UNIVERZA V LJUBLJANI

Fakulteta za elektrotehniko

Uroš Česnik

Asimetrični šifrirni postopki

LJUBLJANA, 2006

Kazalo

UVOD

Živimo v svetu polnem podatkov. Vsak izmed njih ima za nas določen pomen. Nekatere podatke, ki jih imamo, posredujemo vsem, nekatere samo določenim in nekatere obdržimo zase. Te podatke že mnogo let prenašamo na različne načine, z različnimi mediji, z različnimi tehnikami.

V današnjem času je prišlo do eksplozije svetovnega spleta in s tem so se tudi odprle različne možnosti hranjenja in prenosa podatkov. Sam prenos in hranjenje pa zahteva določeno mero varnosti, saj želimo določene podatke ohraniti tajne oz. nedostopne vsiljivcem.

Večino poslov opravljamo v elektronski obliki, prav tako pa imamo v tej obliki vedno več dokumentov. Sama arhitektura svetovnega spleta v osnovi ne podpira varovanja. Zaradi tega je potrebno podatke varovati z različnimi sredstvi, saj je računalnik povezan v splet potencialna tarča vsiljivcev.

Sredstva in ukrepi, ki so nam na voljo, so zelo omejena. Ne moremo preverjati vseh oseb in jim dovoliti, ali prepovedati uporabo do našega računalnika. Računalnike lahko fizično ločimo pred ostalim svetom, vendar je za dostop do njih dovolj le priklop v omrežje. Za zaščito lahko uporabljamo določene proceduralne ukrepe, vendar nam opravljanje transakcij po določeni proceduri ne pomaga veliko, saj lahko vsiljivci pridejo do podatkov ob upoštevanju procedur. Najboljša rešitev so tehnični ukrepi. Najpogostejši način tehničnega varovanja je prikrivanje ali šifriranje.

Izbira pravega načina šifriranja podatkov je zahteven proces, saj poznamo veliko šifrirnih algoritmov z različnimi dolžinami ključev, različnimi matematičnimi temelji, različnimi pogoji za zagotavljanje varnosti itd.

Kriptografija z javnimi ključi je naredila prve korake sredi 70. tih let. Glavna značilnost kriptografije z javnimi ključi je, da uporablja dva različna ključa, zasebnega in javnega, in tako ji pravimo tudi asimetrična kriptografija.

V tem seminarskem delu bo govor o asimetričnem šifriranju, njegovih najbolj značilnih sistemih in lastnostih.

1 KRIPTOGRAFIJA

Kriptografija ali kriptologija je polje matematične in računalniške znanosti povezane z varnostjo informacij in ostalimi povezanimi področji, predvsem z šifriranjem. Tehnično kriptografija obravnava tehnike, kriptologija pa je študija teh tehnik. Kljub temu pa izraz kriptografija uporabljamo za opis celotnega področja.

Kriptografija je interdisciplinaren predmet, ki predstavlja različna področja. Starejše oblike kriptografije so bile predvsem povezane z vzorci v jeziku. Nedavno se je nivo kriptografije dvignil in začel intenzivno uporabljati matematiko, diskretno matematiko vključujoč poglavja številčne teorije, informacijske teorije, kompleksnostjo računanja, statistike in kombinatorike. Kriptografija je orodje uporabljeno skupaj z varnostjo računalnikov in omrežij.

1.1 TERMINOLOGIJA

Kriptologija je veda o tajnosti, šifriranju, zakrivanju sporočil (kriptografija) in o razkrivanju šifriranih podatkov (kriptoanaliza). Beseda prihaja iz grščine: kryptos logos pomeni skrita beseda. Uporabljata se še pojma enkripcija (šifriranje) in dekripcija (dešifriranje). Osnovno sporočilo ponavadi imenujemo čistopis (cleartext, plaintext), zašifrirano pa šifropis ali tajnopis(kriptogram, ciphertext)

1.2 ZGODOVINA KRIPTOGRAFIJE

Zgodovinsko, kriptografija se ukvarja le z šifriranjem, kar je v bistvu pretvarjanje informacij iz njenega običajne, razumljive oblike v nerazumljivo obliko, predstaviti jo kot neberljivo, če ne poznamo skrivnosti branja.

Špartanci so uporabljali naslednji način: na valj so navili ozek trak in sporočilo napisali pravokotno na smer traku. Poslali so odvit trak, naslovnik pa je moral imeti valj enakega premera.

Julij Cezar je svojim vojskovodjem pošiljal sporočila, kjer je vsako črko zamenjal s črko, ki je bila v abecedi nekaj mest za njo. Postopek lahko opišemo kot *zamenjavo črk a -> a+k po modulu n* (n pomeni število črk v abecedi). "k" predstavlja ključ. Cezar je menda običajno uporabil ključ 3. Kako bi dešifrirali HAL, če vemo, da smo uporabili isti algoritem s ključem -1?

V začetnih letih svetovnega spleta je bil v uporabi postopek ROT-13 (zamenjava črk a -> a+13 po modulu 26 - angleška abeceda) v Usene-tu za šifriranje neprimernih šal in podobnega. Seveda ni predstavljal nobene resne zaščite. Netscape-ov brskalnik je imel v prvih verzijah vgrajeno možnost pod View -> Unscramble ROT-13.

To sta primera za monoalfabetsko substitucijo, kjer se črka vedno preslika v isto črko, zato frekvenčna porazdelitev črk kaže podoben vzorec kot v jeziku čistopisa. Pri dešifriranju poiščemo najbolj pogoste črke in jih nadomestimo z najbolj pogostimi črkami v jeziku, potem pa iščemo še najbolj pogoste dvojčke, trojčke oziroma besede v tekstu. Prvi opis dešifriranja monoalfabetske substitucije je znan iz devetega stoletja, uporabljala pa se je še v srednjem veku. Nadomestile so jo varnejše polialfabetske substitucije, na primer Vigenčrjeva.

V prvi polovici 20. stoletja so za polialfabetsko substitucijo uporabljali šifrirne stroje. Najbolj znana je Enigma, ki so jo uporabljali Nemci med 2.svetovno vojno. To napravo za šifriranje je razvil nemški elektroinženir Arthur Scherbius (1878-1929). Za izdelek je 23.februarja 1918 prejel patent (DRP 416219). Za izdelavo teh naprav so 9. julija 1923 v Berlinu ustanovili podjetje *Chiffriermaschinen-Aktiengesellschaft*. Proti koncu leta1920 so oborožene sile začele kazati zanimanje za to napravo, kasneje pa je izginila s civilnega trga.

V času do konca vojne 1945 in še kasneje so prihajali v uporabo številni različni modeli Enigme. Najbolj razširjena je bila ENIGMA I, ki so jo od leta 1930 uporabljale nemške oborožene sile Reichswehr in kasneje Wehrmacht Verjetno je bila med 2. svetovno vojno najbolj uporabljan sistem šifriranja sporočil. ENIGMA I je na prvi pogled videti kot pisalni stroj. Bistveni sestavni deli so tipkovnica, zbir zamenljivih valjev in polja z lučkami za prikaz. Valji so bistveni za šifriranje. Vsak valj ima na obeh straneh po 26 električnih spojev (po enega za vsako črko nemške abecede), ki so na tajen način povezani med seboj. Na primer: vhodni spoj za črko A je na izhodu povezan s spojem za črko G in tako naprej. Ko se pritisne tipka na tipkovnici, steče tok iz vgrajene tipkovnice preko valjev do polja z lučkami. Prižgana lučka je šifra za vneseno črko. Ker se valji po vsakem pritisku na tipkovnici zavrtijo, dobi ista črka na različnih mestih odprtega besedila različne šifre.

Če bi odtipkali *SSSR*, bi dobili kot šifrirano besedilo na primer *HUTW*. Važno in za šifriranje izredno pomebno je, da se za razliko od enoabecedne zamenjave (kjer se ista črka odprtega besedila vedno prevede v isto črko šifriranega besedila) pri ENIGMI po vsaki vneseni črki spremeni sistem zamenjave (poligrafska substitucija).

1.3 MODERNA KRIPTOGRAFIJA

Polje moderne kriptografije lahko razdelimo na več manjših enot. Spodaj opisane so glavne, vendar niso edine.

- **Simetrično šifriranje**
- **Asimetrično šifriranje (šifriranje z javnim ključem)**
- **Analiza šifriranja**

Cilj analize šifriranja je najti slabosti ali nevarnosti v kriptografski shemi. Analiza šifriranja se ukvarja z sovražnim napadalcem, ki namerava spodkopati sistem, ali pa poizkuša sistem spodkopati razvijalec sistema, ki želi ovrednotiti in preizkusiti varnost samega sistema. V moderni praksi vsebujejo šifrirne tehnike dokaze, ki dokazujejo varnost sistema in predstavljene z razumljivimi domnevami.

Medtem, ko je cilj šifriranja preprečiti, da bi do čistopisa ali šifrirnega ključa prišli napadalci sistema, pa se analiza šifriranja ukvarja s tem, kako priti do šifrirnega ključa in s tem tudi do vseh sporočil, ki se prenašajo preko komunikacijskega kanala.

Vsak poskus analize šifriranja z namenom pridobivanja podatkov čistopisa ali samega šifrirnega ključa se imenuje napad na sistem.

Poznamo dve vrsti napadov:

1. Pasivni napad je tisti, pri katerem napadalec samo opazuje komunikacijski kanal, pri tem pa je lahko ogrožena zaupnost podatkov. To se zgodi v primeru, ko napadalec ugotovi šifrirni ključ.

2. Aktivni napad pa je tisti, pri katerem napadalec na kakršen koli način spremeni podatke, ki se prenašajo preko komunikacijskega kanala. Pri tem sta poleg zaupnosti ogroženi celovitost podatkov in overovitev.

1.4 ŠIFRIRNI PROTOKOLI

Šifrirni protokoli opisujejo zaporedje korakov, ki jih morajo upoštevati vsi, ki med seboj komunicirajo, če želijo, da so poslana sporočila zavarovana. Vsi udeleženci morajo protokol poznati in se z njim strinjati. Protokol mora biti nedvoumen in sposoben razrešiti vsako možno situacijo, ki se lahko pojavi med njegovim potekom.

Poznamo več vrst šifrirnih protokolov:

- protokoli, kjer mora biti vedno udeležena še tretja oseba
- protokoli, kjer je tretja oseba udeležena le, če pride med uporabnikoma do spora
- protokoli, kjer navzočnost tretje osebe ni potrebna

2 ASIMETRIČNO ŠIFRIRANJE

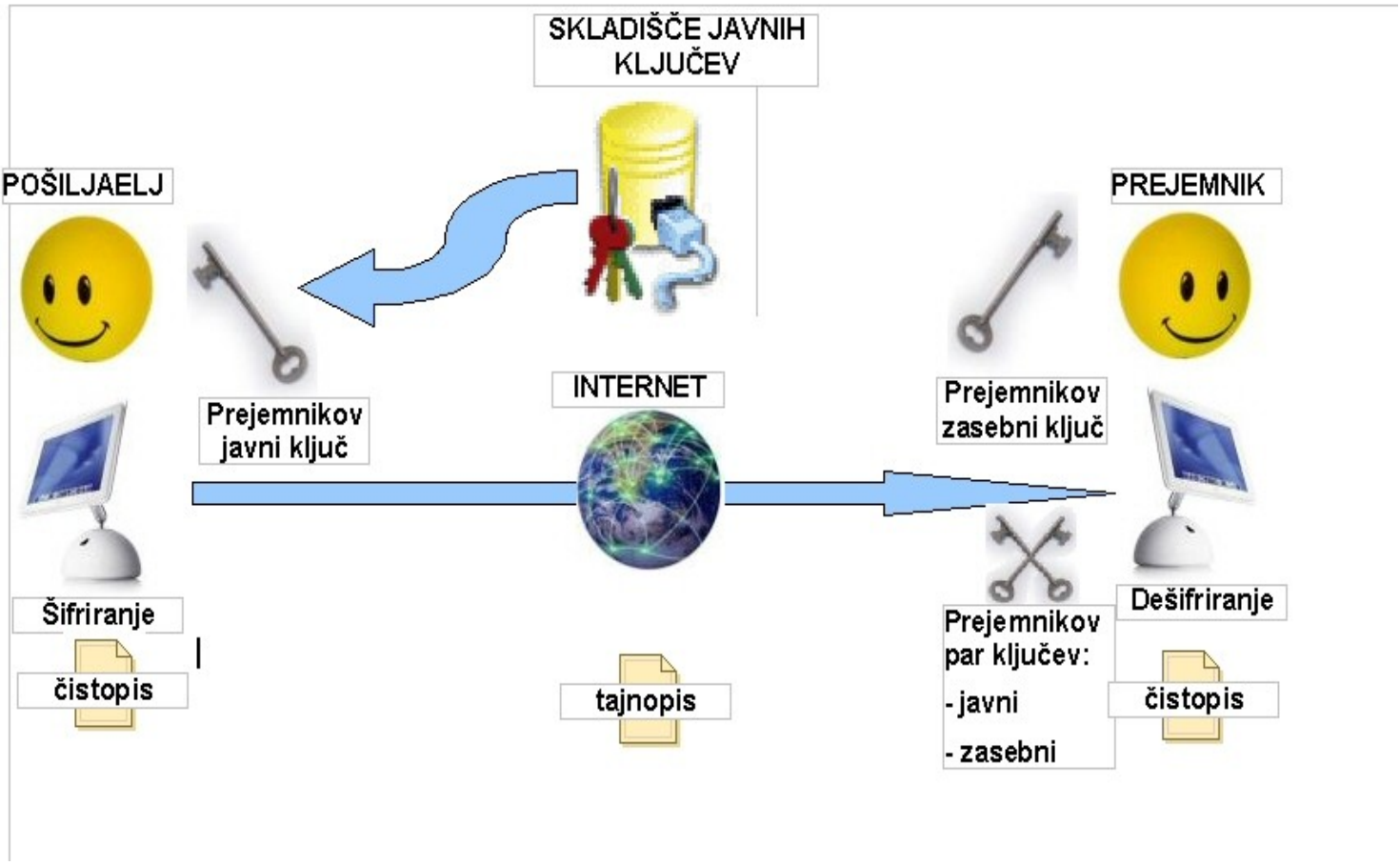
Izraz asimetrično šifriranje je v večini primerov sinonim za šifriranje z javnim ključem. Poleg omenjenega pa obstajajo tudi algoritmi asimetričnih šifriranj, ki ne vsebujejo lastnosti javnega-zasebnega ključa. Pri teh algoritmih morata oba ključa ostati skrivnost.

Šifriranje z javnim ključem je oblika šifriranja, ki v glavnem omogoča varno komunikacijo, ne da bi imeli predhodno dodeljen zasebni ključ. Šifriranje je izvedeno z uporabo para šifrirnih ključev imenovanih javni ključ in zasebni ključ, ki sta matematično sorodna.

Pri šifriranju z javnim ključem je zasebni ključ v glavnem skrit, medtem ko naj bi bil javni ključ široko razširjen. To deluje na takšen način, da eden ključ zaklene ključavnico, medtem ko jo drugi odklene. Iz podanega javnega ključa, ki je eden v paru podanih ključev, pridobitev zasebnega ključa ne sme biti možna

Poznamo več oblik šifriranja z javnim ključem:

- **šifriranje z javnim ključem –** sporočilo ostane skrito za vse, ki ne posedujejo specifičnega privatnega ključa
- **digitalni podpis z javnim ključem –** dovoljuje vsem da lahko preverijo pristnost podpisa specifičnega privatnega ključa.
- **izmenjava ključev –** protokol, ki dovoljuje izmenjavo ključev preko nezaščitenega medija, brez predhodnega tajnega dogovora.



Slika 1: Prikaz asimetričnega šifriranja

Tehnike z javnim ključem so bolj računsko intenzivne kot popolnoma simetrični algoritmi, vendar je uporaba teh tehnik razširjena v širokem spektru aplikacij.

2.1 ZGODOVINA

V glavnem se v zgodovini šifriranja govori o ključu, ki mora biti absolutno skrit in bo predhodno določen z uporabo varne, vendar ne-šifrirane metode. Taka metoda je lahko neposreden sestanek ali zaupen kurir. Takšen način razdeljevanja ključev pa prinese s seboj veliko praktičnih težav.

Zgoraj navedena slabost šifriranja pa je pripeljala do iznajdbe šifriranja z javnim ključem. S šifriranjem z javnim ključem lahko uporabniki komunicirajo preko nezaščitenega kanala, ne da bi predhodno imeli pripadajoč zasebni ključ.

Clifford Cocks je v 70-tih prvi izumil algoritem asimetričnega šifriranja, kasneje pa so pri razvoju pomagali tudi matematični diplomiranci in novi člani na GCHQ (Government Communications Headquarters) v Veliki Britaniji. To je ostalo skrito do leta 1997.

Leta 1976 sta Whitefield Diffie in Martina Hellman objavila šifriranje z asimetričnim ključem. Avtorja sta šifriranje razvijala pod vplivom dela Ralpha Merkleja, ki je razširjalo javni ključ in metodo izmenjave javnega ključa. Ta metoda eksponentne izmenjave ključev, ki ga poznano kot izmenjava ključev po algoritmu Diffie-Hellman je bila prva objavljena praktična metoda za vzpostavljanje skupnega ključa preko nezaščitenega komunikacijskega kanala, ne da bi uporabljali predhodni tajni dogovor. Merkle-jeva tehnika izmenjave javnega ključa poznana kot Merklejev-a sestavljenka je bila objavljena leta 1978.

Rivest,Shamirja in Adleman so leta 1977 na MIT (Massachusetts Institute of Technology) ponovno obdelali Cocksova metodo. Leta 1978 so avtorji izdali njihovo delo in algoritem je postal znan kot RSA. RSA izkorišča težavnost produkta dveh velikih primarnih števil za šifriranje in dešifriranje. Pri šifriranju z javnim ključem in digitalnim podpisom z javnim ključem je njegova varnost je povezana z predpostavljeno težavnostjo razstavljanja velikih števil Za reševanje tega primera v glavnem nimamo zadovoljivih (v praksi prepočasnih) tehnologij.

Od leta 1970 naprej je prišlo do velikega števila različic šifriranja, digitalnega podpisa, izmenjave ključev in drugih tehnik na področju šifriranja z javnim ključem. ElGamal šifrirni sistem izumitelja Taher ElGamal-a se nanaša na težavnost problema diskretnega algoritma. Kot ena izmed variant ElGamalovega algoritma je tudi algoritem digitalnega podpisa (ang.DSA - Digital Signature Algorithm) ki je bil razvit s strani NSA (National Security Agency) in NIST - (National Institute of Standards and Technology) . Predstavitev šifriranja z eliptično krivuljo, ki jo je predstavil Neal Koblitz sredi 80tih je obrodilo novo družino analognih algoritmov z javnim ključem. Čeprav je matematično bolj zahtevna, se zdi, da eliptična krivulja določa bolj zadovoljivo pot s vplivom problema diskretnega algoritma, posebej zaradi spoštljive velikosti ključa.

2.2 VARNOST

Z vidika varnosti, ni pri sistemih z asimetričnim šifriranjem nič več varnosti, kot pri sistemih z simetričnim šifriranjem. Obstaja nekaj popularnih in nekaj nepopularnih različic. Nekaj algoritmov je že razbitih, nekaj, za zdaj, ji še ni razbitih. Na nesrečo popularnost ni zanesljiv pokazatelj varnosti. Nekateri algoritmi imajo varnostna zagotovila z velikim številom posebnosti in spreminjajočo kvaliteto. Veliko dokazov trdi, da je razbijanje algoritma, za razliko do nekaterih dobro definiranih varnostnih ciljev, ekvivalentno rešitvi zelo težavnih matematičnih problemov, ki se smatrajo za neukrotljive, kot so razstavljanje velikih števil ali iskanje diskretnih logaritmov. V glavnem noben za nobenega od teh algoritmov še ni bil dokazan kot absolutno varen v smislu enkratnega ščita. Kot vsi šifrirni algoritmi morajo tudi ti biti skrbno izbrani in uporabljeni.

2.3 UPORABA

Najbolj pogosta uporaba sistema šifriranja z javnim ključem je zagotavljanje zaupnosti. Sporočilo, ki ga pošiljatelj zašifrira z uporabo prejemnikovega javnega ključa je lahko dešifrirano samo s parom prejemnikovega privatnega ključa.

Algoritem digitalnega podpisa z javnim ključem lahko uporabljamo za overovitev pošiljatelja. Postopek je tak, da uporabnik zašifrira sporočilo s svojim privatnim ključem in to sporočilo pošlje. Če prejemnik lahko uspešno dešifrira sporočilo z uporabo ustreznega javnega ključa, potem to dokazuje in jamči, da je to pošiljatelj pravi.

Te značilnosti so uporabne na veliko področjih, kot so digitalni denar, izmenjava z geslom zaščitenega ključa, itd.

2.4 PRAKTIČNI VIDIK

2.4.1 Podobnost dopisnicam

Podobnost ki jo lahko uporabimo za razumevanje prednosti asimetričnega sistema je, da si predstavljamo dve osebi, osebo A in osebo B, ki si pošiljata skrivno sporočilo preko javne pošte. V tem primeru ima oseba A skrivno sporočilo in ga želi poslati osebi B, na katero oseba B odgovori s skrivnim sporočilom

S pomočjo simetričnega sistema, oseba A najprej odda skrivno sporočilo v škatlo in škatlo zaklene s ključavnico in ključem ki ga ima. Škatlo pošlje osebi B preko navadne pošte. Ko oseba B prejme škatlo, jo odklene s ključem, ki je identična kopija ključa osebe A (le tega sta si predhodno izmenjala) in prebere sporočilo. Oseba B lahko za odgovor s skrivnim sporočilom uporabi isto škatlo in isto ključavnico.

Pri asimetričnem sistemu imata oseba A in oseba B ločeni ključavnici. Najprej oseba A prosi osebo B, naj ji pošlje odprto ključavnico po navadni pošti, in ključ obdrži pri sebi. Ko Ana prejme ključavnico z njo zaklene škatlo v kateri je sporočilo in pošlje osebi B. Po prejemu jo lahko oseba B odklene z njegovim ključem in prebere sporočilo osebe A. Za odgovor mora oseba B prav tako dobiti odprto ključavnico osebe A, da zaklene škatlo preden jo pošlje osebi A.

Največja prednost asimetričnega sistema je, da si osebi A in B nikoli ne pošljeta kopije njunih ključev drug drugemu. To utemeljeno zmanjšuje možnost, da tretja oseba prekopira ključ, medtem ko je poslan in lahko tako nepooblaščeno pregleduje vsa sporočila, ki si jih v prihodnje pošiljata osebi A in B. V primeru, da postane oseba B neprevidna in si nekdo skopira njegov ključ, postanejo izpostavljena samo sporočila, ki jih pošilja oseba A osebi B, sporočila, ki jih oseba A pošlje ostalim ljudem, pa ostanejo skrita, ker ostali uporabljajo drugačne ključavnice pri komuniciranju z osebo A.

2.4.2 Dejanski algoritmi – dva povezana ključa

Vsi asimetrični algoritmi pa ne delujejo točno po tem postopku. Najbolj razširjena značilnost je, da imata osebi A in B vsak po dva ključa, enega za šifriranje in drugega za dešifriranje. V varni shemi asimetričnega šifriranja ne smemo ključa za dešifriranje izpeljati iz ključa za šifriranje. Tak način šifriranja poznamo kot šifriranje z javnim ključem, ker je lahko ključ za šifriranje vsem poznan, ne da bi s tem ogrožali varnost šifriranega sporočila. Glede na prejšnji primer bi lahko oseba B objavila navodila, kako narediti ključavnico javni ključ, ta ključavnica pa je takšna, da iz nje ni razvidno kakšen naj bi bil ključ, ki jo odklepa privatni ključ. Tisti, ki želijo poslati osebi B skrivno sporočilo uporabijo javni ključ za šifriranje sporočila, oseba B pa ga dešifrira s svojim privatnim ključem.

2.4.3 Slabosti

Seveda je tukaj tudi možnost, da lahko nekdo vzame ključavnico osebe A ali osebe B. V nasprotju z enkratnim ščitom ali njemu podobnim ni nobenega asimetričnega algoritma, ki bi bil dokazan kot odporen na matematične napade. Se pravi, da ni nemogoče da pride do neke relacije med ključema v paru, ali slabosti v operaciji algoritma.

To lahko omogoči dešifriranje brez uporabe ključa, oz. samo z uporabo ključa za šifriranje. Varnost algoritma asimetričnega šifriranja temelji na domnevi stopnje težavnosti rešitve osnovnega matematičnega problema. Ta ocena se spreminja s padanjem cene računalniške opreme in z novimi matematičnimi odkritji.

Kljub vsemu naštetemu so algoritmi še vedno dovolj močni. Če predvidevamo, da bo trajal nasilen vdor v kodo 1000 let, potem bi bilo to šifriranje za npr. kreditne kartice izredno varno – saj je tako čas vdora veliko daljši od časa trajanja kreditne kartice.

V preteklosti je bilo odkritih nekaj slabosti v uporabi asimetričnega šifriranja. Algoritem 'pakiranja nahrbtnika' se je izkazal za neučinkovitega, ko je bil nepričakovano napaden. Nedavno se je pojavilo nekaj napadov, ki so temeljili na merjenju točne porabe časa poznane strojne opreme za dešifriranje preprostega besedila pri poenostavljanju iskanja verjetnega ključa za dešifriranje. Na ta način, uporaba algoritma asimetričnega šifriranja ne zagotavlja varnosti. To je tudi področje aktivnih raziskav za odkrivanje in preprečevanje novih in nepričakovanih napadov.

Naslednja potencialna nevarnost v procesu uporabe asimetričnega šifriranja je možnost napada 'človeka v sredini'. Pri tem napadu tretja oseba prisluškuje komunikaciji z javnimi ključi. Ta oseba lahko komunikacijo spreminja in tako si priskrbi svoj javni ključ. Da se ta oseba izogne sumu mora dešifrirati vsa sporočila, tudi odgovor, in jih potem nazaj zašifrirati. Napad ni nemogoč in škodoželjen član na ponudniku storitev oseb A in B lahko škoduje. Ta oblika napada je bila naslovljena s strani razvoja distribucijskih metod ključa ki lahko zagotavljajo overovitev pošiljatelja in celovitost sporočila tudi preko nezaščitenih kanalov. Kot napadalec je v tem primeru lahko tudi ena od višjih državnih ustanov. Ona ima potencialno moč da prepriča overovatelja certifikatov, da podpiše ponarejen javni ključ. Nato lahko ta ustanova na strani osebe B odklopi ponudnika internetnih storitev, in na to mesto postavi ponarejen strežnik. Funkcija tega strežnika je, da se predstavlja kot oseba A, preverjeno s popravljenim certifikatom, pobere vsa sporočila in jih posreduje pravemu strežniku osebe A.

2.4.4 Stroški računanja

Večina algoritmov javnih ključev je računsko poceni v primerjavi z nekaterimi algoritmi simetričnega šifriranja z enako stopnjo varnosti. To dejstvo ima velik delež za njihovo praktično uporabo. Največ jih uporabljamo v hibridnih šifrirnih sistemih zaradi učinkovitosti. V takem šifrirnem sistemu je skrivni ključ v seji ustvarjen z ene strani, ta, krajši sejni ključ, je nato šifriran z javnim ključem posameznih prejemnikov in to z vsakim posebej. Za dešifriranje sejnega ključa vsak posamezen prejemnik uporablja pripadajoč privatni ključ. Ko vsi deli prejmejo sejni ključ, lahko uporabljajo veliko hitrejši simetrični algoritem za šifriranje in dešifriranje sporočil.

2.4.5 Združevanje javnih ključev po podobnosti

Povezava med javnim ključem in njegovim lastnikom mora biti pravilna, da bi algoritem pravilno deloval in bi bil v praksi popolnoma zanesljiv. Kot pri večini šifriranja, so protokoli uporabljeni za vzpostavitev in preverjanje te povezave zelo pomembni. Združevanje javnega ključa in njegovega lastnika je ponavadi narejeno s protokoli, ki izpopolnjujejo infrastrukturo javnega ključa. Ti protokoli potrjujejo pravnomočnost združljivosti, da jo lahko formalno dokažemo z referenco zaupanja tretji osebi, ki je na obeh straneh v obliki hierarhije pooblaščanja s certifikati (npr. X.509), model lokalnega zaupanja (npr. SPKI), ali statistične strani zaupanja. (npr. PGP in GPG) .

Šifrirno jamstvo protokolov, združevanje javnih ključev in njihovih lastnikov je konec koncev zadeva subjektivne presoje na strani zaupanja vredni tretji osebi, ključ pa je matematično bistvo, medtem ko to ni lastnik in povezava med njim in ključem. Iz tega razloga, formalnost infrastrukture javnega ključa predpisuje nedvoumne izjave pravil pri pripravi obrazložitve. Kot primer lahko povemo, da.509 standard dovoljuje veljavnost certifikatov za zavarovanje z namenom identifikacije objektov katerih funkcije so indeksi v zbirki registriranih zavarovanj. Zavarovanja obstajajo za veliko različnih namenov, v območju anonimnosti do vojaških vsebin.

2.4.6 Povezava z realnim časom

Javni ključ pozna velika in verjetno neznana množica uporabnikov. Vsi dogodki povezani z varnostjo z zahtevo po javnem ključu, ki zahtevajo preklic ali zamenjavo lahko za zaključitev porabijo veliko časa. Iz tega razloga morajo biti sistemi sposobni reagirati na dogodke v realnem času (varnostno kritični sistemi) in ne smejo potrjevati šifriranja z javnim ključem, ne da bi temu posvetili več pozornosti.

2.4.7 Prednost preklica ključa

Škodljiv ali napačen preklic nekaj ali vseh ključev v sistemu je enak povzročitvi izpada sistema. To je vedno možno, če ključe lahko prekličemo posamezno. Vseeno pa je nekaj, kar lahko zmanjša to možnost. Z vzpostavitvijo certifikatov lahko ustvarimo t.i. „sestavljenega vodjo“. Eden od takih vodij bi bil „Avtoriteta preklica oseb A in B“. Tako lahko ključ prekličeta le oseba A in oseba B skupaj, nikakor pa ne vsak posamezno. Na ta način se zahteva prisotnost obeh oseb, kar poveča zanesljivost. V konkretnem primeru, imamo iz varnostnega vidika samo ena točko izpada sistema – uspešna zavrnitev storitve (ang. DOS - „Denial of Service“) napad, ki bi za eno od oseb, ali obe paraliziral avtoriteto preklica. Dejstvo je, da ime vsaka pregrada med osebama tak efekt, ne glede na to, kako pride do tega.

Avtoriteta za preklic ključa je zelo močan mehanizem in mora vplesti čim več udeležencev za preprečevanje škodljivih napadov. Po drugi strani pa mora vplesti čim manj udeležencev, da lahko zagotovimo, da bo ključ preklican brez zakasnitev.

2.4.8 Razširjanje novega ključa

Po tem, ko je bil ključ preklican moramo nov ključ razširiti po predhodno definirani metodi. Vzemimo, da je bil ključ osebe preklican. Dokler nov ključ ni razširjen je razpoložljivost osebe onemogočena. Nihče ni sposoben poslati njenih podatkov brez prekršitve varnosti sistema. Prav tako bodo podatki, ki prihajajo s strani te osebe zavrnjeni iz istega razloga. Ali z drugimi besedami, del sistema, ki ga nadzoruje oseba je izključena in tako ni na voljo. V tem primeru je na višjem mestu potreba po varnosti, kot potreba po razpoložljivosti.

Lahko se zgodi, da nekdo združi avtoriteto za izgradnjo novega ključa in ga certificira z avtoriteto za preklic ključa, vendar ni potrebe po temu. Pravzaprav je to iz varnostnih pogledov slaba odločitev. Problem nastane, ko mora po eni strani sporočilo, ki preklicuje ključ biti poslano kar se da hitro, medtem ko so lahko deli sistema paralizirani še preden je nov ključ instaliran. Časovni okvir lahko postavimo na 0 tako, da vedno izdamo nov ključ skupaj s certifikatom, ki preklicuje star ključ, vendar to spet zahteva so-lokacijo avtoritete ki prekliče ključ in drugo ki ponovno zažene sistem.

2.4.9 Razširjanje preklica ključa

Obvestilo o preklicu ključa in prenehanju uporabe le tega mora biti dostavljeno vsem, ki potencialno imajo ključ in to v najkrajšem možnem času.

Imamo dva načina razširjanja informacije - preklica v porazdeljenih sistemih. Lahko informacijo vrinemo (push) uporabnikom iz centralnega mesta, ali pa uporabniki iz centralnega mesta povlečejo (pull) informacijo.

Če hočemo sporočilo poslati vse sodelujočim je najbolje, da informacijo vrinemo. Kljub temu pa nimamo informacije o tem, ali je prejemnik res prejel sporočilo in če je število sodelujočih in njihova fizična razdalja izven razumnih mej je možnost uspeha tega pristopa zelo nizka. V tem stanju je sistem ranljiv za zavrnitev storitve in varnost je zlomljena in okno ostane odprto dokler ni zadnjega sporočila. Z drugimi besedami – vrinjanje ni varno in ni zanesljivo.

Alternativa vrinjanju je vlečenje. V tem primeru so vsi ključi znotraj certifikata, ki je zahtevan za verifikacijo pravilnega ključa. V tem primeru je problem, ko je uporabnik blokiran, če ne more doseči servisa za overovitev. Ta servis je ima lahko variabilno zanesljivost, večanje zanesljivosti pa gre na ceno varnosti (več kot je strežnikov za nadgradnjo, dlje je okno odprto).

Naslednja možnost je uporaba nekaj manj zanesljivega, vendar bolj varnega servisa za overovitev, to je izdaja certifikata z določeno življenjsko dobo.

2.4.10 Ponovna vzpostavitev preklicanega ključa

Domnevno, da se je objekt avtoriziran za preklic ključa odločil, da mora biti določen ključ preklican. V večini primerov se to zgodi po določenem dejstvu, ko se je razvedelo da se je v preteklosti sprožil dogodek ki je ogrozil tajni del javnega ključa. Označimo čas, ko je prišlo do te odločitve s T.

Odločitev ima dva dela: Sporočilo šifrirano z javnim ključem po času T ne moremo več smatrati za skrivnega in podpisov narejenih s ključem po času T, ne moremo več smatrati za avtentične brez temeljitega spremljanja dogodkov od tam naprej, kjer je bil podpis narejen.

Če izgubi tajnost in ali avtentičnost je to široka odpoved sistema, tu je na mestu strategija obnovitve. Ta strategija vsebuje podatke o tem, kdo ima avtoriteto za preklic ključa, kako razširiti preklic in tudi podatke o tem, kako obdelovati sporočila ki so bila šifrirana s ključem po času T. Ta procedura obnavljanja je lahko kompliciranа in medtem ko se izvaja je sistem ranljiv v smislu napada zavrnitve storitve.

3 DIFFIE-HELLMAN

Protokol, imenovan Diffie-Hellman protokol za izmenjavo in kreiranje ključev, je kriptografski protokol, ki omogoča dvema ali več strankam, za katere ni nujno da se predhodno poznajo, sestavo skupnega skrivnega ključa preko nezavarovanega komunikacijskega kanala. Ta ključ lahko potem uporabimo za šifriranje sporočil z algoritmom simetričnega šifriranja.

Pod imenom Diffie-Hellman izmenjava ključa poznamo:

- Diffie-Hellman sporazum o ključu
- Diffie-Hellman kreiranje ključa
- Diffie-Hellman pogajanje ključa
- Pojasnilo o izmenjavi ključa

Postopek sta Whitfield Diffie in Martin Hellman prvič objavila leta 1976. Decembra 1997 je postalo znano, da so angleški znanstveniki James Ellis, Clifford Cocks in Malcolm Williamson, zaposleni v tajni službi angleške vlade GCHQ (Government Communications Headquarters), odkrili postopek enak postopku Diffie Hellman več let pred Diffiejem in Hellmanom, niso pa smeli tega objaviti, ker je šlo za vojaško skrivnost. Leta 2002 je Hellman predlagal ime algoritma Diffie-Hellman-Merkle izmenjava ključev, kot spomin na prispevek Ralpha Merkle-ja pri tem

Čeprav je izmenjava ključa po Diffie-Hellman-u samo po sebi anonimen (ne-overovljen) protokol izmenjave ključev, predstavlja temelj za različne protokole overovitve in je uporabljen kot predpis popolne napredne diskretnosti v kratkotrajnih načinih TLS.

3.1 ZGODOVINA PROTOKOLA

Protokol Diffie-Hellman, namenjen izmenjavi ključev, je bil iznajden leta 1976 med sodelovanjem Winfield Diffie in Martinom Hellmanom in je bila prva metoda v praksi za prenos tajnih podatkov preko nezaščitenega komunikacijskega kanala. Na to je vplivalo tudi delo Ralph-a Merkle-ja pri razširjanju javnega ključa. John Gill je predlagal aplikacijo problema diskretnega algoritma. Iznašel ga je Malcom-a Williamson-a iz GCHQ v Veliki Britaniji nekaj let prej vendar se je GCHQ odločil da tega ne bo objavil in to je veljalo do leta 1997. Kmalu zatem se je pojavila metoda RSA, še ena izvedba kriptografije javnega ključa z uporabo asimetričnih algoritmov.

3.2 OPIS

Najenostavnejša in originalna izvedba protokola uporablja množilno skupino števil modula p kjer je p praštevilo in g generator, ki je celo število, manjše od p. Modulo pomeni da, lahko dobimo katerokoli število od 1 do p-1, če ga potenciramo in vzamemo vrednost po modulu p.

Oseba A		
Zasebno		Izračuna
	p, g	
a		
		$\mathbf{g^a}$ mod p
	…	
	($\mathbf{g^b}$ mod p)$^{\mathbf{a}}$ mod p	

→
←
=

Oseba B		
Izračuna		Zasebno
	p, g	
		b
	…	
$\mathbf{g^b}$ mod p		
	($\mathbf{g^a}$ mod p)$^{\mathbf{b}}$ mod p	

Tabela 1: Primer izmenjave ključev

Praktičen primer protokola:

1. Oseba A in oseba B se dogovorita, da bosta uporabljala praštevilo p=23 in generator g=5
2. Oseba A si izbere tajno število a=6 in pošlje B (g^a mod p)
   - 5^6 mod 23 = 8.
3. Oseba B izbere tajno število b=15 in pošlje A (g^b mod p)
   - 5^{15} mod 23 = 19.
4. Oseba A izračuna g^b mod $p)^a$ mod p
   - 19^6 mod 23 = 2.
5. Oseba B izračuna (g^a mod $p)^b$ mod p
   - 8^{15} mod 23 = 2.

Obe osebi sta prišli do iste vrednost, ker sta g^{ab} in g^{ba} enaka. Edino *a, b in* $g^{ab} = g^{ba}$ ostane tajno. Vse ostale vrednosti se pošljejo takšne kot so. Ko A in B izračunata skupni ključ, ki ga uporabljata kot šifrirni ključ, ki ga poznata samo onadva, za pošiljanje sporočil preko istega odprtega komunikacijskega kanala. Seveda pa potrebujemo večje vrednosti *a,b* in *p* da naredimo ta primer varen, saj je lahko poizkusiti vse možne vrednosti g^{ab} mod 23 (tukaj je samo 22 takih vrednosti, tudi če sta *a* in *b* veliki števili). Če je *p* praštevilo večje od 300 števk in *a* in *b* večja od 100 števk, potem celo najboljši poznani algoritmi za iskanje *a* s podanimi samo *g.p,* in g^a mod p (poznanimi kot problem diskretnega logaritma) trajajo dlje kot je npr življenjska doba ustanove, ki bi to izvajala. Število *g* je lahko majhno in je ponavadi 2 ali 5.

Splošen opis protokola:

1. Osebi se dogovorita o zaključeni končni skupini G in generiran element g v G (To je ponavadi narejeno daleč prej kot ostali protokol, g je izmišljen in poznan vsem napadalcem). Skupino G bomo ustvarjali množilno.
2. Oseba A izbere naključno naravno število a in pošlje g^a osebi B
3. Oseba B izbere naključno naravno število b in pošlje g^b
4. Oseba A izračuna $(g^b)^a$
5. Oseba B izračuna $(g^a)^b$

Obe osebi imata sedaj v lasti element g^{ab} , ki lahko služi kot skupen tajni ključ. Vrednosti $(g^b)^a$ in $(g^a)^b$ sta enaki, zaradi asociativnosti.

3.3 DIAGRAM

Spodaj je diagram, ki pojasnjuje, potek izmenjave ključev in možnost prisluškovanja. P je prisluškovalec, ki prisluškuje, kaj je poslano med A in B, vendar ne spreminja vsebine njunih komunikacij.

Spremenljivke:

- s = skupen tajni ključ. S = 2
- a = privatni ključ osebe A. a = 6
- b = privatni ključ osebe B. b = 15
- g = javni generator. b = 15
- p = javno praštevilo. p = 23

Oseba A	
Pozna	Ne pozna
p = 23	b = 15
base g = 5	
a = 6	
$5^6 \bmod 23 = 8$	
$5^b \bmod 23 = 19$	
$19^6 \bmod 23 = 2$	
$8^b \bmod 23 = 2$	
$19^6 \bmod 23 = 8^b \bmod 23$	
s = 2	

Oseba B	
Pozna	Ne pozna
p = 23	a = 6
base g = 5	
b = 15	
$5^{15} \bmod 23 = 19$	
$5^a \bmod 23 = 8$	
$8^{15} \bmod 23 = 2$	
$19^a \bmod 23 = 2$	
$8^{15} \bmod 23 = 19^a \bmod 23$	
s = 2	

Prisluškovalec	
Pozna	Ne pozna
p = 23	a = 6
base g = 5	b = 15
	s = 2
$5^a \bmod 23 = 8$	
$5^b \bmod 23 = 19$	
$19^a \bmod 23 = s$	
$8^b \bmod 23 = s$	
$19^a \bmod 23 = 8^b \bmod 23$	

Tabela 2: Potek izmenjave ključev in možnost prisluškovanja.

Za osebo A mora biti privatni ključ osebe B težko dostopen in prav tako mora biti za osebo B težko dostopen privatni ključ osebe A. Če temu ni tako, potem lahko prisluškovalec izračuna svoj par privatnega in javnega ključa, vključi javni ključ osebe B v svoj privatni ključ, ustvari ponarejen skupni tajni ključ in ugotovi privatni ključ osebe B in to uporabi pri ugotavljanju skupnega tajnega ključa. Prisluškovalec bo poizkusil izbrati par javnega in privatnega ključa, tako, da bo lahko z lahkoto ugotovila privatni ključ osebe B.

3.4 VARNOST

Protokol velja za varnega proti prisluškovanju, če pravilno izberemo *G* in *g*. Prisluškovalec mora rešiti Diffie-Hellma-ov problem, da pride do g^{ab}. To se smatra za težko izvedljivo. Zmogljiv algoritem za reševanje diskretnega logaritma, bi omogočil enostaven izračun *a* in *b* in bi rešil Diffie-Hellmanov problem in s tem naredil protokol nezaščiten.

Praviloma moramo za *G* izbrati praštevilo ali pa mora imeti za faktor veliko praštevilo, da dobimo zaščito pred izračunom *a* in *b* z uporabo Pohlig-Hellmanovega algoritma. Iz tega razloga uporabljamo praštevilo *q* po izreku Sophie Germain za izračun $p=2q+1$, tako imenovano varno praštevilo. Iz tega sledi, da je *G* deljiv samo z 2 in *q*. *g* je tako pogosto izbran za ustvarjanje *q* podmnožice *G,* namesto *G,* tako da Legendrejev simbol g^a nikoli ne odkriva spodnjega dela *a*.

Če oseba A in oseba B za generator uporabljata naključni števili katerih rezultat ni popolnoma naključen, oziroma je lahko to število v nekem obsegu, potem je delo prisluškovalca veliko lažje.

Skriti števili *a* in *b* sta na koncu seje brezpredmetni. Po zaključku algoritma Diffie-Hellman izmenjava ključev sama po sebi odstrani skriti števili, kar izboljšuje varnost, saj ne obstaja noben zasebni material, ki bi ga lahko kdorkoli zlorabil.

3.5 OVEROVLJENJE

V originalnem opisu protokola Diffie-Hellman, izmenjava sama po sebi ne zagotavlja overovitve udeležencev in je na ta način je nezaščiten pred napadom prisluškovalca. Prisluškovalec lahko vzpostavi dva različna ključa Diffie-Hellman, enega z osebo A in drugega z osebo B, in se nato poskuša maskirati kot oseba A osebi B in obratno, največ zaradi dešifriranja in ponovnega šifriranja sporočil poslanih med njima. Tukaj je potrebno vpeljati nekatere metode overjanja.

Izmenjava ključev Diffie-Hellman vsebuje različne načine overjanja. Ko imata osebi A in B infrastrukturo javnega ključa lahko digitalno popišeta dogovorjen ključ, ali g^a ali g^b, kot v protokolih MQV, STS in IKE, ki so komponente IPsec nizov protokolov za varne IP komunikacije. Ko si osebi A in B izmenjujeta geslo, potem naj bi uporabljala ključ, ki je overovljen z geslom protokola Diffie-Hellman.

4 RSA

RSA algoritem spada v družino algoritmov za šifriranje z javnim ključem. Je prvi algoritem, praktično primeren za podpisovanje in šifriranje in ena prvih prednosti šifriranja z javnim ključem. RSA se na široko uporablja v protokolih namenjenim komercialnim komunikacijam in velja kot varen, če je le dolžina ključev dovolj velika.

4.1 ZGODOVINA

Algoritem so leta 1977 opisali Ron Rivest, Adi Shamir in Len Adleman na tehnološkem inštitutu Massachusetts-a (ang. *Massachusetts Institute of Technology, MIT*). Črke RSA so začetnice njihovih imen.

Clifford Cocks, britanski matematik zaposlen pri GCHQ, je predstavil ekvivalenten sistem v notranjem dokumentu leta 1973. Predstavljen sistem bi potreboval relativno drage računalniške komponente kar je povzročalo pomisleke in, kot je znano se sistem ni razširil. Ponovno je bil sistem obravnavan leta 1997 in je bil razvrščen v razred stroge tajnosti.

MIT je algoritem patentiral leta 1983 v ZDA kot „*U.S. Patent 4,4050,829"*. Prenehal je veljati 21.9.2000. Do tega časa je bilo treba v ZDA plačevati licenčnino, zato algoritem ni bil vključen v nekatere produkte.

4.2 POSTOPEK

4.2.1 Predstavitev

RSA vsebuje dva ključa: javnega in zasebnega (ključ je konstantno število, ki ga kasneje uporabljamo v formuli za šifriranje). Javni ključ poznajo vsi in se uporablja za šifriranje sporočil. Sporočila lahko dešifriramo samo s privatnim ključem. Z drugimi besedami: vsi lahko sporočilo zašifrirajo, prebere pa ga lahko samo lastnik privatnega ključa.

RSA tako kot vsi asimetrični šifrirni postopki, temelji na principu enosmerne funkcije in izkorišča težavnost faktorizacije velikih števil.

Praktičen primer:

Oseba A pošlje osebi B škatlo z odprto ključavnico za katero ima ključ samo oseba A. Oseba B prejme škatlo in vanjo položi sporočilo napisano v preprostem jeziku. Škatlo zaklene z ključavnico osebe a in ji jo pošlje. Oseba B pošlje škatlo osebi A, kjer jo ta odpre s svojim ključem. V tem primeru je škatla s ključavnico javni ključ osebe A in ključ za to ključavnico je njen privatni ključ.

4.2.2 Kreiranje ključev

Predvidevamo, da osebi A in B komunicirata preko nezaščitenega prenosnega medija in oseba A želi osebi B poslati zasebno ali tajno sporočilo. Z uporabo RSA bo oseba A izvedla naslednje korake za kreiranje javnega in zasebnega ključa:

- Izbere dve praštevili, *p* in *q* tako da velja $p \neq q$. Izbere ju tako, da sta naključni in neodvisni druga od druge.
- Izračuna $n = pq$.
- Izračuna koeficient $\phi(n) = (p-1)(q-1)$.
- Izbere celo število *e*, tako da velja $1 < e < \phi(n)$, ki je tuje število številu $\phi(n)$
- Izračuna *d*, tako da je $de \equiv 1 \pmod{\phi(n)}$

  - praštevila lahko s preizkušanjem predhodno testiramo
  - korake 4 in 5 lahko izvedemo z razširjenim Euclidean-ovim algoritmom
  - korak 5 lahko izvedemo tudi z iskanjem celega števila *x*, s pomočjo katerega dobimo celo število $d = \frac{x(p-1)(q-1)+1}{e}$, ki ga potem uporabimo za $d \mod (p-1)(q-1)$

Javni ključ je sestavljen iz:
- *n,* generator
- *e,* zasebni eksponent (včasih eksponent šifriranja)

Zasebni ključ je sestavljen iz:
- *n,* generatorja, ki je javen in se pojavlja v javnem ključu
- *d,* zasebna komponenta (včasih komponenta dešifriranja), ki mora ostati tajna

Ponavadi shranimo različno obliko zasebnega ključa (z upoštevanjem CRT parametrov)
- *p* in *q* sta praštevili iz kreiranega ključa
- *d mod(p-1)* in *d mod (q-1)* pogosto poznana kot *dmp1* in *dmq1*,
- *(1/q) mod p* največkrat poznan kot *iqmp*

Ta oblika dovoljuje hitrejše dešifriranje in podpisovanje z uporabo teorema kitajskih ostankov (CRT Chinese Remainder Theorem). V tej obliki morajo biti skriti vsi deli privatnega ključa.

Oseba A pošlje javni ključ osebi B , pri tem pa skrbi za tajnost zasebnega ključa. *p* in *q* sta občutljiva, ker sta faktorja *n* in omogočata izračun *d* s podanim *e*. Če *p* in *q* nista spravljena v CRT obliki zasebnega ključa, sta varno izbrisana, prav tako, kot so izbrisane tudi ostale vrednosti, ki so bile ustvarjene pri računanju javnega ključa.

4.2.3 Šifriranje sporočil

Denimo, da želi oseba B poslati sporočilo M osebi A. Oseba B spremeni M v število $m<n$, z uporabo predhodno dogovorjenega povratnega protokola poznanega kot protokol z dopolnjevanjem velikosti.

Oseba B ima sedaj *m* in *e*, ki ga je napovedala oseba A. Oseba B nato izračuna razdrobljeno besedilo *c* ki ustreza *m:*
- $c = m^e \bmod n$

To lahko hitro izvedemo s kvadriranjem. Oseba B nato pošlje *c* osebi A.

4.2.4 Dešifriranje sporočil

Oseba A prejme *c* osebe B in ima v lasti zasebni ključ *d.* Iz *c* lahko dobi *m* z naslednjo proceduro:

$$m = c^d \mod n$$

Z dobljenim *m* lahko dobi originalno sporočilo *M* po naslednjem postopku dešifriranja:

$$c^d \equiv (m^e)^d \equiv m^{ed} \pmod{n}$$

Imamo $ed \equiv 1 \pmod{p-1}$ in $ed \equiv 1 \pmod{q-1}$, Fermat-ov *majhen* teorem prinese:

$$m^{ed} \equiv m \pmod{p}$$

in

$$m^{ed} \equiv m \pmod{q}$$

Ker sta *p* in *q* različni praštevili, je skladen prinos uporabljen v teoremu Kitajskih ostankov:

$$m^{ed} \equiv m \pmod{pq}$$

Tako, da je:

$$c^d \equiv m \pmod{n}$$

4.2.5 Praktični primer

Predstavitev primera šifriranja in dešifriranja z uporabo RSA algoritma. Uporabljeni parametri so relativno majhni, vendar lahko z uporabo OpenSSL pridemo do realnih vrednosti za par ključev.

Parametri:

- p=61 - prvo praštevilo (mora biti tajno ali varno izbrisano)
- q=53 - drugo praštevilo (mora biti tajno ali varno izbrisano)
- $n=pq$=3233 - generator (ustvarjen javno)
- e = 17 - javni eksponent (ustvarjen javno)
- d = 2753 - tajni eksponent (mora ostati tajen)

Javni ključ je *(e,n)*. Tajni ključ je *d.* Funkcija šifriranja je:

encrypt(m) = m^e mod n = m^{17} mod 3233

Kjer je *m* čistopis. Funkcija dešifriranja je:

decrypt(c) = c^d mod n = c^{2753} mod 3233

Kjer je *c* razčlenjeno besedilo.

Za šifriranje čistpisa 123 izračunamo

Za dešifriranje razdrobljenega besedila 855 izračunamo.

Oba izračuna lahko izvedemo z uporabo koreni in deli algoritma za modularno eksponiranje.

4.2.6 Sheme za dodajanje

Ko RSA uporabljamo v praksi mora biti usklajen z obliko sheme za dodajanje bitov do zahtevane dolžine bloka, tako da so vrednosti *M* rezultat nezaščitenega šifriranega besedila. Če RSA uporabljamo brez sheme dodajanja lahko naletimo na naslednje probleme:

- Vrednosti m=0 ali m=1 vedno podajo šifrirano besedilo enako 0 ali 1 oziroma pričakovane glede na lastnosti enačbe
- Ko šifriramo z eksponentom nizke vrednosti (npr., e=3) in majhnimi vrednostmi *m*, rezultat od m^e bo nedvomno manj kot modul *n*. V tem primeru je šifrirno sporočilo lahko enostavno dešifrirano z n-tim korenom *e* šifriranega sporočila glede na modul.
- Ker je RSA šifriranje algoritem deterministočnega šifriranja – nima naključne komponente – napadalec lahko uspešno sproži napad na šifrirano besedilo proti šifrirnemu sistemu, tako, da si gradi slovar s šifriranjem čistopisov z javnim ključem in shranjuje pridobljena šifrirana besedila. Ko se na komunikacijskem kanalu pojavi podobno šifrirno besedilo, lahko napadalec uporabi ta slovar in prebere poslana sporočila.

V praksi lahko na prva dva problema naletimo, ko pošiljamo kratka ASCII sporočila, kjer je *m* povezan z enim ali več ASCII šifriranimi znaki. Sporočilo, ki predstavlja en ASCII *NULL* znak, katerega vrednost je 0, bo šifriran kot m=0, ki proizvede šifrirano besedilo 0, ne glede kakšen *e* ali *N* uporabljamo. Prav tako, enojen ASCII *SOH,*katerega numerična vrednost je 1, bo vedno proizvedel šifrirano besedilo 1. Za sisteme, ki navadno uporabljajo majhne vrednosti *e,* kot npr. 3, bodo vsi enojni ACSII znaki sporočila, šifrirani z uporabo tega sistema, nezaščiteni, odkar ima največji *m* vrednost 255 in 255^3 ki je manj kot bilo kateri razumljiv modul. Taka gola besedila lahko dobimo z kvadratnim korenom nad šifriranim besedilom.

Da se izognemo tem problemom, praktične RSA implementacije vstavljajo strukturni, organiziran dodatek v vrednost *m* preden ga šifrirajo. To dodajanje zagotavlja, da *m* ne pade v območje nezavarovanih čistopisov in da bo podano sporočilo, ko prejme dodatek, šifrirano v eno večje število različnih možnih šifriranih sporočil. Zadnja lastnost lahko poveča ceno napada s slovarjem preko razumnih zmožnosti napadalca.

Standardi, kot je PKCS (Public-Key Cryptography Standards) so bili previdno razviti za zavarovano pot sporočil pred RSA šifriranjem. Zaradi dodatka tem sistemov je čistopis *m* z nekim številom dodatnih bitov večji, kot je velikost ne-dodanega besedila *M* . RSA sistemi z dodatnimi biti morajo biti previdno načrtovani, da zaščitijo napade z namenom škodovanja, ki so imajo lažje delo zaradi predvidljive strukture sporočila. Zgodnje verzije PKCS standarda uporabljajo ad-hoc konstrukcijo, ki se je kasneje izkazala kot ranljiva pred praktično in prilagodljivo izbranim napadom na tajnopisi. Moderne konstrukcije uporabljajo varne tehnike kot so Optimal Asymetric Encryption Padding (OAEP) za zaščito sporočil, ki preprečujejo takšne napade. PKCS standard prav tako vključuje shemo procesiranja, s konstrukcijo, ki zagotavlja dodatno varnost za RSA podpise, npr. Probalistic Signature Scheme za RSA (RSA-PSS).

4.2.7 Podpisovanje sporočil

RSA lahko uporabljamo za podpisovanje sporočil. Predvidevamo, da oseba A želi osebi B poslati podpisano sporočilo. Oseba A ustvari vrednost sporočila z zgoščevalno funkcijo, poveča z močjo *d* mod *n,* tako kot to naredi pri dešifriranju sporočila in to pripne na sporočilo kot podpis. Ko oseba B prejme podpisano sporočilo, vzame podpis z močjo *e* mod *n,* tako, kot je to naredil, ko je šifriral sporočilo in primerja dobljeno vrednost z dejansko vrednostjo na sporočilu. Če se ti dve vrednosti ujemata, potem oseba B ve, da je avtor sporočila imel v lasti tajni ključ osebe A, in da to sporočilo od takrat ni bilo spremenjeno.

Vedeti moramo, da so varne sheme z dodajanjem kot je RSA-PSS bistvene za varnost podpisovanja sporočil, prav tako, kot so za šifriranje sporočil. Za šifriranje in podpisovanje naj se nebi uporabljal isti ključ.

4.2.8 Varnost

Varnost RSA šifrirnega sistema temelji na dveh matematičnih problemih: problem faktoriranja velikih števil in problem RSA. Polno dešifriranje RSA šifriranega besedila temelji na domnevi, da sta oba problema težka oz. ne obstaja algoritem za njihovo rešitev. Zagotavljanje varnosti proti delnemu dešifriranju zahteva dodajanje sheme za dodajanje bitov do zahtevane dolžine bloka.

RSA problem je definiran kot naloga jemanja modula n-tega korena sestavljenega *n* obnovljene vrednosti *m* tako da je $m^e=c$ mod n, kjer (e,n) je RSA javni ključ in *c* je RSA šifrirano sporočilo. Trenutno najbolj obetaven pristop k reševanju RSA problema je, da faktoriramo moldulus *n*. Z zmožnostjo obnavljanja primarnih faktorjev, napadalec lahko izračuna skrivni eksponent *d* iz javnega ključa *(e,n),* nato dešifrira *c* z uporabo standardne procedure. Za zagotavljanje tega, napadalec faktorira *n* v *p,* in izračuna *(p-1)(q-1),* ki dovoljuje determinanto *d* od *e*. Za faktoriranje velikih števil na klasičnem računalniku še ni bilo odkrite nobene polinomsko-časovne metode, vendar tudi ni bilo dokazano, da nobena ne obstaja.

Do leta 2006 je bilo največje število, ki je bilo faktorirano z uporabo splošno-namenskih metod, dolgo 640 bitov, z uporabo „state-of-the-art“ razporejenih metod. RSA ključi so tipično dolgi od 1024-2048 bitov. Nekateri strokovnjaki so mnenja, da bodo 1024-bitni ključi kmalu postali zlomljivi, nekaj pa jih je celo mnenja, da bodo celo ključi z dolžino 4096 v bližnji prihodnosti. Zaradi tega v glavnem domnevajo da je RSA varen, če je *n* zadostno velik. Če je *n* dolžine 256 bitov ali manj, potem je lahko razstavljen v nekaj urah na osebnem računalniku, z uporabo poceni programske opreme. Če je *n* dolžine 512 bitov ali manj je lahko razstavljen na nekaj 100 računalnikih od leta 1999. Leta 2003 sta Shamir in Tromer opisala teoretično napravo imenovano TWIRL ki je pod vprašanje spravila 1024 bitno dolžino ključa. Trenutno je priporočeno, da se uporablja ključe z dolžino 2096 bitov ali več.

Leta 1993 je Peter Shor predstavil algoritem ki prikazuje, kako kvantni računalnik v principu izvede razstavljanje v mnogočlenskem času, ki izkazuje RSA algritem kot izrabljen. Kljub temu ni pričakovano, da se bo kvantno računanje razvilo na takšen nivo še nekaj let.

<table>
<tr><th>Oznaka</th><th>Velikost</th><th>Datum</th><th>Rešil</th><th>Oprema</th><th>Čas</th></tr>
<tr><td>RSA-640</td><td>193 digits,640 bits</td><td>2.11.2005</td><td>F. Bahr, M. Boehm, J. Franke, T. Kleinjung</td><td>30 2.2GHz-Opteron-CPU</td><td>5 mesecev</td></tr>
<tr><td>RSA-200</td><td>200 digits</td><td>5.2005</td><td></td><td>80 2.2 GHz Opterons</td><td>8 mesecev</td></tr>
<tr><td>RSA-576</td><td>174 digits-576bit</td><td>2.12.2003</td><td>J. Franke and T. Kleinjung</td><td></td><td></td></tr>
<tr><td>RSA-160</td><td>160 digits-530bit</td><td>6.1.2003</td><td>Raziskovalci na BSI</td><td><table><tr><th>št.</th><th>Oprema</th></tr><tr><td>32</td><td>R12000</td></tr><tr><td>72</td><td>Alpha EV 67</td></tr></table></td><td>17 dni</td></tr>
<tr><td>RSA-155</td><td>155 digits-512 bit</td><td>22.11.1999</td><td>Skupina raziskovalcev</td><td><table><tr><th>št.</th><th>Oprema</th></tr><tr><td>160</td><td>175-400 MHz SGI</td></tr><tr><td>8</td><td>250 MHz SGI</td></tr><tr><td>120</td><td>300-450MHz P II</td></tr><tr><td>4</td><td>500 MHz Digital/Compaq</td></tr></table></td><td>3.7 mesecev</td></tr>
</table>

Tabela 3: Prikaz faktoriranja velikih praštevil pri RSA Challenge

4.3 PRAKTIČNA OCENA

4.3.1 Kreiranje ključev

Pridobivanje velikih praštevil *p* in *q* je izvedemo s testiranjem naključnih števil za pravilno velikost z verjetnostnim osnovnim preizkusom, ki hitro izloči vsa ne-praštevila. *p* in *q* ne smeta biti preblizu, da nebi bilo razčlenjevanje Fermat za *n* neuspešno. Če ima eden od obeh *p*-1 ali *q-1* samo majhna praštevila, je *n* razčlenjen hitro in iz tega sledi, da sta *p* in *q* neuporabna.

Nikakor pa ne smemo uporabiti metode iskanja praštevil, ki bi lahko kakorkoli podala kakršnokoli informacijo napadalcu. Iz tega sledi, da moramo uporabljati dober generator naključnih števil. Poudariti moramo, da imamo zahtevo za naključno in nepredvidljivo, kar pa ni isto merilo. Število je lahko izbrano z naključnim procesom, vendar, če je predvidljivo na kakršen koli način, bo uporabljena metoda povzročala nižanje stopnje varnosti. Kot primer lahko povemo, da je naključna tabela števil, ki jo je leta 1950 objavil Rand Corp, lahko zelo naključna,vendar je bila objavljena in to je lahko pomoč napadalcem. Če napadalec odkrije polovico števk *p* ali *q*, potem lahko hitro izračunajo drugo polovico.

Pomembno pri šifriranju je, da je tajni ključ *d* dovolj dolg. Wiener je leta 1990 dokazal, da če je *p* med *q* in q^2 , kar je dokaj običajna vrednost, in $b<n^{1/4p}/3$, potem lahko *d* izračunamo iz *n* in *e*. Čeprav so bile vrednosti *e* v preteklosti nizke, je nizek RSA eksponent nezaželen iz razlogov predhodno omenjene ranljivosti čistopisa, ki nima dodanih bitov. 65537 je vrednost, ki je navadno uporabljena za *e* in je glede na to dovolj velika, da se izogne napadalcem na nizke eksponente, prav tako pa ima zadostno težo da zagotovi uspešno naraščanje.

4.3.2 Hitrost

RSA je mnogo počasnejši kot DES in ostali simetrični šifrirni sistemi. V praksi oseba B v večini primerov zašifrira sporočilo s simetričnim algoritmom, zašifrira ga s primerljivo krajšim simetričnim ključem z RSA in oba, RSA zašifriran simetrični ključ in simetrično zašifrirano sporočilo, pošlje osebi A.

Tukaj se pojavi nekaj varnostnih vprašanj. Previdni moramo biti pri generiranju ključa in moramo uporabiti dovolj zmogljiv generator naključnih števil pri izdelavi simetričnega ključa, saj se lahko v nasprotnem primeru prisluškovalec izogne RSA na ta način, da ugane simetrični ključ.

4.3.3 Izmenjava ključev

Kot pri ostalem šifriranju je tudi pri RSA, iz varnostnih razlogov, pomemben način izmenjave ključev. Izmenjava ključev mora biti zaščitena pred prisluškovalci. Predvidevamo, da ima prisluškovalec možnost, da pošlje osebi B poljuben ključ, oseba B pa misli, da ima ključ osebe A. Iz tega sledi, da prisluškovalec lahko prestreza prenos med osebo A in osebo B. Prisluškovalec pošlje osebi B svoj javni ključ, zadrži kopijo sporočila, zašifrira sporočilo z javnim ključem osebe A in pošlje novo šifrirano besedilo osebi A. V principu niti oseba A niti oseba B ne bo zaznala prisotnosti prisluškovalca. Za zaščito pred tako vrsto napada pa najpogosteje uporabljamo digitalna potrdila in ostale komponente v infrastrukturi javnega ključa.

4.3.4 Sinhronizirani napadi

Kocher je leta 1995 predstavil iznajdljiv napad na RSA. Če prisluškovalec podrobno pozna strojno opremo osebe A in je sposoben meriti čase dešifriranja za različne poznane tajnopise, lahko hitro ugane šifrirni ključ *d*. Tak napad se lahko uporabi proti RSA sistemu podpisovanja. Leta 2003 sta Boenh in Brumley demonstrirala praktičen napad, ki je bil sposoben odkrivanja RSA faktorizacije preko omrežne povezave. Ta napad izkorišča uhajanje informacije pri optimizaciji problema kitajskih ostankov, ki ga uporablja večina izvedb RSA algoritma.

Ena izmed možnosti zaščite pred takimi napadi je, da zagotovimo konstanten čas izvajanja operacije šifriranja za vsako šifrirano besedilo. Tak pristop pa zmanjša učinkovitost. Namesto tega, večina izvedb RSA uporablja alternativno tehniko poznano kot šifrirno slepljenje. RSA slepljenje naredi uporabo multiplikativne lastnosti RSA. Namesto, da bi oseba A izračunala $c^d \bmod n$, najprej izbere skrivno naključno vrednost *r* in izračuna $(r^e c)^d \bmod n$. Rezultat tega izračuna je $rm \bmod n$ in tako je efekt *r* – ja umaknjen z njegovim inverznim množenjem. Nova vrednost *r* je izbrana za vsako šifrirano besedilo.

5 KRIPTOGRAFIJA Z ELIPTIČNO KRIVULJO

Kriptografija z eliptično krivuljo je pristop k kriptografiji javnega ključa, ki temelji na matematiki eliptične krivulje preko zaključenih polj. Uporabo te metode sta predlagala Neal Koblitz in Victor S. Miller leta 1985.

Eliptične krivulje se uporablja v večini algoritmov za faktorizacijo števil, ki imajo aplikacije v kriptografiji, kot primer faktorizacija z Lenstra – jevo krivuljo, vendar se ta uporaba ponavadi ne nanaša na šifriranje z eliptično krivuljo.

5.1 PREDSTAVITEV

Šifriranje z javnim ključem temelji na tvorbi matematične sestavljanke, ki jo je težko sestaviti brez znanja o sestavljanju delov v celoto. Sestavljavec skriva to znanje, ki predstavlja zasebni ključ in objavi delčke, javni ključ. Delčke sporočila lahko premešamo v takšni meri, da ga lahko sestavi samo sestavljavec. Nekateri algoritmi javnih ključev uporabljajo produkt dveh velikih praštevil, tako da so delčke v tem primeru posamezna praštevila. Zaradi napredka v faktoriranju števil morajo biti RSA javni ključi dolgi tisoče bitov, da lahko zagotavljajo visoko varnost. Drugi razred sestavljank vplete izračun enačbe $a^b = c$, kjer a in c poznamo. Taka enačba, ki vsebuje cela števila je z lahkoto rešljiva z uporabo logaritmov. V matematičnem sistemu poznano zaključene skupine kjer je rešitev mnogo bolj zapletena in to je poznamo kot problem diskretnega logaritma.

Eliptična krivulja je ravninska krivulja, definirana z enačbo oblike:

$$y^2 = x^3 + a\,x + b.$$

Točke na taki krivulji so prikazane kot množica točk in če gre enačba preko zaključenega polja je tudi množica točk zaključna. Rešitev problema diskretnega algoritma na skupinah eliptičnih krivulj je verjetno težavnejša, kot reševanje ciklični skupin, ki temeljijo na celih številih in zato so lahko ključi pri šifriranju z eliptično krivuljo krajši, kot pri sistemih, ki temeljijo na celih številih. Kot je značilno za vsa šifriranja z javnim ključem, do sedaj še bi bilo predstavljenih še nobenih dokazov o rešitvi matematične težavnosti. Ko je prenehal veljati RSA patent je za ECC (Elliptic curve cryptography) ostalo še nekaj patentov.

5.2 MATEMATIČNA PREDSTAVITEV

Eliptične krivulje, ki se uporabljajo v kriptografiji so definirane preko dveh tipov zaključenih polj. Polja preseženih karakteristik, $\mathbb{F}_p$, kjer je $p > 3$ je veliko praštevilo) in polja druge karakteristike $\mathbb{F}_{2^m}$. Kjer ni posebej določeno pokažemo obe kot F_q, kjer je $q=p$ ali $q=2^m$. V F_p so polja cela števila ($0 \le x < p$), ki sta sestavljena z uporabo modularne aritmetike. Primer F_{2m} je rahlo bolj kompliciran. Imamo več možnih prikazov elementov polja kot koščkov povezovanja in tistih zmanjšanih binarnih polinomov $f(x)$ stopnje m mora biti določeno.

Pari povezanih koordinat (x,y), kjer $x \in \mathbb{F}_q$ in $y \in \mathbb{F}_q$ oblika ploščate F_q x F_q. Izmed vseg smatramo tiste ki zadovoljijo enačbi eliptične krivulje in točki neskončnosti O (kljub skupni uporabi besede, krivulja v tem kontekstu sestavljena izključno iz individualnih, diskretnih in nepovezanih točk dokler je spodnja plast polja diskretna). V prvotnem primeru je bila definicija enačbe $E(F_p)$ sledeča: $y^2 = x^3 + ax + b$, kjer je a element F_p in b je element F_b sta konstanti kot $4a^3 + 27b^2 \neq 0$. Čeprav točka v neskončnosti 0 nima povezanih koordinat, je primeren za uporabo para koordinat, ki v nasprotnem primeru ne zadostijo definiciji enačbe npr. $O=(0,0)$ če $b \neq 0$ in $O=(0,1)$.

Glede na Hasse – jev teorem na eliptični krivulji je število točk na krivulji je v grobem iste velikosti, kot je velikost spodnjega nivoja polja:

$$|E(\mathbb{F}_q)| = q + 1 \pm 2\sqrt{q}$$

Za vsake dve točki na krivulji $P \in E(\mathbb{F}_q)$ in je možno $Q \in E(\mathbb{F}_q)$ najti tretjo točko $S = P + Q \in E(\mathbb{F}_q)$ tako, da določena relacija drži za vse točke na krivulji.

- $(P + Q) + R = P + (Q + R)$
- $P + O = O + P = P$
- obstaja $(-P)$ tako, da je $-P + P = P + (-P) = O$
- P+Q = Q+P

in tako je skupina vseh točk seštevek abelijske skupine.

Definirali smo O. Negativ točke $P = (x,y)$ je definiran kot $-P = (x, -y)$ za $P \in E(\mathbb{F}_p)$ in $-P = (x,x + y)$ za $P \in E(\mathbb{F}_{2^m})$.

Natančna pravila seštevanja so sledeča:

- če velja $Q = O$ potem $P + Q = P$
- če $Q = -P$ potem $P + Q = O$
- če $Q = P$ potem $P + Q = R$, kjer
  1. v primeru praštevil $x_R = \lambda^2 - 2x_P, y_R = \lambda(x_P - x_R) - y_P$, in
  2. v binarnem primeru $x_R = \lambda^2 + \lambda + a$, $y_R = x_P^2 + (\lambda + 1)x_R$ in $\lambda = x_P + \frac{y_P}{x_P}$
- če velja $Q \neq P$ potem $P + Q = R$, kjer
  3. v primeru praštevil $x_R = \lambda^2 - x_P - x_Q, y_R = \lambda(x_P - x_R) - y_P$, in $\lambda = \frac{y_Q - y_P}{x_Q - x_P}$
  4. v primeru binarnih števil $x_R = \lambda^2 + \lambda + x_P + x_Q + a, \lambda(x_P + x_R) + x_R + y_P$, in $\lambda = \frac{y_P + y_Q}{x_P + x_Q}$

Predhodno smo že opisali polje $\mathbb{F}_q$ in skupino točk eliptične krivulje $E(\mathbb{F}_q)$ vendar obstaja še ena matematična struktura, ki je pogosto uporabljena v kriptografiji – ciklična podskupina . Za vsako točko $E(\mathbb{F}_q)$ G je niz ciklična skupina. $(O, G, G + G, G + G + G, G + G + G + G, \ldots)$ Je primerna za uporabo naslednjga sistema simbolov $0G = O$, $1G = G$, $2G = G + G$, $3G = G + G + G$, itd. Izračun kG, kjer je k celo število in G točka, se imenuje skalarna multiplikacija.

5.3 Diagram šifriranja

Težavnost iskanja *k* podanih točk *kG* in *G* imenujemo problem diskretnega logaritma eliptične krivulje (ECDLP - Elliptic Curve Discrete Logarithm Problem) . Predhodno opisana seštevalna zaključena skupina je podobna kot multiplikativna skupina z močjo $(g^0, g, g^2, g^3, g^4, \ldots)$ celih števil generatorja *g* in primarnega števila *p*

Domnevna težavnost večine problemov povezanih z diskretnim logaritmom v podskupini $E(\mathbb{F}_q)$ dovoljuje kriptografsko uporabo eliptičnih krivulj. Večina diagramov eliptične krivulje je povezanih v diagram diskretnega logaritma, ki so bili v originalu formulirani za običajno modularno aritmetiko:

- sistem izmenjave ključev *Diffie-Hellman eliptične krivulje* temelji na sistemu Difie-Hellmana
- sistem dogovora ključev *Algoritma digitalnega podpisa eliptične krivulje* temelji na *Algoritmu digitalnega podpisa.*
- Dogovor ključev *ECMQV* temelji na sistemu *MQV*

Vseh DLP sistemov pa ne smemo šteti v domeno eliptične krivulje. Kot primer naj povemo, da dobro poznan sistem šifriranja ElGamal ni bil nikoli uradno standardiziran in ne sme biti neposredno uporabljen preko eliptične krivulje (standarden sistem šifriranja za ECC se imenuje *enoten sistem šifriranja z eliptično krivuljo)*. Čeprav je protokol namenjen pretvorbi poljubnega sporočila omejene dolžine v celo število modula *p*, ni tako preprosto pretvoriti binarnega besedila v točko krivulje (za vsak x_0 obstaja *y* tako da $(x, y) \in E(\mathbb{F}_q)$).

Predvideva se, da naj bi v prihodnosti kriptografija, ki temelji na ECDLP zamenjala kriptografijo, ki temelji na faktorizaciji celih števil, kot je npr. RSA in kriptografijo z omejenimi polji kot je npr. DSA. Na RSA konferenci leta 2005 je nacionalni inštitut za varnost (NSA) predstavil Garnituro B (Suite B), ki ekskluzivno uporablja ECC za kreiranje digitalnih podpisov in izmenjavo ključev. Garnitura (suite) je namenjena zaščiti tajnih in javnih varnostnih sistemov in informacij.

Eden izmed povodov za uporabo kriptografskih aplikacij eliptičnih krivulj je bi-linearni operator (temelji na paritvi Weil ali paritvi Tate) ki dovoljuje, npr. izdelavo zmogljive kriptografije ki temelji na ID.

5.4 POMEMBNO PRI IZVAJANJU

Čeprav so bile podrobnosti vsakega posebnega sistema eliptične krivulje opisane v zgornjem odstavku, so za nekatere izvedbe potrebna dodatna pojasnila.

5.4.1 Parametri področja

Pri uporabi KEK se morajo vsi sodelujoči strinjati o vseh elementih, ki določajo eliptično krivuljo. To so področni parametri sheme. Polje je definirano s *p* v primeru praštevil in s parom *m* in *f* v primeru binarnih števil. Eliptična krivulja je določena s konstantama *a* in *b* uporabljenima v definicijski enačbi. Zaključena podskupina je definirana z generatorjem *G*. Za kriptografske aplikacije mora biti razred *G* praštevilo in *n* najmanjše ne-negativno število tako da je *nG=O*.

Če je *n* velikosti $E(\mathbb{F}_q)$ podskupine, potem iz Lagrangejevega teorema sledi, da je število $h = \frac{|E|}{n}$ celo število. V kriptografskih aplikacijah mora biti število *h*, imenovano tudi sofaktor, majhne vrednosti ($h \leq 4$) in po možnosti, *h*=1. Naj povzamemo: V primeru praštevil so parametri področja (*p*,*a*,*b*,*G*,*n*,*h*), v primeru binarnih števil pa (*m*,*f*,*a*,*b*,*G*,*n*,*h*).

Parametri področja morajo biti potrjeni pred uporabo, razen, če niso ustvarjeni s strani stranke, ki je vredna zaupanja. Področni parametri niso ustvarjeni od vsakega sodelujočega, ker to zaplete štetje točk na krivulji, kar je časovno potratno in težavno za izvajanje. Kot rezultat so številne ustanove za standarde objavile področne parametre eliptične krivulje za različne velikosti polj.

- NIST, Recomended Elliptic Curves for Government Use
- SECG, SEC 2: Recommended Elliptic Curve Domain Parameters

Če hoče nekdo zgraditi svoje področne parametre mora izbrati osnovno polje in nato porabiti eno izmed sledečih strategij, da najde krivuljo, ki ustreza številu točk, oz. je najbližje praštevilu, z uporabo ene od sledečih metod:

- izberemo naključno krivuljo in izberemo splošen algoritem štetja točk npr. Schoof-ov algoritem ali Schoof-Elkins'Atkinov algoritem.
- izberemo naključno krivuljo iz družine, ki dovoljuje lahek izračun števila točk (npr. Koblitz-eve krivulje)
- izberemo število točk in s tem številom kreira krivuljo z uporabo tehnike kompleksne multiplikacije

Nekaj razredov krivulj imajo šibke točke in se jim zato izogibamo:

- krivulje preko $\mathbb{F}_{2^m}$ z številom *m*, ki ni praštevil,o so izpostavljene Weil-ovim nepričakovanim napadom.
- krivulje katerih *n* deli q^B-1 za majhen B *(B<20)* so izpostavljene napadom MOV, ki uporabljajo običajen DLP v polju nizke stopnje obsežnosti za rešitev ECDLP.
- krivulje, ki so izpostavljene napadom, ki prirejajo točke na krivulji

5.5 Velikost ključev

Iz podatka, da vsi najhitrejši algoritmi, ki dovoljujejo reševanje ECDLP potrebujejo $O(\sqrt{n})$ korakov sledi, da mora biti velikost osnovnega polja približno dvakratnik varnostnega parametra. Kot primer lahko povemo, da za 128-bitno varnost potrebujemo krivuljo velikosti preko $\mathbb{F}_q$ kjer je $q \approx 2^{256}$. To lahko primerjamo z kriptografijo s končnimi polji, kot je npr. DSA, ki zahteva 3072-bitni javni ključ in 256-bitni privatni ključ in s kriptografija z algoritmom faktorizacije celih števil, kot je npr. RSA, ki zahteva 3072-bitni javni in privatni ključ. Najtežja ECC poznana shema, ki je bila do sedaj razbita ima 109-bitni ključ, kar je približno 55-bitov varnosti. Razbita je bila v začetku leta 2003 z uporabo preko 10.000 osebnih računalnikov, ki so nenehno delovali 540 dni.

5.6 Projekcijske koordinate

Podrobna raziskava dodatnih pravil prikazuje, da zaporedje dodajanja dveh točk enega zahtevka ne potrebuje samo različnih dodatkov in multiplikacij $\mathbb{F}_q$, temveč tudi inverzno operacijo. Inverzija, ki za podani $x \in \mathbb{F}_q$ najde $y \in \mathbb{F}_q$ tako, da je *xy*=1, je razred ali dva počasnejša kot multiplikacija. Na srečo so lahko točke na krivulji prikazane v različnih koordinatnih sistemih ki ne zahtevajo inverzne operacije za dodajanje dveh točk.

Na voljo imamo več takšnih sistemov:

- V projekcijskem sistemu je vsaka točka predstavljena s tremi koordinatami *(X,Y,Z)*, z uporabo naslednje relacije: $x = \frac{X}{Z}$ in $y = \frac{y}{Z}$
- V spremenjenem Jacobian-ovem sistemu so uporabljene iste relacije, vendar so koordinate shranjene in uporabljene za izračun (X,Y,Z,aZ^4). V Chudnovsky – Jacobianovem sistemu je uporabljenih petih koordinat (X,Y,Z,Z^2,Z^3). Vedeti moramo, da je pri poimenovanju prišlo do različnih dogovorov. Za primer: IEEE P1363-200 standard uporablja „koordinate projeciranja“ in se nanaša na to, kar navadno pravimo Jacobianove koordinate. Če uporabljamo različne koordinate, pridobimo na hitrosti izvajanja.

5.7 Hitrost spreminjanja – NIST Krivulje

Spreminjanje modula *p*, ki ga uporabljamo za dodajanje in multipliciranje se lahko izvaja hitreje, če je praštevilo *p* pseudo-Mersen-ovo praštevilo tako da je $p \approx 2^d$, za primer, $p=2^{521}-1$ ali $p = 2^{256} – 2^{32} – 2^9 – 2^8 – 2^7 – 2^6 – 2^4 – 1$. V primerjavi z Barrett-ovoim zmanjševanjem pride lahko do stopnje pospeševanja. NIST priporoča krivulje preko $\mathbb{F}_p$ z nepravim-Mersenne *p* . NIST krivulje imajo tudi to prednost, da uporabljajo *a= -3*, ki izboljša dodajanje v Jacobian-ovih koordinatah.

5.8 Napadi po stranskem kanalu

Za razliko od DLP sistemov (kjer je možno uporabljati isto proceduro za kvadriranje in množenje) se EC seštevanje bistveno razlikuje pri podvojevanju *(P=Q)* in pri osnovnem $P \neq Q$ seštevanju . Posledično je pomembno preprečiti napade po stranskem kanalu (npr. napadi s pomočjo časovne in močnostne analize) z uporabo, npr. metode vzorca okna. Povdariti moramo, da to ne poveča časa računanja.

6 ALGORITEM DIGITALNEGA PODPISA

Algoritem digitalnega podpisa (DSA, Digital Signature Algorithm) je standard zvezne vlade združenih držav za digitalno podpisovanje.

Kaj mora zagotoviti postopek, da digitalnemu podpisu verjamemo kot običajnemu?

- avtentičnost (verjamemo, da je podpisnik res tisti, za kogar se proglaša);
- podpisa se ne da ponarediti;
- podpisa se ne da kopirati;
- podpisanega dokumenta se ne da spremeniti;
- podpisa se ne da zanikati (podpisnik ne more reči, da ni on podpisal dokumenta).

Za pojem digitalni podpis (digital signature) je več razlag. Ena možnost je podpisovanje z uporabo simetričnega algoritma s pomočjo notarja. Danes se običajno uporablja asimetrične algoritme - datoteko podpisnik zašifrira s svojim zasebnim ključem. Dešifrira jo lahko vsak, ki pozna njegov javni ključ, in to se šteje kot preverjanje podpisa. Ker pa bi bilo šifriranje dolgih datotek z asimetričnim algoritmom prepočasno, pri digitalnem podpisovanju najprej naredimo povzetek (hash) datoteke z zgoščevalno funkcijo in samo tega zašifriramo z zasebnim ključem. Prednost tega je tudi to, da tak podpis lahko dodamo ne-zašifrirani datoteki.

Naj opozorimo na razliko med pojmoma *elektronski* in *digitalni* podpis: Elektronski podpis pomeni kakršnekoli oznake, narejene z elektronskimi mediji, z namenom, da označijo nek dokument ali datoteko.

Digitalni podpis pa je elektronski podpis, narejen z uporabo kriptografije tako, kot smo to ravnokar opisali.

Zaenkrat je najbolj pogosto v uporabi kombinacija SHA-1 z RSA, kot je to določeno v standardu PKCS#1. Ko je NIST objavil svoj Digital Signature Standard (DSS), ki uporablja poseben asimetrični algoritem, je dobil precej kritik od proizvajalcev, ki so večinoma že implementirali digitalni podpis z uporabo RSA.

6.1 POTEK ALGORITMA

6.1.1 Opis

Pošiljatelj izračuna povzetek dokumenta z zgostitvenim algoritmom. Podpis naredi tako, da ta povzetek zašifrira s svojim zasebnim ključem. Odpošlje dokument, ki mu priloži podpis. Naslovnik z javnim ključem pošiljatelja dešifrira podpis, dobi povzetek. Ponovno izračuna povzetek pisma z istim zgostitvenim algoritmom kot pošiljatelj. Če se ujemata, pomeni, da je dobil tak dokument, kot ga je pošiljatelj podpisal. Poleg tega imamo na izbiro: dokument sam lahko zašifriramo (z naslovnikovim javnim ključem) ali pa tudi ne.

Kadar gre za dokumente, ki se ne smejo večkrat uporabiti (npr. čeke - naslovnik bi jih lahko večkrat vnovčil), mora biti del podpisa tudi **čas nastanka (timestamp)**. To omogoča kontrolo kopiranja in lažje razčiščevanje ob morebitnih sporih. Tak podpis lahko nastane samo na računalniku, ki je časovno sinhroniziran z neodvisnim časovnim strežnikom.

6.1.2 Generiranje ključev

- Izbira 160-pitnega praštevila q
- Izbira L-bitnega praštevila p tako da je $p=qz+1$ za nekatera cela števila in da je $512 \leq L \leq 1024$ in L je deljiv z 64.
  - Sprememba po FIPS-182-2 (Federal Information Processing Standards) opisuje, da si L lasti samo vrednost 1024
  - Sprememba po FIPS-182-3 uporablja HA-224, SHA-256, SHA-384, and SHA-512 kot hash funkcijo, q velikosti 224,256,384 in 512 bitov z L jem enakim 2048, 3072, 7680 in 15360.
- Izbira h, kjer $1 < h < p - 1$ tako da $g = h^z \bmod p > 1$.
- Izbere x s pomočjo naključne metode, kjer $0 < x < q$.
- Izračun $y = g^x \bmod p$.
- Javni ključ je (p, q, g, y). privatni ključ je.

p,g in q so v skupni rabi med uporabniki in sistemi

Podpisovanje
- Izdelava naključne vrednosti k kjer je $0 < k < q$
- Izračun $r = (g^k \bmod p) \bmod q$
- Izračun $s = (k^{-1}(\text{SHA-1}(m) + x*r)) \bmod q$, kjer SHA-1($m$) je SHA-1 hash funkcija uporabljena za sporočilo m
- ponoven izračun podpisa v redkih primerih, ko je r=0 ali s=0
- Podpis je (r,s)

Overovljenje
- Izločanje podpisa, če ni zadoščeno pogojema 0<r<q ali 0<s<q
- Izračun $w = (s)^{-1} \bmod q$
- Izračun $u1 = (\text{SHA-1}(m)*w) \bmod q$
- Izračun $u2 = (r*w) \bmod q$
- Izračun $v = ((g^{u1}*y^{u2}) \bmod p) \bmod q$
- Podpis je veljaven, če velja pogoj $v = r$

Pravilnost algoritma
Shema podpisovanja je pravilna v smislu, da overitelj sprejme avtentičen podpis. To lahko opišemo na naslednji način:

Iz $g = h^z \bmod p$ sledi $g^q \equiv h^{qz} \equiv h^{p-1} \equiv 1 \pmod p$ po Fermat-ovem *majhnem* teoremu. V primeru, da je $g>1$ in q je praštevilo sledi, da je g reda q.

Podpisnik izračuna:

$$s = k^{-1}(\text{SHA-1}(m) + xr) \mod q.$$

Tako, da je:

$$\begin{aligned} k &\equiv \text{SHA-1}(m)s^{-1} + xrs^{-1} \\ &\equiv \text{SHA-1}(m)w + xrw \pmod q. \end{aligned}$$

Če je *g* razreda *q,* potem imamo:

$$\begin{aligned} g^k &\equiv g^{\mathrm{SHA-1}(m)w} g^{xrw} \\ &\equiv g^{\mathrm{SHA-1}(m)w} y^{rw} \\ &\equiv g^{u1} y^{u2} \pmod{p}. \end{aligned}$$

Na koncu dobimo popravek DSA iz enačbe:

$$r = (g^k \mod p) \mod q = (g^{u1} y^{u2} \mod p) \mod q = v.$$

6.1.3 Uporaba istega para ključev

Uporabnik lahko uporablja isti par ključev za podpisovanje in za šifriranje podatkov. Vendar pa to ni dobro iz naslednjih razlogov:

- Zasebni ključ, ki se ga uporablja za podpisovanje, sme poznati samo lastnik, saj bi sicer lahko tajil, da je podpisal nek dokument. Torej v nobenem primeru ne sme nihče razen lastnika imeti varnostnih kopij tega ključa. Nasprotno pa je za zasebni ključ, ki ga uporablja za šifriranje, včasih nujno, da ga pozna še kdo drug in da imamo varnostne kopije, saj bi sicer lahko izgubili pomembne podatke.
- Javni ključ, ki pripada zasebnemu ključu za podpisovanje, se mora hraniti tudi po tem, ko ni več veljaven, da lahko preverimo podpise na starih dokumentih. Za javni ključ, ki ga uporabljajo njegovi dopisovalci za šifriranje podatkov, pa to ni potrebno. Ko preneha veljati ali se izgubi, tvori in objavi novega.
- Ni nujno, da imamo za oba para ključev isto obdobje veljavnosti - za par za podpisovanje je to obdobje običajno daljše.
- Ni nujno , da za oba para uporabimo isti algoritem (za podpis lahko uporabimo algoritem, ki ni primeren za šifriranje).
- Ameriške izvozne omejitve ne veljajo za programsko opremo za podpisovanje. Izvozne omejitve za opremo za šifriranje so bile januarja 2000 zelo omiljene, vendar ne povsem odpravljene.

7 ELGAMAL ŠIFRIRANJE

Šifriranje po ElGamalu je algoritem asimetričnih ključev za šifriranje z javnim ključem, ki temelji na skladnosti ključev po Diffie-Hellman-u. Algoritem je predstavil Taher Elgamal leta 1984. Algoritem Elgamal upoerablja GPG - GNU Privacy Guard programska oprema, različne verzije PGP in ostali kriptosistemi. Algoritem digitalnega podpisa je različica sheme podpisaovanja Elgamala, ki se jo zamenjuje za ElGamal.ov algoritem.

ElGamal-a lahko definiramo preko katerekoli zaključene skupine *G*. Njegova varnost je odvisna od teže posameznega problema v *G* v odvisnosti izračuna diskretnih logaritmov.

7.1 ALGORITEM

ElGamal je sestavljen iz treh komponent: generatorja ključev, algoritma šifriranja in algoritma dešifriranja.

7.1.1 Generator ključev

Deluje na naslednji način:

- oseba A generira zadovoljiv opis ciklične skupine *G* reda *q* z generatorjem *g*
- oseba A izbere naključen *x* iz $\{0, \ldots, q-1\}$
- oseba A izračuna $h = g^x$.
- oseba A objavi *h*, skupaj z opisom *G*,*q*,*g*, kot njen javni ključ. Oseba A obdrži *x* kot njen skrivni ključ.

7.1.2 Algoritem šifriranja

Delovanje:

- za šifriranje sporočila *m* namenjenega osebi A pod njenim javnim ključem *(G,q,g,h),*
- oseba B preoblikuje *m* v element *G*-ja.
- oseba B izbere naključen $\{0, \ldots, q-1\}$ *y* iz , nato izračuna $c_1 = g^y$ in $c_2 = m \cdot h^y$

7.1.3 Algoritem dešifriranja

Delovanje:

- za dešifriranje šifriranega besedilat (c_1,c_2) s tajnim ključem osebe A *x*
- oseba A izračuna $c_2(c_1^x)^{-1}$
- Dešifrirni algoritem proizvede pričakovano sporočilo ko je:

$$c_2(c_1^x)^{-1} = \frac{m \cdot h^y}{g^{xy}} = \frac{m \cdot g^{xy}}{g^{xy}} = m$$

Če je je število predvidenih sporočil večje kot *G*, potem lahko sporočilo razbijemo na različne dele in vsak del je lahko šifriran posebej. Ponavadi je krajši ključ šifre simetričnega ključa najprej šifrirano z ElGamal, in predvideno daljši del sporočila je šifriran s simetričnim ključem – tak postopek imenujemo *hibridno šifriranje.*

7.2 VARNOST

ElGamal je preprost primer pomenske varnosti algoritma asimetričnega šifriranja pod razumljivimi predpostavkami. Verjetno je, če vzamemo, da je čistopis lahko zašifriran v različna tajnopise, s posledicami ta osnovno ElGamal šifriranje proizvede 2:1 povečanje v velikosti, če primerjamo čistopis in tajnopis.

ElGamal-ova varnost temelji na težavnosti reševanja problema diskretnega logaritma v G. Iz prej omenjenega je razvidno, da če lahko rešimo problem diskretnega logaritma, potem lahko pride do vdora. Kljub temu, pa se varnost ElGamal-ovega sistema pravzaprav nanaša na t.i. odločitveno Diffie-Hellmanovo predpostavko (DDH – Decisional Diffie-Hellman). Ta predpostavka je v večini primerov močnejša kot predpostavka diskretnega logaritma, vendar se še vedno smatra za pravilno za veliko razredov skupine.

7.3 IZVEDBA SKUPINE G

Kot je bilo predhodno opisano lahko ElGamal-ov algoritem lahko definiramo preko katerekoli končne skupine G in je varen, če je določena izračunana predpostavka (DHH predpostavka) o skupini pravilna. Na nesrečo, nekompromisna uporaba $G=Z_p$ za praštevilo p ni varno, ker je DHH predpostavka na tej skupini nepravilna. Računanje diskretnih logaritmov v Z_p se smatra za težavno, vendar to za varnost ElGamal-a ni dovolj.

Dva najbolj popularna tipa skupin uporabljenih v ElGamalu sta podskupini Z_p in skupine definirane preko določenih eliptičnih krivulj. Opis najbolj popularne poti za iskanje najprimernejše poti Z_p , ki se smatra za varno:

- Izbira naključnega velikega praštevila p tako da je p-1=kq za nekatera majhna praštevila k in velikega praštevila q. To lahko izvedemo z k=2, s tem da najprej izberemo veliko praštevilo q in preverjamo, če je p= 2q+1 praštevilo.
- Izberemo naključen $g \in Z_p$ element tako, da velja in $g^q = 1 \bmod p$, tako da je g razreda q
- Skupina G je podskupina Z_p izvedba s setom k-tega preostanka mod p

Ko šifriramo, moramo biti pozorni na pravilno kodiranje sporočila m kot element G in ni samo poljuben element Z_p.

7.4 ZMOGLJIVOST

Šifriranje z ElGamal-om zahteva dve eksponenta. Ta dva eksponenta sta neodvisna od sporočila in je lahko izračunan vnaprej od potrebnega časa. Dešifriranje zahteva samo en eksponent in eno deljenje, ki se izvede bistveno hitreje. Za razliko od RSA in Rabin sistemov, ElGamal dešifriranje ne moremo pospešiti z uporabo teorema kitajskih ostankov.

7.5 POVZETEK

ElGamal je zelo prilagodljiv. Pri šifriranju (c_1,c_2) poljubnega sporočila m, lahko nekdo izdela šifriranje $(c_1,2*c_2)$ za sporočilo 2m in tako postane sistem nezaščiten. Po drugi strani pa imamo Chramer-Soup sistem, ki temelji na ElGamal-u in je varen.

8 ZAKLJUČEK

Kateri sistem naj izberemo? Naj se odločimo za simetričnega ali asimetričnega? Vsak ima svoje prednosti in slabosti. Pomembno je, da temeljito preučimo potrebe, želje in možnosti izpeljave posameznega sistema.

Največkrat pa v praksi zasledimo hibridni pristop. Največkrat se zgodi, da asimetrične postopke uporabljamo za šifriranje krajših sporočil ter z njimi izmenjujemo ključe za simetrično šifriranje, ki jih kasneje uporabljamo za šifriranje sporočil. Na ta način se izognemo slabostim posameznih sistemov.

Nenazadnje pa je varnost podatkov v veliki meri odvisna od nas samih, našega pristopa do varovanja le teh in vsakodnevnega spremljanja sprememb varnostnih mehanizmov, ki se morajo dandanes zaradi novejših znanj in novih tehnologij nenehno izpopolnjevati.