Algoritem RSA

Uporaba, prednosti in slabosti

Benjamin Benčina

Univerza v Ljubljani
Fakulteta za matematiko in fiziko
Oddelek za matematiko

8. junij 2018

Kazalo

1 Uvod v kriptografijo

1.1 Kaj je kriptografija in kje se uporablja

Že od samih začetkov kulture človeštva je pripradnike naše vrste zanimal fundamentalen koncept zasebnosti. Kar se je začelo kot želja po osebnem prostoru in lastništvu osnovnih za preživetje potrebnih predmetov, se je kmalu sprevrglo v željo, če ne celo nujo, po anonimnosti. Kakor je povprečen človek rad v središču pozornosti, ima prav tako, če ne še bolj, rad skrivnosti. Nekatere misli imamo radi tajne, bodisi zaradi sramu ali spornosti bodisi ker je to potrebno za učinkovito in neovirano delovanje družbe.

Tako so luč dneva uzrli prvi zametki kriptografije, ki jo, kljub mnogim bolj ali manj poljudnim definicijam, lahko preprosto definiramo sledeče:
Kriptografija je umetnost skrivanja podatkov oz. informacij vsem na očeh.

Tukaj je ključen del ta, da so podatki skriti vsem na očeh. Seveda lahko podatke vedno fizično skrijemo na bolj ali manj premeten način, vendar tako vedno tvegamo, da jih nekdo najde, to pa ima pogosto lahko strašljive posledice. Ne tvegamo vedno le razkritja manjše osebne skrivnosti, zaradi katere nam je nerodno nekaj dni; vojne so bile dobljene in izgubljene zaradi neprimerne hrambe občutljivih podatkov, kar pomeni, da je lahko ogroženih na tisoče življenj.

V nasprotju s tem (pogosto) naivnim pristopom, kriptografski pristop reši problem tako, da preoblikuje podatke same. Tudi če so najdeni, to najditelju nič ne pomaga, saj iz prestreženega navideznega nesmisla podatkov ne more razbrati. Zato so podatki pogosto hranjeni kar javno; zanimivo je, da je internet kljub stremenju k zasebnosti popolnoma javen, le šifriran (oz. s tujko kriptiran) je v veliki meri.

Zaradi očitnih prednosti takega prenosa občutljivih informacij, pomembnih sporočil in podatkoh sploh ni presenetljivo, da se je koncept premetenega skrivanja podatkov po svetu hitro razširil. Med njegovimi prvimi uporabniki so bili mnogi dvorci, tajne in varnostne službe ter vojska, seveda pa tudi zločinci in kriminalne mreže. Kmalu je ideja postala tako popularna, da so jo uporabljali tudi otroci v igri, pripadniki intelektualne elite, da jim drugi ne bi kradli idej in zapiskov. Med bolj znanimi primeri je Leonardo da Vinci, ki je v najboljšem primeru pisal zrcalno, kar je mnogim kradljivcem povzročalo preglavice.

Človeški radovednosti pa vendar ni meja in mnogi se niso pustili prepričati v neomajno varnost zgodnje kriptografije. Tako so začeli iskati načine, kako bi šifre oz. kode na najbolj učinkovit način razbili, tj. našli njihov pravi pomen brez **ključa** - informacije, ki avtorizirani osebi omogoči preprosto razbiranje šifriranih podatkov. Tako se je rodila nova veda, če ne celo znanost, imenovana **kriptanaliza** - znanost razbiranja kodov brez avtorizacije. Če je kriptografija temeljila predvsem na izvirnosti avtorja šifre, je kriptanaliza po svoji naravi izrazito matematična veda. Nekaterim se je ideja razbijanja šifer, ki so inherentno uganke, zdela nadvse zabavna, ne le pomembna disciplina. Tako so se začeli s kriptografijo in kriptanalizo ukvarjati tudi ugankarji, vplive pa v razvedrilni matematiki vidimo še danes.
Ne smemo zanemariti tudi vpliva kriptografije in predvsem kriptanalize na lingvistiko, razbiranje pomena neznanega jezika v morda neznani pisavi z omejenim številom podatkov je v samem srcu kriptanalize, če zapis jemljemo kot kod.

1.2 Kriptografija pred računalnikom

Kriptografska praksa se je tako skozi stoletja razvijala in upodabljala v vseh možnih oblikah. Od tajnih pisav, piktogramov, premišljenih premetank pa do menjave črk ali celo celih besed za druge. Kot bolj zvito premetanko je vredno omeniti *skitalo*, pripomoček špartanskih generalov že pred našim štetjem. Na palico z natanko določenim polmerom so ovili za eno črko širok trak zaporedoma, da se ovoji niso prekrivali. Nato so vzdolž palice napisali sporočilo in odvili trak, kar je na sistematičen način premešalo črke besedila. Le imetnik palice z natanko enakim polmerom je lahko ta trak ovil okrog nje in razbral izvorno sporočilo.
Prav tako ne smemo pozabiti na *cezarjanko*, šifro iz rimskih časov izrazito polinomske oblike ($x \mapsto x + c$), kjer je c zamik abecede. Cezarjanka namreč deluje tako, da celotno abecedo zamaknemo za fiksno število črk, nato pa prvotne črke sporočila prepišemo v novi, zamaknjeni abecedi. Njena izboljšava, t.i. *Vigenèrov kvadrat*, kjer isto ciklično po črkah počnemo z dogovorjenim številom abeced, vsaka zamaknjena drugače, je bila dolgo časa sprejeta kot neomajno varna (število zamaknjenih abeced kot njihovi zamiki so seveda tajni), čeprav sta obe šifri močno občutljivi na kriptanalitično metodo *frekvenčne analize* (v katero se ne bom poglabljal), kjer pogledamo ponovitve posameznih šifriranih črk in to primerjamo z znanimi frekvencami črk v nešifriranem zapisu, nato pa po smislu uganemo neznane črke. Pri Vigenèrovem kvadratu je potrebno najprej ugotoviti število abeced, kar doda nivo varnosti. Zgodovinsko in filmsko bolj pismen človek najbrž pozna *enigmo*, strašljivo učinkovito šifro, ki so jo uporabljali nacisti v času druge svetovne vojne.

Tokrat šifer niso pisali na roke, temveč je za njih šifriral stroj podoben pisalnemu stroju, prav tako imenovan enigma, zato ta sistem spada v kategorijo t.i. mehanskih šifer. Dolgo časa nihče ni vedel, kako ga razbiti, ključ so namreč menjali na 24 ur, algoritem je bil skoraj odvečno zapleten, fizične stroje pa so varovali z življenjem in jih raje uničili, kot prepustili v roke Zaveznikom. Pa vendar je matematik *Alan Turing* ugotovil njene pomanjkljivosti in svoj abstrakten koncept, ki ga danes imenujemo Turingov stroj, udejanil v t.i. Turingove bombe, stroje, ki so bili sposobni v nekaj urah razbiti šifriranje enigme, danes pa so znani ne le kot sklop naprav, ki je zmagal vojno, vendar tudi kot prvi električni računalniki na svetu. S tem je Alan Turing postal oče računalništva in nezavedno postavil izziv:
Je kdo sposoben najti šifro, ki je računalnik ne bo sposoben razvozlati?

1.3 Simetrično in asimetrično šifriranje

Ni težko opaziti, da vse do sedaj omenjene šifre zahtevajo le en ključ. Podatek, s katerim smo šifrirali podatke je neposredno in predvsem vidno povezan s podatkom, s katerim podatke nazaj dešifriramo. Pri tajnih pisavah je to kar njihov predpis, pri skitali polmer palice, pri cezarjanki zamik abecede itd. Pa vendar temu ni vedno tako. Tako kot kvadratna enačba z dvema enakima ničlama, imajo vsi šifrirni sistemi (s tujko kriptosistemi) dva ključa, le enaka sta (lahko tudi samo vidno povezana, npr. pri cezarjanki je drugi ključ zamik abecede za isto število znakov v drugo smer). Tema ključema rečemo šifrirni in dešifrirni ključ (ang. encryption/decryption key), njuna funkcija je očitna iz imena.

Kriptosistemom, kjer sta šifrirni in dešifrirni ključ enaka oz. vidno povezana, pravimo **simetrične šifre**. Temu pa seveda ni vedno tako, le zaradi lažjega dojemanja in hitrejšega računanja sta si ključa enaka. Kriptosistemom, kjer šifrirni in dešifrirni ključ nista enaka, pravimo **asimetrične šifre**. Ravno asimetričnost pa je ključna lastnost, ki zagotavlja varnost algoritmu RSA.

2 Algoritem RSA

2.1 Motivacija

Z nadaljnim razvojem računalnika so izvirne tradicionalne šifre postale vse bolj nezanesljive, ročne so bile prepočasne in preprosto jih je bilo razvozlati, mehanične pa so bile izredno drage, njihovi algoritmi so bili prezapleteni in so pogosto v sebi skrivali človeški element, ki je vedno vir napak. Poleg tega so skoraj vse tradicionalne šifre, ročne ali mehanične, kot so pokazale Turingove bombe, proti računalniku neuporabne. Tako se je pojavila potreba po *univerzalnem* algoritmu na *matematično trdnih temeljih*, ki bi ga lahko uporabljali računalniki in bi hkrati bil proti njim tudi *varen*.

2.2 Ideja je rojena

Svet ni čakal dolgo. Leta 1978 so matematiki in računalničarji Ronald Rivest, Adi Shamir in Leonard Adleman objavili prvo različico svoje ideje, prvi kompleten asimetričen kriptosistem z **javnim in zasebnim ključem**. Imenovali so ga kar po začetnicah svojih priimkov - algoritem RSA. Do končne različice so potrebovali še nekaj let in leta 1983 so jim odobrili patent.

Če šifrirni algoritem razumemo kot matematično funkcijo, je bil izumiteljem algoritma RSA cilj napisati tako funkcijo, ki je *neobrnljiva* - torej iz šifrirane vrednosti ne moremo na noben smiselen način hitro razbrati originalne vrednosti, razen če smo imetnik posebnih privatnih podatkov, ki nam zagotavljajo, da le mi lahko razberemo šifrirane podatke. Če želimo *mi* prejeti šifrirano sporočilo, na poseben, kasneje opisan način tvorimo t.i. javni ključ, ki tukaj igra vlogo ključa za šifriranje. Nato vsem potencialnim pošiljateljem rečemo, naj po algoritmu zakodirajo svoje sporočilo in naj nam ga pošljejo. Ker je šifrirna funkcija neobrnljiva, tudi pošiljatelj sam ne more nazaj dešifrirati izvornega sporočila, prav tako pa ga ne more dešifrirati drug nelegitimen prestreznik podatkov. Iz zasebnih podatkov, ki jih poleg nas ne ve nihče in s katerimi smo tvorili javni ključ, sedaj tvorimo zasebni ključ, ki je z javnim sicer povezan, vendar na računsko zelo zapleten način. Z zasebnim ključem lahko sedaj le mi dešifriramo dobljene podatke. Če želimo svoje sporočilo poslati komu drugemu, prosimo za njegov javni ključ (če ga še nimamo), nato pa z njim šifriramo sporočilo in ga pošljemo.

V naslednjih podrazdelkih si bomo pogledali matematične osnove potrebne za razumevanje algoritma RSA, nato pa algoritem sam.

2.3 Matematične osnove

2.3.1 Modularna aritmetika in kolobar $\mathbb{Z}_n$

DEFINICIJA: **Modularna aritmetika** (včasih tudi urna aritmetika) po modulu n je aritmetika omejena s kongruenčno relacijo $aR_nb \iff n|b-a$.
Z drugimi besedami je to aritmetika v kolobarju $\mathbb{Z}_n$, tj. kolobar ostankov pri deljenju celih števil z n, kjer je n **modul**.

Zaradi lažjega pregleda bom definiral še novo operacijo in novo relacijo na množici celih števil.

OZNAKE:

- ☞ Operacija mod : $\mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$, ki slika $(a, b) \mapsto$ ostanek števila a pri deljenju z b,
- ☞ Relacija $a = b \pmod{n}$ pomeni, da celi števili a in b vrneta isti ostanek pri deljenju z n. Drugače: $a \bmod n = b \bmod n$.

Modularna aritmetika igra v algoritmu ključno vlogo, zagotavlja namreč neobrnljivost šifrirne funkcije. Funkcija, ki v svojem predpisu kot končno operacijo uporablja definirano uperacijo mod oz. modulo, je namreč skoraj agresivno neinjektivna, vsaka slika ima števno neskončno množico praslik.

PRIMER: Naj bo $f_5(x) = x \bmod 5$. Vprašajmo se, katero vrednost smo vstavili v funkcijo, da smo dobili vrednost 3?
Odgovor je očitno 3, pa tudi $8, 13, 18, \ldots$ Hiter premislek nam pove, da je množica rešitev enaka kar $f^{-1}(\{3\}) = \{5 \cdot k + 3; k \in \mathbb{Z}\}$, ta pa je v očitni bijektivni korespondenci z množico celih števil.

2.3.2 Funkcija φ in Eulerjev izrek

DEFINICIJA: Eulerjeva funkcija $\varphi(n)$ vrne število vseh naravnih števil manjših od n, ki so n tuja. Torej je $\varphi(n) = |\{a \in \mathbb{N}; a \leq n, \gcd(a, n) = 1\}|$.

Eulerjeva funkcija φ nam pomaga formulirati naslednji izrek:

IZREK: Če sta si števili x in n tuji, velja: $x^{\varphi(n)} = 1 \pmod{n}$.

Izreka samega ne bom dokazal(dokaz si lahko pogledate tukaj: [1], stran 26, izrek 2.1.20), služil pa bo kot ključno orodje pri dokazovanju trditve, ki nam zagotavlja matematično trdnost algoritma RSA. Obstajajo tudi drugi bolj ali

manj konstruktivistični dokazi, vendar se bomo tukaj omejili na malce bolj matematičen pristop.

OPOMBA: Vredno je omeniti, da če je p praštevilo, je $\varphi(p) = p - 1$. Če je n produkt samih tujih praštevil, velja tudi, da je $\varphi(n)$ enak produktu vseh $\varphi(p_i)$ za vse p_i, ki n delijo.

2.3.3 Polinomi in številski sistemi

Čeprav so za algoritem brezpomenski, polinomi še vedno igrajo veliko vlogo pri šifriranju; ne le pri algoritmu RSA, temveč tudi pri drugih kriptosistemih. Pomembna opazka, ki se bo pojavila, ko bomo algoritem podrobneje spoznali, je namreč, da šifrirna funkcija sprejema in vrača števila. Besedilna sporočila je potrebno zato "kodirati", tj. predstaviti z enoznačnim številom. V računalništvu obstaja mnogo vrst kodiranja, npr. ASCII in UTF-8, včasih pa je pametno podatke predstaviti na svoj način. Slednje nam v veliki meri omogočajo polinomi.

DEFINICIJA: Polinom f je formalna vsota $f(X) = a_0 + a_1X + \cdots a_nX^n$. X imenujemo spremenljivka, številom a_i pravimo koeficienti, n pa je stopnja polinoma. Vsak polinom porodi **polinomsko funkcijo**.

DEFINICIJA: Število je zapisano v n-iškem **številskem sistemu**, $n \geq 2$, če je enako vrednosti neke polinomske funkcije iz $\mathbb{Z}_n$ izračunane za $X = n$ in je a_i števka tega števila na i-tem mestu, šteto od desne proti levi od 0 naprej.

Znake lahko sedaj zakodiramo na primer tako:

- izračunamo moč množice znakov (abeceda), ki bi jih radi kodirali, to število naj bo m,
- vsakemu znaku iz te množice priredimo enolično vrednost iz množice $\mathbb{Z}_m$,
- preštejemo število znakov, ki bi jih radi zakodirali (besedilo), naj bo to število j,
- sestavimo polinom $f(X) = a_0 + a_1X + \cdots + a_{j-1}X^{j-1}$,
- naše iskano število je vrednost polinomske funkcije tega polinoma izračunane za $x = m$, kjer so koeficient a_i vrednosti črk na i-tem mestu besedila.

PRIMER: Premislimo lahko, kako sploh pojmujemo naš standardni desetiški zapis. Oglejmo si na primer število 1327. Načeloma je $1327 = 1 \cdot 10^3 + 3 \cdot 10^2 + 2 \cdot 10 + 7$. Po obratnem postopku sedaj namesto 10 pišemo X in vidimo, da je število v desetiškem zapisu le polinom s koeficienti iz $\mathbb{Z}_{10}$, torej med 0 in 9, izračunan v $x = 10$.
Če bi na primer delali s slovensko abecedo, bi za m izbrali 25 in na očitno enoličen način tvorili polinomsko funkcijo, kjer bi A priredili vrednost 0, B vrednost 1 itd., te vrednosti pa pisali kot koeficiente na ustrezno mesto glede na besedilo, ki bi ga kodirali.

2.4 Algoritem

2.4.1 Kako deluje

Kljub visoki kompleksnosti in, kot bomo videli, visoki ravni varnosti, je vsaj idejno algoritem sila preprost. Kot tvorec ključev sledimo naslednjemu postopku:

- izberemo si dve *različni praštevili* p in q, to bosta naša privatna podatka, zato jih skrbno hranimo,
- izračunamo $n = p \cdot q$ in $\varphi(n) = \varphi(p) \cdot \varphi(q) = (p-1) \cdot (q-1)$,
- izberemo naključen e, za katerega velja $\gcd(e, \varphi(n)) = 1$,
- **javni ključ nam tvori par** (e, n),
- z razširjenim Evklidovim algoritmom lahko sedaj relativno hitro kot edini imetniki $\varphi(n)$ izračunamo d, ki je multiplikativen inverz za e v kolobarju $\mathbb{Z}_{\varphi(n)}$, drugače: $e \cdot d = 1 \pmod{\varphi(n)}$,
- **zasebni ključ nam tvori par** (d, n).
- Sporočilo m sedaj šifriramo tako: $c = m^e \bmod n$.
- Skrito sporočilo dešifriramo tako: $m = c^d \bmod n$.

2.4.2 Zakaj deluje

Ko smo po tem ne preveč zapletenem postopku zašifrirali sporočilo, se je potrebno vprašati, ali podan obraten postopek res vrne nazaj izvorno sporočilo, z drugimi besedami, je to kljub neobrnljivosti skriti inverz šifrirne funkcije? Glede na masovno uporabo algoritma, je odgovor očitno pritrdilen, to pa

nam zagotavlja naslednji izrek.

IZREK: Naj bo $n \in \mathbb{Z}$ produkt samih različnih praštevil in naj bosta $d, e \in \mathbb{N}$ taki, da za $\forall p \in \mathbb{P}$, kjer $p|n$, velja: $(p-1)|(d \cdot e - 1)$. Tedaj:

$$\forall a \in \mathbb{Z} : \; a^{d \cdot e} = a \pmod{n}.$$

DOKAZ: Ker $n|a^{de} - a$ natanko tedaj, ko $p|a^{de} - a$ za vsak praštevilski delitelj p števila n, je dovolj pokazati, da za vsak tak p velja $a^{de} = a \pmod{p}$. Ločimo dve možnosti:
Če $\gcd(a, p) \neq 1$, potem je $a = 0 \pmod{p}$ očitno, saj je zaradi praštevilskosti števila p $a = kp$ za neko celo število k. Trivialno sledi, da je tudi $a^{de} = a \pmod{p}$, saj sta obe strani enaki 0.
Če pa $\gcd(a, p) = 1$, po Evklidovem izreku sledi, da je $a^{\varphi(p)} = a^{p-1} = 1 \pmod{p}$. Ker po predpostavki $(p-1)|(de-1)$, velja tudi $a^{de-1} = 1 \pmod{p}$. Če pomnožimo še obe strani enakosti z a, dobimo željeni rezultat. □

Oznake v izreku so sugestivne, predstavljajo namreč istoimenske oznake v opisu postopka algoritma. Ker izrek velja za vsako celo število a, velja tudi za naše sporočilo m (potrebno je sicer paziti, da je $m < n$, vendar to zaradi običajne velikosti števila n ni problem, sicer pa sporočilo m razdelimo na več kosov).
Naše sporočilo m je šifrirano tako: $c = m^e \bmod n$. Ali z obratnim postopkom res dobimo prvotno sporočilo?
$c^d \bmod n = (m^e)^d \bmod n$, to pa je po izreku enako $m \bmod n$. Če smo pazili, da je $m < n$, je m res enolično določen in zato izvorno nešifrirano sporočilo.

Sedaj, ko vemo, da algoritem deluje, bi radi preizkusili njegovo varnost. Najbolj učinkovito je, da se postavimo v čevlje napadalca, bodisi neavtoriziranega prestreznika sporočila bodisi prisluškovalca pri prejemniku sporočila, ter se vprašamo o slabostih in šibkih točkah RSA kriptosistema. Te šibke točke bomo poskusili kar se da izkoristiti, nato pa je naš cilj algoritem popraviti, če se le da.

3 Napadi na RSA kriptosistem

3.1 Načrt napada

Če si podrobneje pogledamo algoritem, v njem vidimo tri možne točke napada, te se pojavijo vsakič, ko je potrebno nekaj izračunati (v našem primeru so to $n, \varphi(n), d$). Obstajajo torej 3 bolj ali manj očitne taktike:

- s faktorizacijo določimo praštevili p in q modula n, nato pa sami izračunamo $\varphi(n)$ in posledično d,
- določimo $\varphi(n)$ direktno iz javnega ključa (e, n) in posledično izračunamo d,
- določimo d direktno iz javnega ključa (e, n).

Najbolj očiten deluje prvi pristop in pokazati se da, da je zahtevnost drugega in tretjega pristopa enaka zahtevnosti faktorizacije števila n. Torej je varnost algoritma proti matematičnim napadom določena s časom potrebnim za faktorizacijo števila n na p in q, kar pa je računsko zelo zahteven problem.

Že v antični Grčiji so vedeli, da je vsako celo število n možno zapisati kot produkt samih praštevil (upoštevajoč predznak). Prav tako so poznali problem faktorizacije števila, torej problem iskanja teh praštevil, ki n sestavljajo. Učinkovitega algoritma za faktorizacijo niso poznali, števila so razstavljali s poskušanjem. Majhna števila je bilo zato razmeroma lahko faktorizirati, medtem ko se je pri večjih številih z velikimi prafaktorji stvar precej zapletla. Podobno velja še danes. Izkaže se, da je najtežje faktorizirati velika števila, ki so sestavljena le iz dveh praštevil podobnega velikostnega reda, kar pa n iz algoritma RSA je. Algoritmi za faktorizacijo so sicer v dobi računalnikov postali malce bolj učinkoviti, vendar pa zares učinkovitega algoritma ni. Že samo preverjanje praštevilskosti je za splošno praštevilo počasen proces, potencialne delitelje moramo namreč preveriti do korena tega praštevila (po korenu je zapis očitno simetričen).

Več o algoritmih za faktorizacijo celega števila si lahko preberete tukaj [4], sam pa se v podrobnosti ne bom spuščal.

Vendar pa v napadalnem računalništvu velja pravilo, da je sistem le tako trden, kot je ga je trdnega naredil tisti, ki ga je postavil, ljudje pa delamo napake. Tako lahko ob naši nepazljivosti pri izbiri ključev ali hranjenju skrivnih

podatkov vešč napadalec uspešno vdre v naš kriptosistem. V nadaljevanju bo sledilo 8 takih napadov.

3.2 Nekaj možnih napadov in njihova izvedba

3.2.1 Faktorizacija n, če poznamo $\varphi(n)$

Recimo, da nam je kot napadalcu znano postalo število $\varphi(n)$. To se sicer zgodi le redko, pa vendar je možno, da je hranitelj tajnih podatkov, torej prejemnik sporočil, podatke hranil slabo. Čeprav bi lahko direktno izračunali d, si želimo vseeno faktorizirati n na p in q, da bomo o kriptosistemu res vedeli vse. Kot bomo videli, nam imetje podatka $\varphi(n)$ faktorizacijo n trivializira.

Sedaj imamo za dve neznanki p in q namreč dve enačbi:

$$n = p \cdot q \tag{1}$$

$$\varphi(n) = (p-1) \cdot (q-1) \tag{2}$$

Po (2) velja $\varphi(n) = pq - (p+q) + 1$. Torej poznamo tako $pq = n$ kot $p + q = n + 1 - \varphi(n)$.
Po Vietovih formulah pa sedaj vemo, da sta p in q rešitvi kvadratne enačbe

$$x^2 - (p+q)x + pq = (x-p)(x-q) = 0 \tag{3}$$

in kot taki izračunljivi po splošni formuli za rešitve kvadratne enačbe.

Rešitev: Moralo bi biti očitno, vendar matematično manj pismenim pogosto ni, da varna hramba le p in q ni dovolj. Vse informacije, ki niso javne narave, so praviloma zasebne narave. Zato nikoli ne delimo o našem kriptosistemu ničesar z izjemo javnega ključa.

3.2.2 Napad na majhno razliko števil p in q

Iskanje praštevil je za računalnik precej naporno in morda se zdi milostna ideja, da vzamemo za p in q kar zaporedni praštevili, morda celo praštevilski dvojček. Ker pa varnost RSA kriptosistema sloni predvsem na težavni faktorizaciji števila n, to seveda nikoli ni dobra ideja. Poglejmo si zakaj.

Denimo, da vemo (oz. se nam dozdeva), da je razlika števil p in q majhna. Potem lahko poskusimo n faktorizirati s *Fermatovo faktorizacijsko metodo.*

Brez izgube splošnosti lahko rečemo, da $p > q$ (v nasprotnem primeru je postopek popolnoma simetričen na p in q). Potem velja:

$$n = \left(\frac{p+q}{2}\right)^2 - \left(\frac{p-q}{2}\right)^2. \tag{4}$$

Ker je razlika p in q majhna je $s := \frac{p-q}{2}$ majhno število, $t := \frac{p+q}{2}$ pa je le malo večji od $\sqrt{n}$. Velja:

$$t^2 - n = s^2. \tag{5}$$

Za t preprosto jemljemo vrednosti $\lceil \sqrt{n} \rceil + k$, kjer k teče po zaporednih nenegativnih celih številih, in jih vstavljamo v enačbo (5). Ko dobimo na desni strani popolni kvadrat, smo zadostili enačbi in tako našli pravi števili s in t.

Sedaj preprosto sledi $p = t + s$ in $q = t - s$.

3.2.3 Faktorizacija n, če poznamo d, oz. napad na skupen modul

Recimo, da smo nekako izvedeli, kaj je zasebni ključ d naše tarče. Radi bi faktorizirali n na praštevili p in q. Pojavi se seveda vprašanje, zakaj bi to naredili? Odgovor je preprost: izkaže se, da si skupina ljudi včasih deli skupen modul n, uporabljajo le različen e. Če poznamo d enega od pripadnikov te skupine in znamo z uporabo tega podatka učinkovito faktorizirati n, smo vdrli v kriptosistem celotne mreže ljudi, ki si deli modul n.

Znan je verjetnosti algoritem, tj. algoritem z visoko verjetnostjo uspeha, ki nam netrivialno razstavi n.

ALGORITEM: Recimo, da je d znan podatek, (e, n) pa je javni ključ za ta d. Definiramo $m := de - 1$. Torej velja $a^m = 1 \pmod{n}$.

- Če je m sod in $a^{m/2} = 1 \pmod{n}$ za več naključnih a-jev, nastavimo $m = \frac{m}{2}$. Korak ponavljamo dokler so pogoji na m zadoščeni.
- Izberemo naključen a in izračunamo $g = \gcd(a^{m/2} - 1, n)$.
- Če je g pravi delitelj n, smo našli enega izmed faktorjev in s tem oba, postopek zaključimo. V nasprotnem primeru se vrnemo nazaj na prejšnji korak.

Algoritma sam ne bom dokazal (dokaz si lahko pogledate tukaj: [1], stran 64, algoritem 3.4.5). Ta napad se namreč v praksi najmanj uporablja, saj velja dogovor:

REŠITEV: Dve osebi naj nikoli ne uporabljata istega modula n. Praštevil je neskončno mnogo, zato za neupoštevanje tega dogovora ni izgovora.

3.2.4 Napad z izbranim šifropisom

To je preprost napad, ki temelji na standardni velikosti **blokov** sporočila. Algoritem RSA je namreč t.i. *bločna šifra*, kar pomeni, da je sporočilo razdeljeno v bloke navadno standardne velikosti. To sledi iz pogoja, da mora biti za enoličnost dešifrirne funkcije sporočilo $m < n$. Ker temu seveda pri poljubnem sporočilu ni tako, sporočilo preprosto razbijemo v take kose, da je številčna vrednost m vsakega kosa manjša od n.
Problem se pojavi, ko napadalec iz velikosti blokov lahko sklepa o našem izvornem sporočilu (npr. napadalec ve, da ob določeni uri pošljemo vremensko napoved). Tako lahko določi najbolj verjetna sporočila in jih šifrira še sam, nato pa primerja s prestreženimi bloki. Če se šifri ujemata, je našel izvorno sporočilo in iz tega lahko sklepa o njegovem preostanku in tako določi najbolj verjetna sporočila za preostale (še neznane) bloke.

REŠITEV: Najbolj učinkovita metoda, ki je dandanes kar standard bločnih šifer, je t.i. *soljenje* sporočila. Začetek vsakega bloka posolimo tako, da mu dodamo dogovorjeno število naključnih znakov. To zagotavlja, da se isto sporočilo nikoli ne šifrira večkrat v isti kod. Vsekakor pa je vedno potrebno paziti, da je naša izbira javnega ključa takšna, da je množica možnih sporočil kar se da velika.

3.2.5 Napad na majhen šifrirni eksponent e

Spet se morda zdi, da je zaradi računske učinkovitosti pametno izbrati čim manjši e. Spet je to huda napaka, močno lahko namreč škodi neobrnljivosti šifrirne funkcije.

Zamislimo si, da ima sporočilo m glede na n relativno majhno vrednost. Če je eksponent e dovolj majhen, se lahko pripeti, da je tudi $m^e < n$. Potem je $c = m^e$, saj modulo n v šifrirni funkciji vrednosti ne spremeni. Torej lahko iz šifre c preprosto dobimo izvorno sporočilo, če izračunamo $m = c^{1/e}$.

REŠITEV: Premajho sporočilo m na začetku spet posolimo in mu tako z dodanimi znaki povečamo vrednost, da velja $m^e > n$. Poleg tega je v splošnem pametneje izbrati malo večji e.

3.2.6 Napad na majhen dešifrirni eksponent d

Zopet bi se mogoče v imenu računske učinkovitosti zdelo smiselno izbrati take e, p in q, da bo d čim manjši. Podobno kot v prejšnjem primeru je to huda napaka. Kriptograf Michael J. Wiener je namreč pokazal, da se da d učinkovito določiti, če velja $d < \frac{1}{3} \cdot n^{1/4}$ in če za praštevili p in q velja $q < p < 2q$ (analogno za $q > p$), kar pa ni tako redko, saj praštevila pogosto izbiramo v določenenem velikostnem redu, ki pa je še vedno dovolj velik, da števili p in q nista preveč skupaj.

Bolj podroben opis tega napada si lahko pogledate tukaj: [5].

REŠITEV: Tukaj se rešitev zdi očitna: pazljivo moramo izbrati ne premajhen e, da bo $d > \frac{1}{3} \cdot n^{1/4}$.

3.2.7 Napad na majhen prostor sporočil

Če je slika šifrirne funkcije pri izbranem javnem ključu premajhna, lahko napadalec kriptosistem očitno učinkovito napade že s surovo silo (*brute force*), saj lahko preprosto izračuna vse možne šifrirane nize in jih primerja s prestreženimi. Na velikost prostora sporočil vpliva veliko parametrov, predvsem velikost praštevil p in q, izbira e in posledično izbira d. Potemtakem je očitno, da je ta napad preprosta kombinacija tehnik napada z izbranim šifropisom, napada na majhen šifrirni eksponent in napada na majhen dešifrirni eksponent.

REŠITEV: Splošna rešitev je zopet naključno soljenje sporočila, predvsem pa pametna izbira ključa. Že zaradi problema faktorizacije števila n na prafaktorja p in q, morata biti praštevili zelo veliki - priporočen velikostni red je najmanj 2^{4096}, večje organizacije pa uporabljajo tudi mnogo večja praštevila.

3.2.8 Cikličen napad

Zadnji napad je poleg napada s surovo silo (pregled vseh možnosti) konceptualno najpreprostejši, pa vendar še najbolj zvit. Sledi preprostemu algoritmu:

ALGORITEM: Naj bo c_0 prestreženo šifrirano sporočilo.

- ➢ c_0 še enkrat šifriramo in dobimo novo šifro c_1. Korak ponavljamo na c_i, dokler ne velja $c_i = c_0$, kjer je c_0 prvotno prestreženo sporočilo.

➢ Ko je $c_i = c_0$, preprosto pogledamo eno šifriranje nazaj, katero sporočilo smo šifrirali, da smo dobili nazaj c_0. To sporočilo zaradi enoličnosti algoritma mora biti izvorno sporočilo.

Vprašanje, ki si ga lahko zastavimo, je: ali je možno, da se nam algoritem na neki točki vrne na nek $c_i \neq c_0$? S tem bi seveda nehal delovati, saj bi v neskončnost krožil po istih nepravilnih vrednostih. Na napadalčevo srečo se to ne more zgoditi:

KOMENTAR: Ko zaporedoma šifriramo prestreženo skrito sporočilo, je na i-tem koraku sporočilo tako: c^{e^i}. Radi bi, da obstaja tak k, da velja $c^{e^k} = c$. Iz podatka $\gcd(e, \varphi(n)) = 1$ sledi, da je $e \in \mathbb{Z}^*_{\varphi(n)}$, tj. element multiplikativne grupe obrnljivih elementov aditivne grupe $\mathbb{Z}_{\varphi(n)}$. Podgrupa $< e >$ te multiplikativne grupe, ki je generirana z elementom e, je torej ciklična in končna, vsebuje vse možne potence števila e, med drugim tudi inverz števila e, ki pa smo ga označili z d. Ko z i iteriramo zaporedoma po vseh potencah števila e torej zaradi končnosti grupe $< e >$ mora obstajati tak k, da $c^{e^k} = c^{e^0} = c$.

Vredno je omeniti, da je ob pravilni izbiri ključev ta napad časovno ekvivalenten napadu s surovo silo. Njegova edina prednost je, da je odporen na soljenje sporočila, saj je algoritem popolnoma neodvisen od ključev in sporočila samega, od tega je odvisno le njegovo trajanje.

4 Zaključek

Kot smo videli, obstajajo mnogi bolj ali manj zapleteni napadi na RSA kriptosistem, vsakega pa se da relativno enostavno preprečiti, če le malo pazimo pri izbiri ključev in sporočilu dodamo mero naključnosti. Algoritem RSA je zato zelo varen in širom uporabljen še danes. Uporablja ga recimo popularen protokol za varen prenos podatkov čez omrežje OpenSSH, s katerim se povprečen uporabnik interneta nezavedno sreča skoraj dnevno, matematiki in programerji pa ga poznamo predvsem zato, ker ga uporablja platforma za nadzor različic GitHub.
Pogosto se algoritem RSA uporablja tudi za overjanje istovetnosti oz. avtentikacijo. Če vemo, da določen javni ključ pripada neki osebi, lahko ta oseba izkaže svojo istovetnost tako, da nam s preprosto izmenjavo sporočila dokaže, da si lasti pripadajoči zasebni ključ.

Algoritem RSA ima seveda ključno pomanjkljivost - računsko je zelo počasen. Zato ponavadi ni primeren za izmenjavo velike količine podatkov v realnem času. Uporablja se predvsem za varno izmenjavo ključa neke hitrejše simetrične šifre, preko katere potem poteka nadaljna izmenjava podatkov. Kljub temu je nepogrešljiv in izredno varen del kriptografije, ki je kljub svoji starosti preživel zob časa.

Literatura

[1] Stein W. *Elementary Number Theory: Primes, Congruences, and Secrets*, US San Diego, Harvard, University of Washington, 2017, dostopno na `https://wstein.org/ent/ent.pdf`

[2] Singh S. *The code book*, Four Estate Ltd, 2003

[3] Gupta P.C. *Cryptography and network security*, PHI Learning Private Limited, Delhi, 2015

[4] *Integer factorization*, dostopno na `https://en.wikipedia.org/wiki/Integer_factorization`

[5] *Wiener's attack*, dostopno na `https://en.wikipedia.org/wiki/Wiener%27s_attack`