas till 21, 14–18].

Shor's algoritm kan eventuellt användas till får att knäcka RSA om man har tillgång till tillräckligt kraftiga kvantdatorer med tillräckligt många kvantbitar. Dessvärre finns det inte tillräckligt kraftiga datorer i dagsläget som har förmågan att knäcka RSA som implementerats med tillräckligt stor bitlängd, det vill säga 2048 bitar som motsvarar 617 siffror.

Många forskare under 2000-talet har arbetat kring detta och allra flesta är överens om att kvantdatorer kommer i långa loppet agera som ett rejält hot mot RSA och andra krypteringsmetoder som grundar sig på svårigheten med primtalsfaktorisering.
35 som är ett tal med 6 bitar, är framtill idag det största heltalen som kvantdatorer lyckats faktorisera med hjälp av Shor's algoritm. Detta är långt ifrån 2048 bitar som är det rekommenderade storleken på RSA nycklar. För att kunna knäcka en lämplig implementerade nyckel med 2048 bitar krävs det en kvantdator med mycket större kapacitet. En kvantdator med 4099 kvantbitar kommer att ha förmågan att knäcka en RSA nyckel med 2048 bitar på 10 sekunder. Man förväntar sig att utvecklingen ska ungefär ta ytterligare 30 år. Däremot om det finns en sak man har lärt sig genom historien är det att genombrott sker när man minst anar det. [16, 1]

5.2 Faktorisering av stora heltal

Tidigare diskuterade vi primtalen, primtalstester och teorier kring detta under kapitel 3. Däremot är det minst lika fascinerande att undersöka hur faktorisering av större heltalen går till och hur man bevisar primalitet i praktiken. Under detta avsnitt ska vi försöka hitta svar på sådana frågor

5.2.1 Hur bevisar man primalitet

Primtalstester som diskuterades tidigare i arbetet och bland annat Miller-Rabins primtalstest, är i och för sig praktiska metoder till för att hitta stora heltal som är antagligen prima. Testet har dessutom förmågan att erbjuda en väldigt hög sannolikhet på primaliteten baserad på antal repetitioner (se kapitel 3). Trots det så finns det tydlig skillnad mellan ett relativt starkt påstående och rigoröst bevis [4, 136].

Rigorösa beviset är enkelt att förse åt små heltalen. Det gäller då bara att dividera givna heltalen N med alla tal i intervallet $\{1, \sqrt{N}\}$. Däremot krävs det flera livstid att bevisa primaliteten för heltalen med flera hundra siffror genom samma algoritm. Detta beror på att vi bemöter ett exponentiellt växande problem när det gäller bitlängderna. Det vill säga, tiden

som krävs för att lösa uppgiften växer exponentiellt beroende på hur många siffror som ingår i bitlängdens uppbyggnad. Med andra ord, antalet siffror som ingår i binära representationen av ett tal har en direkt koppling till hur långt tid det krävs att faktorisera talet.

Matematiker har arbetat på skapandet av en algoritm till för att bevisa primaliteten som är baserad på hypoteser som går att bevisa för över 150 år och det var Slutligen Agrawal, Kayal och Saxena som lyckades komma upp med ett sådant algoritm år 2002.
Testet som de kom fram till namnges efter dem och heter därmed AKS.
AKS testet går ut på följande:

För alla $\epsilon > 0$ finns det en algoritm som slutgiltigt avgör huruvida a givet heltal a är primt under $O((\ln n)^{6+\varepsilon}$ antal steg.
Idén bakom algoritmen bygger på följande egenskapen hos primtalen som är i och för sig en generalisering av Fermats lilla sats:

Lemma:

Låt $a \in \mathbb{Z}, n \in \mathbb{N}, n \geq 2$ och $(a, n) = 1$. Då är n prim om och endast om:

$$(X + a)^n = X^n + a \ \ (mod\ n)$$

Bevis:

För $0 < \mathrm{i} < n$ gäller det att koefficienten framför x^{i} i $((x + a)^n - (x^n + a))$ är lika med $\binom{n}{i} a^{n-\mathrm{i}}$.

Antag att n är primt. Då har vi att $\binom{n}{i} = 0 \ \ (mod\ n)$ och därmed alla koefficienter är 0.

Antag att n är sammansatt. Välj då ett primtal q som delar n och låt $q^k \parallel n$. Det vill säga, q^k ska vara parallellt med n.

Då kan q^k inte dela $\binom{n}{q}$ och är relativt prim till a^{n-q}. Därmed är koefficienten framför x^q nollskild $(mod\ n)$.

Utifrån detta kan vi dra slutsatsen om att $((x + a)^n - (x^n + a))$ över $\mathbb{Z}_n$ är inte identiskt noll. Detta innebär att kvantiteten inte kan anta värdet noll rigoröst. □

Ovanstående identitet föreslår ett enkelt test för primalitet.

Givet ett n, kan vi välja ett a och testa ifall kongruensen $(x+a)^n = x^n + a \ \ (mod\ n)$ är uppfylld. Denna process kan vara tidskrävande då vi behöver beräkna n koefficienterna. Man kan minska antal koefficienter genom att beräkna bägge sidor av kongruensen modulo ett polynom i form av $x^r - 1$ för en relativt liten godtycklig vald r.
Med andra ord ska man undersöka ifall följande ekvationen uppfylls:

$$(x+a)^n = x^n + a(mod\ x^r - 1, n).$$

Baserad på lemman vet vi sedan tidigare att alla prima n uppfyller ekvationen för alla godtyckliga värden på a och r. Ett problem som uppstår är då att vissa sammansatta n kan också uppfylla ovanstående ekvatioen.
Däremot kan man återställa ekvationen. Vi har då för ett lämpligt valt r, att n måste vara primpotenser om ekvationen är uppfylld för flera a. Antalet a och det lämpligt valde r, är både begränsade av ett polynom i $\log n$ och detta innebär att vi kan skapa en algoritm som testar primaliteten. [17, 783]
Vi ska nu gå vidare och redogöra för algoritmen.
För heltalen $n > 1$ har vi att:

I. Om $n = a^b$ (för $a \in \mathbb{N}$ och $b > 1$) returnerar algoritmen ”sammansatt”.
II. Hitta det minsta värdet på r så att $O_r(n) > Log^2 n$
III. Om $1 < (a, n) < n$ för några $a \leq r$, returnerar algoritmen sammansatt.
IV. Om $n \leq r$, algoritmen returnerar primtal och bekräftar primaliteten.
V. För alla a i intervallet $\{1, \lfloor \sqrt{\phi(r)} \log n \rfloor\}$, försöker algoritmen avgöra huruvida följande olikhet gäller. Om olikheten gäller, returnerar algoritmen sammansatt

$$((x+a)^n \neq x^n + a\ (mod\ x^r - 1, n))$$

Däremot returnerar algoritmen prime om olikheten inte uppfylls.

Algoritmen returnerar prime om och endast om n är primtal vilket innebär att primalitet som bevisas med AKS algoritmen är rigoröst. Algoritmen är däremot betydligt långsammare process i jämförelse med Miller-Rabins primtalstest. Detta leder till att de flesta är villiga att acceptera att ett heltal är primt såvitt de klarar 50–100 Miller-Rabins tester över det rigorösa beviset som AKS erbjuder och trots att AKS har en storslagen betydelse för praktisk kryptering, har Miller-Rabin fortfarande lyckats erhålla sin plats inom praktiken framtill idag [4, 137].

5.2.2 Kvadratisk Sieve algoritmen

Kvadratisk Sieve algoritmen som är även känd under namnet QS algoritmen, har funnits tillgänglig sedan 1980-talet och ansågs då att vara den mest effektiva metoden för faktorisering under 80-talet och även i början på 90-talet. QS som uppfanns av Carl Pomerance, är förstahandsvalet även idag när det kommer till faktorisering av heltalen med 50–100 siffror. QS Algoritmen är snabbaste faktoriseringsmetoden för heltal med upp till 110 siffror.

Om vi låter n beteckna heltalen som ska faktoriseras, försöker QS hitta två tal, x och y, så att:

$$x \not\equiv \pm y \ (mod\ n) \text{ och } x^2 \equiv y^2 \ (mod\ n).$$

Detta medför att:

$$(x - y)(x + y) \equiv o \ \ (mod\ n)$$

Vi kan då med hjälp av Euklides algoritmen beräkna följande:

$$(x - y, n)$$

Vilket kan avgöra ifall denna faktor är en icke-trivial delare. Sannolikheten att faktorn är icke-trivial är då minst 50 procent [18, 2]. Observera att icke triviala delaren som är nollskilda, är sammansatta [hänvisas till 18 för bevis].

Det första steget i algoritmen är att definiera följande funktion:

$$Q(x) = (x + \lfloor \sqrt{n} \rfloor)^{\,2} - n = \tilde{x}^{\,2} - n,$$

Och beräkna därefter $Q(x_1), Q(x_2), \dots, Q(x_k)$. För att ta reda på x_i och dessutom få produkten av $Q(x_i)$ att vara kvadratisk, behöver man införa en bas och bestämma ett Sieve intervall. För produkten att vara en kvadrat, krävs det att alla exponenter till primtalsfaktorer hos produkten ska vara jämna. Därmed behöver vi faktorisera alla $Q(x_i)$. Det är då viktigt att de är små och dessutom att man faktoriserar med en fast uppsättning av små primtal (där -1 är inkluderat). Detta är då vår bas.

För $Q(x)$-värden att vara små, behöver man välja x värden som ligger intill 0. Då kan man bestämma en begränsning M och endast inkludera x-värden som ligger i Sieve intervallet $[-M, M]$.

Då har vi om x befinner sig inom detta intervall och det finns dessutom några primtal p som delar $Q(x)$ att:

$$(x - \lfloor \sqrt{n} \rfloor)^2 \equiv n \ \ (mod\ p)$$

n är då en kvadratisk rest (mod p) och primtalen som befinner sig i vår bas måste uppfylla Legendre symbolen på följande sätt:

$$\left(\frac{n}{p}\right) = 1$$

Legendre symbolen används inom talteorin för att bestämma kvadratiska rester
För ett primtal p och ett relativt primt heltal med p som kallas för a har vi:

$\left(\frac{a}{p}\right) = 1$, om a är en kvadratisk rest mod p och är skild från 0 (mod p)

$\left(\frac{a}{p}\right) = -1$, om a är en icke-kvadratisk rest (mod p)

$\left(\frac{a}{p}\right) = 0$, endast om a är kongruent med 0 (mod p)

När vi har en mängd som består av primtalen som ingår i faktoriseringsbasen, kan vi börja insätta x-värden som är valda ifrån Sieve intervallet och beräkna $Q(x)$. Om ett x-värde inte är smidig i relation till faktorer som finns i basen kan vi släppa den och gå vidare och prova nästa x-värdet inom Sieve intervallet. Syftet med detta är att ta reda på vilka värden är smidiga över hela basen inom algoritmen (se definitionen nedan).

Definition:
Ett heltal n anses vara smidig i relation till B om alla primfaktorer till n är mindre eller lika med B. [4, 150]
Exempel på heltal som är smidiga i relation till 5:
2,3,4,5,6,8,9,10,12,15,16,18,20,24,25,27
Exempel på heltal som inte är smidiga i relation till 5 är:
7,11,13,14,17,19,21,22,23,26,28,

Det är viktigt att notera att processen ska utföras av datorer som har förmågan att täcka hela intervallet på en gång och eventuellt arbeta parallellt med andra datorer för att kunna täcka flera olika intervaller samtidigt [18, 3].
Denna process kallas för Sieving och vi ska nu redogöra för teorin som ligger bakom det.
Om p är en primfaktor till $Q(x)$ som vi tidigare definierade, har vi då att:

$$p \mid Q(x + p)$$

Omvänt gäller det också att om $x \equiv y \pmod{p}$ har vi:

$$Q(x) \equiv Q(y)\ (mod\ p)$$

Då ska vi för varje primfaktor p som befinner sig i vår bas, beräkna:

$$Q(x) = s^2 \equiv 0\ \ (mod\ p), x \in \mathbb{Z}_p$$

Vi ska erhålla 2 lösningar som kan kallas för s_{1_p}och $s_{2_p} = p - \ s_{1_p}$. Då ska ett antal $Q(x_i)$ som befinner sig i Sieve intervallet vara delbara med p när $x_i = s_{1_p}, s_{2_p} + pk$ för vissa heltal k. Detta kan lösas med hjälp av bland annat Shanks-Tonelli algoritmen (hänvisas till [18] för lösning och detaljer kring algoritmen). Sieving Processen kan därefter ske på flera olika sätt men den mest exakta metoden är att bestämma ett delintervall baserat på vilken storlek på minnet man har, och ordna $Q(x_i)$ för alla x-värden i delintervallet.

För varje p, börja vid s_{1_p}och s_{2_p} och dela ut den högsta möjliga potensen av p till var och en utav elementet och registrera de lämpliga potenserna av p (mod 2) som vektorer. Då kommer alla faktoriserbara element $Q(x_i)$ att ha varsin vektor och därmed kommer alla möjliga input överensstämma med ett unikt primtal i basen [18,4].

Exempel:
Primtalsfaktorisera 580 och bestäm dess vektor.
Lösning: $580 = 2^2 \times 5 \times 29$
Då ser motsvarande vektorn ut som (2,1,1) där siffrorna representerar primfaktorernas potenser.

Efter att alla primtalen har gått genom Sieve processen kommer de ordnade element som är nu 1 vara de faktorer som faktoriserar smidigt över hela basen. Då kan vektorer av primpotenser placeras i en matris inför beräkningen.

Sista steget i algoritmen är att bygga matrisen. Om $Q(x)$ går att faktorisera helt, då placerar vi primexponenterna (mod 2) i en vektor och vektorerna i en matris A. Inom matrisen representerar raderna våra $Q(x_i)$ och primexponenterna (mod 2) representeras av kolumner. Om vi exempelvis har basfaktorer enligt nedan:

$$\{-1, 2, 3, 13, 17, 19, 29\}$$

och har dessutom att:

$$Q(x) = 2 \times 3 \times 17^2 \times 19$$

Ser raden som motsvarar $Q(x)$ ut på följande sätt:

$$(0,1,1,0,0,1,0)$$

Vi ska dock komma ihåg att produkten av $Q(x_i)$ måste vara kvadratisk. Detta kan tolkas som att vi behöver summan av alla exponenter till var och en utav primfaktorer i basen att vara jämna och därmed kongruent med 0 (mod 2).

Givet $Q(x_1), Q(x_2), \dots, Q(x_k)$, ska vi då hitta lösning till:

$$Q(x_1)e_1 + Q(x_2)e_2 + \cdots + Q(x_k)e_k$$

Där e_i kan antingen var 0 eller 1. Detta gör vi då vi strävar efter att producera kvadratiska produkter ifrån givna $Q(x_i)$.

Därmed gäller det att:

Om $\vec{a}_i$ motsvarar rad A i matrisen som överensstämmer i sin tur med $Q(x_i)$, ska vi då beräkna följande:

$$\vec{a}_1 e_1 + \vec{a}_2 e_2 + \cdots + \vec{a}_k e_k \equiv \vec{0} \pmod 2$$

Ovanstående kongruensen innebär att vi behöver hitta lösning till:

$$\vec{e}A = \vec{0} \pmod 2$$

Där $\vec{e} = (e_1, e_2, \dots, e_k)$.

Vi kan då med hjälp av Gauss elimination beräkna höljet av vektorer i vektorrummet (se definitionen nedan). Vi behöver således hitta minst lika antal värde på $Q(x_i)$ som det finns faktorer i basen. Var och en av delmängderna i höljet motsvarar en utav delmängden i uppsättningen av alla $Q(x_i)$ vars produkt är kvadratisk [18, 5].

Definition:

Om alla vektorer i ett vektorrum V kan skrivas som en linjär kombination av $v_1, v_2, \dots, v_k$ kan vi påstå att V är genererad av $v_1, v_2, \dots, v_k$ och vektorer $\{v_1, v_2, \dots, v_k\}$ är då ett linjärt hölje till V. [19,1]

Därefter analyserar man produkterna i samband med att ta reda på ifall motsvarande produkten av $Q(x_i)$ och x_i kan resultera i tillverkningen av faktorer av n.

Detta görs genom att beräkna största gemensamma delaren på samma sätt som vi gick genom tidigare i avsnittet. Om processen misslyckas går systemet vidare till nästa element i höljet. När algoritmen slutligen lyckas med att hitta faktorer till n är det dags att testa primaliteten

hos de faktorerna. Det är värt att notera att primaliteten av dessa faktorer är garanterat vid tillämpning av algoritmen inom RSA.

5.3 Säkerheten hos RSA

Säkerheten som erbjuds av RSA är väldigt hög men inte perfekt. Som vi tidigare diskuterade, har invecklingen av kvantdatorer påverkat säkerheten som ett tvåeggat svärd. Man kan självklart påstå att kapabla kvantdatorer och dess förmåga att påskynda beräkningar med stora tal kan möjliggöra användning av nycklar med betydligt större bitlängd och därmed resulterat i ökad säkerhet hos RSA kryptering. Däremot kan man inte heller bortse ifrån att kvantdatorer kommer i långa loppet användas till för att knäcka lika stora koder och förenklar processen åt det andra hållet. Det finns ett antal olika typer av angrepp som kan hota systemets säkerhet och det är som sagt inte helt otänkbart att systemet knäcks av andra kvantdatorer i alla fall i framtiden. Vi ska nu beskriva två typer av angrepp som har potentialen att ställa frågor kring metoden.

Matematiska angrepp

Denna typ av angrepp som vi diskuterade under avsnitt 4.2, kan ske på 3 olika sätt:
Genom att bestämma primfaktorer p och $q\ (mod\ n)$. $\varphi(n)$ kan då beräknas enkelt genom utförandet av följande multiplikation:

$$\varphi(n) = (p-1)(q-1)$$

Därefter kan man enkelt ta reda på den privata nyckeln d med hjälp av Euklides algoritmen. Trots att algoritmen bakom faktorisering har ständigt förbättrats, är sannolikheten av att en sådant angrepp ska ha förödande hot mot RSA ganska låg om vi utgår ifrån att RSA systemet har implementerats på rätt sätt och nyckeln har tillräckligt stor bitlängd [2,204].

Delvis utsatthet av nyckeln

Låt $\langle N, d\rangle$ representera en privat RSA nyckel.
Om en utomstående lyckas fånga upp en del av bitar som tillhör nyckeln d, finns det en risk att hela bitlängden kan återuppbyggas. exempelvis kan det räcka med en fjärdedel av bitlängden, för att rekonstruera hela nyckeln. Detta beror dock på huruvida motsvarande privata nyckeln är tillräckligt stort.

Vi har: om $e < \sqrt{n}$, är det då möjligt att rekonstruera hela d endast med hjälp av en bråkdel av bitlängden. Detta betonar återigen betydelsen av säkerhetsåtgärder för RSA systemet i helhet.

Bevis:

Vi argumenterade att utsattheten av endast en fjärdedel av nyckeln kan leda till återuppbyggnaden av hela längden. Vi ska nu bevisa det.

I matematiska termer har vi:

Låt $N = pq$ vara en n-bit lång RSA nyckel. Då kan vi endast genom att ha tillgång till $\frac{n}{4}$ av de bitar som ingår i uppbyggnaden av p, faktorisera N.

Definitionen av e och d garanterar att ett k sådant att:

$$ed - k(N - p - q + 1) = 1$$

Existerar.

Med tanke på att $d < \varphi(n)$, måste vi ha:

$$0 < k \leq e$$

Om vi då låter $q = \frac{N}{p}$ och reducera ekvationen modulo $2^{\frac{n}{4}}$ får vi fram följande ekvationen:

$$(ed)p - kp(N - p + 1) + kN = p \mod 2^{\frac{n}{4}}$$

Vi vet sedan tidigare att tredje parten som försöker knäcka koden har tillgång till $\frac{n}{4}$ av signifikanta bitar av nyckeln. Detta innebär att $ed \mod 2^{\frac{n}{4}}$ är känd. Utomstående parten erhåller då en ekvation med p och k.

För e antal möjliga k-värden, ska ekvationen lösas för p. Då får man fram ett antal möjliga kandidater som värde på $p \mod 2^{\frac{n}{4}}$. För var och en av kandidaterna, använder man sig av algoritmen ovan för att faktorisera N. Det finns ett ändligt antal kandidater och högsta antalet kan uttryckas som $e \log_2 e$. [2, 210]

Dessa två, tillsammans med ett par andra typer av angrepp som beskrivs i mindre detaljer inom [13], kan möjligtvis utgöra ett hot mot RSA systemet som inte implementeras på ett korrekt sätt. I det nästkommande avsnitt ska vi presentera en modifierad metod som försvårar faktorisering av p och q och har förmågan att öka säkerheten av RSA systemet [13, 2].

5.4 Modifierad RSA

Vi ska nu beskriva en metod som har förmågan att eliminera N från nyckeln helt och hållet. Genom att introducera två nya steg till processen kan man göra så att det blir omöjligt att hitta p och q genom att faktorisera N. Detta kan då leda till en ökad säkerhet hos RSA och minimera risken av förödande skador som är orsakade av olika typer av angrepp mot systemet [13, 3]

Algoritmen kan delas upp i 3 olika faser som är i och för sig identiska med hur den vanliga algoritmen ser ut. Vi har:

1) Tillverkning av nycklar
2) Kryptering av meddelandet
3) Dekryptering av meddelandet

Vi påbörjar processen för att tillverka nycklarna. Detta sker under följande 6 steg:

I. Generera två distinkta godtyckliga primtal A och B

II. Beräkna $N = A \times B$. Observera att längden på detta ger i bitar.

III. Beräkna $\varphi(N) = (A - 1) \times (B - 1)$ på samma sätt som tidigare under RSA

IV. Beräkna K_1 som har följande egenskaper:

- $\sqrt{N} < k_1 < \varphi(N)$
- $sgd(\varphi(N), k_1) = 1$
- k_1 har kort bitlängd

V. Beräkna X som ersättare till N:

Om $A > B$ måste då X uppfyllas följande:

$$N - A < X < N$$
$$sgd(X, N) = 1$$

Om $A < B$ måste då X uppfylla följande:

$$N - B < X < N$$
$$sgd(X, N) = 1$$

VI. Hitta k_2 sådant att:

$$k_1 \times k_2 \ mod\ (X) = 1$$

Denna algoritm ger slutligen offentliga nycklar (k_1, X) medan den privata nyckeln är (k_2, X).

Nästa steg i algoritmen är kryptering av meddelandet. Avsändaren krypterar meddelandet PT, där PT står för plain text med hjälp av de offentliga nycklarna (k_1, X). Då har vi:

$$CT = PT^{k_1} \ mod \ (X)$$

Där CT representerar Chiffertexten som tillverkas efter kryptering.
Därefter under tredje och sista steget dekrypteras meddelandet CT av mottagaren med hjälp av privata nyckeln (k_2, X) enligt följande:

$$PT = \sqrt{(CT^{k_2} \ mod(X))}$$

Denna modifiering följer i princip samma algoritm och enda skillnaden i jämförelse med själva RSA algoritmen är att primfaktorer av N har inte längre lika förödande påverkan på säkerheten ifall de hamnar i fel händer.

I RSA är meddelandet begränsad då det ska vara mindre än värdet på n. I vår nya algoritm är begränsningen i stället att det ska vara mindre än kvadratroten av N. Detta beror på att det erhållna svaret när vi beräknar mod av ett heltal kommer alltid vara mindre än modulvärdet, det vill säga, talet med vilket Mod är tagen [13, 3].
Det är värd att notera att denna metod kan utvecklas vidare och garantera högre säkerhetsnivå. Detta gör man med hjälp av upprepning av steg 5, genom att hitta k_2 efter att X är känd. Skillnaden är dock att man behöver beräkna fjärde roten av svaret i stället. [13,3]

Referenslista:

[1] Calderbank, Michael. The RSA cryptosystem: history, algorithm, primes. (2017). Hämtad ifrån: http://math.uchicago.edu/~may/VIGRE/VIGRE2007/REUPapers/INCOMING/REU%20paper.pdf

[2] Boneh, Dan. 20 years of attacks on the RSA cryptosystem. p203-213. (1999). Hämtad ifrån: https://www.ams.org/notices/199902/boneh.pdf

[3] Löfwall, C. Thorbjörnson, J. En dag blev det mest onyttiga nyttigt: kryptering, talteori och RSA. (2003). föreläsningar-matematiska-institutionen. Hämtad ifrån: https://people.kth.se/~johantor/foredrag/lusttur/ht03/rsa/index.html

[4] Hoffstein, J. Pipher, J. Silverman, H, J. An introduction to mathematical cryptography. Second edition. Brown university. (2013). Hämtad ifrån: https://link-springer-com.ezp.sub.su.se/content/pdf/10.1007%2F978-1-4939-1711-2.pdf

[5] Flöjt, Linnea. Krypteringens historia och användningsområden. Göteborgs universitet, (2013). Examensarbete inom lärarprogrammet. Hämtad ifrån: http://www.math.chalmers.se/Math/Grundutb/GU/MMGL99/V13/VT13-3001-05%20Linnea%20Flojt.pdf

[6] Martinsson, Anders. Elementär talteori och primtalstester. Göteborgs universitet, (2017). Hämtad ifrån: http://www.math.chalmers.se/Math/Grundutb/GU/MMG100/S17/primalitet.pdf

[7] Tambour, Torbjörn. RSA kryptering. Stockholms universitet. Hämtad ifrån: http://mattefysik.se/ma5/RSA.Tambour.pdf

[8] Ek, Bengt. RSA kryptering och primalitetstest. Kursmaterial till diskret matematik. (2016). Kungliga tekniska högskolan. Hämtad ifrån: https://docplayer.se/109706271-Rsa-kryptering-och-primalitetstest.html

[9] Tambour, Torbjörn. Mängdteori och aritmetik. Kursmaterial till MM4000. Stockholms universitet. (2017).

[10] Shoup, Victor. A computational introduction to number theory and algebra. Betaversion 5. New York. (2004). Hämtad ifrån: https://shoup.net/ntb/ntb-b5.pdf

[11] Conrad, Keith. The Miller-Rabin test. Published in encyclopaedia of cryptography. (2011). Hämtad ifrån: https://kconrad.math.uconn.edu/blurbs/ugradnumthy/millerrabin.pdf

[12] Stallings, William. Cryptography and network security principles and practice. Seventh edition. Pearson. (2017). Hämtad ifrån: https://www.dl.hiva-network.com/Library/security/Cryptography-and-network-security-principles-and-practice.pdf

[13] Minni, R. Sultania, K. Mishra, S. An algorithm to enhance security in RSA. VIT university. India. (2013). Hämtad ifrån: https://www.academia.edu/11777363/An_Algorithm_to_Enhance_Security_in_RSA

[14] Hasib, Abdullah. A comparative study of the performance and security issues of AES and RSA cryptography. Norwegian university of science and technology. (2008). Hämtad ifrån: https://www.researchgate.net/publication/232615754_A_Comparative_Study_of_the_Performance_and_Security_Issues_of_AES_and_RSA_Cryptography

[15] Lundberg, J. Johannesson, T. Kvantdatorn-hot eller hype. Högskolan i Halmstad. (2021). Hämtad ifrån: https://www.diva-portal.org/smash/get/diva2:1570394/FULLTEXT02.pdf

[16] Baumhof, A. Breaking RSA Encryption – an Update on the State-of-the-Art. (2019). [Online]. Hämtad ifrån: https://www.quintessencelabs.com/blog/breaking-rsaencryption-update-state-art/.

[17] Agrawal, M. Kayal, N. Saxena, N. Primes is in p. Annals of mathematics. (2004). 781–793. Hämtad ifrån: https://annals.math.princeton.edu/wp-content/uploads/annals-v160-n2-p12.pdf

[18] Landquist, Eric. The quadratic Sieve factoring algorithm. Math 488. (2001). Hämtad ifrån: https://www.cs.virginia.edu/crab/QFS_Simple.pdf

[19] https://www.math.purdue.edu/files/academic/courses/2010spring/MA26200/4-4.pdf

[20] Bogvad, R. Xantcha, Q. Granath, H. Algebra 1. Matematiska institutionen. Stockholms universitet. Tionde tryckningen. (2018). Hämtad ifrån: https://kurser.math.su.se/pluginfile.php/73028/mod_resource/content/12/Algebra-i-tionde-tryckningen.pdf

[21] Shor, P. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Review, 41:303–332. (1999). Hämtad ifrån: https://arxiv.org/pdf/quant-ph/9508027.pdf

[22] https://anandsoni.in/shor/