HRVATSKA AKADEMSKA I ISTRAŽIVAČKA MREŽA
CROATIAN ACADEMIC AND RESEARCH NETWORK

Napadi na RSA

CCERT-PUBDOC-2003-04-19

CARNet CERT u suradnji s LS&S

Sigurnosni problemi u računalnim programima i operativnim sustavima područje je na kojem CARNet CERT kontinuirano radi.

Rezultat toga rada ovaj je dokument koji je nastao suradnjom CARNet CERT-a i LS&S-a, a za koji se nadamo se da će Vam koristiti u poboljšanju sigurnosti Vašeg sustava.

CARNet CERT, www.cert.hr - nacionalno središte za **sigurnost računalnih mreža** i sustava.

LS&S, www.lss.hr - laboratorij za sustave i signale pri Zavodu za elektroničke sustave i obradbu informacija Fakulteta elektrotehnike i računarstva Sveučilišta u Zagrebu.

Sadržaj

1. Uvod

Postoje dvije temeljne vrste kriptografskih algoritama:

- algoritmi temeljeni na tajnim ključevima (engl. *secret-key algorithms*, *symmetric algorithms*),
- algoritmi temeljeni na javnim ključevima (engl. *public-key algorithms*, *asymmetric algorithms*).

Simetrični algoritmi imaju svojstvo da se kod njih ključ za dešifriranje može na jedinstveni (i jednostavan) način odrediti iz ključa za šifriranje i obratno. Kod većine takvih algoritama ključevi za šifriranje i dešifriranje su identični, tako da se pošiljatelj i primatelj moraju na neki način usuglasiti oko njihove vrijednosti. Sigurnost rada temelji se na tajnosti ključa; ukoliko je ključ otkriven, svaka daljnja komunikacija između sudionika je kompromitirana. Najpoznatiji i najrašireniji simetričan algoritam je DES.

Asimetrični protokoli koriste se algoritmima temeljenim na javnim ključevima. Takovi algoritmi koriste dva ključa; javni ključ, koji se obznanjuje javno, te privatni ključ koji mora biti tajan. Svako može uzeti bilo čiji tajni javni ključ (iz javne baze ključeva), te šifrirati poruku, dok tako šifriranu poruku može dešifrirati samo onaj kome je ta poruka namijenjena, odnosno onaj tko zna odgovarajući privatni ključ.

Zajedničko načelo tih algoritama, odnosno zahtjeve koje moraju zadovoljiti funkcija šifriranja `E`, te funkcija dešifriranja `D`, predstavljaju tri slijedeća zahtjeva:

1. `D(E(P))=P`
2. izrazito je teško (praktički nemoguće) zaključiti `D` iz poznavanja `E`
3. `E` se ne može razbiti ni uz mogućnost šifriranja proizvoljnih podataka (engl. *chosen plaintext attack*)

Ukoliko su ispunjeni svi gore navedeni uvjeti, nema razloga da se ključ za šifriranje `E` ne obznani javno. Ukoliko neka osoba želi koristiti takav algoritam, odnosno primati tajne poruke, treba samo generirati dvije funkcije `E` i `D`, te zatim objaviti `E`. U tom trenutku svaka osoba koja želi poslati tajnu poruku prvoj osobi za šifriranje koristi funkciju `E` (javni ključ). Tako šifriranu poruku može dešifrirati samo osoba koja poznaje odgovarajući, privatni ključ `D`.

Najpoznatiji takav algoritam jest RSA.

2. RSA algoritam

RSA algoritam razvili su Rivest, Shamir i Adelman 1977. godine. Sigurnost algoritma temelji se na složenosti izračunavanja vrlo velikih prim brojeva. Do ovog trenutka algoritam se pokazao prilično sigurnim, iako mnoge stvari koje podrazumijevaju prilikom uporabe algoritma nisu dokazane.

Nije dokazano da je korištenje modulo aritmetika najbolji način za izračunavanje velikih prim brojeva, a također i za napade na RSA. Teoretski je moguće da postoje još neotkriveni načini na koje se RSA može napasti. Također nije dokazano da je izračunavanje prim brojeva toliko složeno kako se ovaj tren čini. Postoji mogućnost da će napredak u teoriji brojeva dovesti do otkrića algoritma čija složenost je polinomska.

Niti jedan od tih potencijalnih nedostataka do sad nije otkriven, no postoje druge mogućnosti i načini koji vremenom utječu na sigurnost RSA algoritma. Napreci u tehnici izračunavanja, pad cijena sklopovlja i porast računalne moći omogućavaju razne vrste napada, no RSA se zasad uspješno odupire tim napadima, pošto isti ti napreci paralelno omogućavaju i povećanje duljine RSA ključeva.

Algoritam se sastoji od sljedećih koraka:

1. odabrati dva velika prim broja; `p` i `q`, (obično veći od 10^{100}),
2. izračunati `n=p*q` i `z=(p-1)*(q-1)`,
3. izabrati broj `d` takav da su `z` i `d` relativno prosti,
4. pronaći `e` takav da je `(e*d) mod z =1`,
5. privatni ključ čini par `(n,d)`, dok javni ključ čini par `(e,d)`,
6. `p` i `q` ne smiju biti nikad otkriveni, poželjno ih je uništiti (neki kriptografski sustavi ipak čuvaju te brojeve da bi ubrzali operacije šifriranja/dešifriranja).

Šifriranje poruke provodi se dijeljenjem poruke u blokove duljine M takve da je $2^M<n$ (u nastavku dokumenta podrazumijeva se da n nije decimalni broj, već binarni, tako da je u tom slučaju ekvivalentni izraz $M<n$) te korištenjem modulo aritmetike:

$$C=M^e \bmod n$$

Dešifriranje predstavlja inverznu operaciju, tako da se izvorna poruka dobiva na sljedeći način:

$$M=C^d \bmod n$$

3. Napadi

3.1. Napad primjenom sile

Sama načela kriptografije temeljene na javnim ključevima omogućavaju napadaču pristup javnom ključu. Dakle, uz par vrijednosti (e,n) koje predstavljaju javni ključ, napadač želi doći do privatnog ključa. Drugim riječima, napadač želi izračunati d. Općenito, ovakva vrst napada naziva se napad primjenom sile (engl. *brute force attack*). Dosad najefikasniji napad protiv RSA jest faktorizacija broja n. Također postoji mogućnost napada koja se temelji na pokušaju računanja $(p-1)*(q-1)$, no vremenska složenost tog napada nije ništa jednostavnija nego prethodno opisani napad. Moguće je i direktno traženje broja d, no pokazuje se da je taj postupak složeniji nego ranije opisane mogućnosti.
Postoji nekoliko algoritama za faktorizaciju:

- dijeljenje – predstavlja najstariju i najmanje efikasnu metodu, a podrazumijeva iskušavanje svih prim brojeva koji su manji ili jednaki $n^{1/2}$ (eksponencijalna složenost),
- kvadratni Sieve algoritam – najbrži algoritam za brojeve manje od 110 znamenki,
- više polinomski kvadratni Sieve algoritam – brža inačica prethodnog algoritma,
- Sieve algoritam sa općim brojčanim poljima (engl. GNFS – *General Number Field Sieve*),
- Sieve algoritam sa specifičnim brojčanim poljima (engl. SNFS – *Specific Number Field Sieve*).

Spomenuti algoritmi predstavljaju najbolje mogućnosti za napad na RSA algoritam. Sieve algoritmi imaju tzv. super-polinomsku složenost (sub-eksponencijalnu), a složenost Sieve algoritma s brojčanim poljima se asimptotski približava polinomskom ponašanju.
Faktorizacija velikih brojeva je matematički složena i zahtijeva mnogo vremena. Napreci u teoriji brojeva i povećanje procesorske moći olakšavaju postupak faktorizacije, čime se olakšavaju napadi na RSA algoritam. Tablica 1 daje okvirnu složenost razbijanja RSA ključeva izraženu u MIPS-godinama (broj godina koji je potreban da računalo koje ima brzinu računanja od 1 milijuna instrukcija u sekundi obavi traženu faktorizaciju).

DULJINA KLJUČA	MIPS-godine potrebne za faktorizaciju	
	GNFS	SNFS
512	~30 000	<200
768	~200 000 000	~100 000
1024	~300 000 000 000	~30 000 000
2048	~300 000 000 000 000 000 000	~400 000 000 000 000

Tablica 1: Vrijeme i procesorska moć potrebna za napad na RSA ključeve

3.1.1. Obrana

Definitivno je da u današnje vrijeme ključevi duljine 512 bita, 768 bita, pa i dulji mogu biti probijeni primjenom sile ukoliko se upotrijebi odgovarajuća procesorska moć. To naravno nije razlog za paniku, pošto financijska sredstva potrebna za to u većini slučajeva nadilaze korist od probijanja tih ključeva. Ipak, logički je da RSA ključevi koje koriste organizacije ili još više državne ustanove budu dulji nego ključevi koje koriste obični korisnici.

Najjednostavniji način zaštite jest uporaba ključeva što veće duljine. Prilikom toga ne treba pretjerivati, iako neke implementacije omogućavaju duljine ključeve 8192 bita,16 384 bita pa i dulje. Za obične korisnike duljina ključa od 2048 bita u ovom trenutku je dovoljna.

3.2. Napad iskorištavanjem malog eksponenta

Pokazuje se da, ukoliko je e relativno malen, to ne utječe na sigurnost samog algoritma. Ukoliko je eksponent za šifriranje malen (npr. 3, 17, 65537), operacija šifriranja je mnogo brža. Jedini nedostatak korištenja malenog eksponenta dolazi do izražaja prilikom šifriranja kratkih poruka. Ukoliko se za e odabere 3, i ukoliko vrijedi da je $M<n^{1/3}$ (poruka je kraća od trećeg korijena od n), poruku se može jednostavno dešifrirati operacijom $M^{1/3}$ (treći korijen nad M) pošto vrijedi sljedeće:

$M^3 \bmod n = M^3$, ukoliko je $M \leq n^{1/3}$

odnosno:

$(M^3)^{1/3} = M$

3.2.1. Obrana

Obrana od ove specifične vrste napada prilično je jednostavna. Ukoliko je poruka koju treba šifrirati kratka toliko da je duljina poruke M kraća od $n^{1/3}$, poruku je potrebno nadopuniti slučajnom ispunom dok ne bude zadovoljen uvjet $M>n^{1/3}$, na taj način se osigurava da će poruka M biti reducirana modulo operacijom.

3.3. Napad korištenjem odabranog šifriranog teksta

Napad korištenjem odabranog teksta (engl. *chosen chipertext attack*) temelji se na tom da napadač na neki način uspije doći do šifriranog teksta po njegovom izboru.
Napadač prisluškuje komunikacijski kanal kroz koji se razmjenjuju RSA šifrirane poruke, dolazi do šifrirane poruke C, te želi saznati njen izvorni sadržaj, odnosno matematički promatrano želi saznati $M=C^d \bmod n$.
Uz pretpostavku da mu je poznat javni ključ (e,n), da bi došao do M napadač prvo bira slučajnu poruku R, takvu da je $R<n$, te zatim šifrira R poznatim javnim ključem:

$X=R^e \bmod n$

Šifriranu poruku C multiplicira korištenjem dobivenog X:

$Y=X*C \bmod n$

Također računa modulo inverzne vrijednosti od R

$T=R^{-1} \bmod n$

Pri tom napadač podrazumijeva da vrijedi sljedeće svojstvo:

ako je $X=R^e \bmod n$, onda je $R=X^d \bmod n$

Napadač mora čekati da korisnik digitalno potpiše Y svojim privatnim ključem, čime efektivno dešifrira Y, te šalje $U=Y^d \bmod n$ napadaču. Napadač tada računa sljedeće:

$$T*U \bmod n = (R^{-1} \bmod n)*(Y^d \bmod n) \bmod n =$$
$$(R^{-1} \bmod n)*[(X*C \bmod n)^d \bmod n] \bmod n =$$
$$(R^{-1} \bmod n)*[(X*C)^d \bmod n] \bmod n =$$
$$(R^{-1} \bmod n)*[(X^d \bmod n)*(C^d \bmod n) \bmod n] \bmod n =$$

$(R^{-1} \bmod n)*(R*M \bmod n) \bmod n =$
$R^{-1}*R*M \bmod n = M$

3.3.1. Obrana

Ovakav napad podrazumijeva neku vrst interakcije korisnika s napadačem. Da bi napad funkcionirao napadač mora korisniku podmetnuti proizvoljnu poruku koju zatim korisnik treba digitalno potpisati.
Obrana od ovakve vrste napada temelji se na tome da korisnik nikad ne potpisuje niti jedan dokument koji mu se šalje, već samo potpisuje vrijednost jednosmjerne funkcije izračunate nad tim dokumentom.

3.4. Vremenski bazirani napad

Napretkom kriptografije temeljene na javnim ključevima uočene su neke činjenice i pravilnosti. Npr. modulo i eksponencijalne operacije koje se koriste za RSA za obradu zahtijevaju diskretne vremenske intervale. Ukoliko se RSA operacije izvode korištenjem teorema o kineskim ostacima (engl. *Chinese Remainder Theorem*) napadač može iskoristiti malene vremenske razlika prilikom izvođenja RSA operacija, i u mnogim slučajevima na temelju doga otkriti d. Napad se temelji na pasivnom prisluškivanu RSA operacija.
Napadač pasivno promatra k operacija i mjeri vrijeme t potrebno za računanje $M=C^d \bmod n$. Pretpostavka je da napadač pozna C i n. Pseudokod ovakvog, vremenski baziranog napada (engl. *timing attack*) je sljedeći:

```
s0=1
za i=0 do w-1 (w je broj bitova od d)
  ako je (bit i od d) je 1 onda
        Mi=(si*C) mod n
  inače
        Mi=si
  si+1=Mi^2 mod n
kraj
```

Na ovaj način napadač, ukoliko poznaje bitove eksponenta d od 0 do $i-1$, može pronaći bit i. Da bi došao do čitavog eksponenta počinje sa $i=0$ i ponavlja napad dok nije poznat čitav eksponent.
Pošto je poznato prvih i bitova eksponenta d, napadač može izračunati prvih i iteracija `for` petlje da bi pronašao vrijednost s_i. Sljedeća iteracija zahtijeva prvi nepoznati bit eksponenta. Ukoliko je vrijednost tog bita 1, računa se $M_i=(s_i*C) \bmod n$. Ukoliko je pak vrijednost tog bita 0, operacija se ne izvodi.
Pretpostavka je da ciljni sustav koristi modulo mutliplikativnu funkciju koja je u nekim slučajevima jako brza, a u nekim slučajevima pak zahtijeva mnogo više vremena neko čitavo modulo eksponenciranje . Za nekoliko vrijednosti s_i i C računanje $M_i=(s_i*C) \bmod n$ biti će izrazito dugo, a napadač, ukoliko poznaje dizajn ciljnog sustava može odrediti koje su to vrijednosti. Može se pretpostaviti; ukoliko spore $M_i=(s_i*C) \bmod n$ operacije uvijek rezultiraju dugim ukupnim vremenom potrebnim za modulo eksponenciranje, vrijednost bita eksponenta je vjerojatno 1. Jednom kada je poznat bit i, napadač može provjeriti da je ukupno vrijeme obrade dugo ukoliko se očekuje da $s_{i+1}=M_i^2 \bmod n$ bude dugo. Isti način mjerenja vremena može se ponovno iskoristiti da bi se pronašli sljedeći bitovi eksponenta.
Ukoliko se pak bit i eksponenta krivo pretpostavi, vrijednosti za $M_{k \geq i}$ biti će pogrešne, s gledišta napadača slučajne. Vrijeme potrebno za multiplikaciju nakon pogreške neće se očitavati u ukupnom vremenu potrebnom za eksponenciranje. Na taj način u napad je ugrađen mehanizam detekcije pogreške, pošto se nakon pogrešne pretpostavke više ne mogu provoditi smislene usporedbe. Npr., napadač može imati popis najvjerojatnijih posrednih eksponenata zajedno sa vrijednostima koje predstavljaju vjerojatnost da su oni ispravni. Ukoliko trenutno odabrana vrijednost nije točna, njenu vjerojatnost treba smanjiti, dok se vjerojatnosti ispravnih vrijednosti trebaju povećati. Tehnike detekcije i korekcije pogreške zahtijevaju veće memorijske i procesorske resurse, ali umanjuju broj potrebnih uzoraka.

Ovakav napad zahtijeva praćenje kriptografskih operacija u stvarnom vremenu, što uvelike ograničava mogućnosti provođenja.

3.4.1. Obrana

Postoji nekoliko načina obrane od vremenski baziranih napada. Najjednostavniji način je osigurati da kriptografske operacije traju vremenski ujednačeno, bez obzira na podatke koji se kriptografski obrađuju. Kod RSA je za to dovoljno osigurati da modulo multiplikacija uvijek bude vremenski jednaka, bez obzira na operande.
Drugi način zaštite od vremenski baziranog napada je maskiranje (engl. *blinding*). Podaci se maskiraju prije obrade, zatim se vrše kriptografske operacije, te se na kraju maska uklanja. U ovom slučaju kriptografske operacije ne traju jednako, no operacije maskiranja i uklanjanja maske zahtijevaju određeni vremenski period (koji nije jednak, već slučajan u određenim okvirima), tako da vrijeme obrade u tom slučaju ne ovisi o operandima.
Maskiranje se provodi uvođenjem slučajne vrijednosti R. Standardni postupak dešifriranja

$$M = C^d \bmod n$$

postaje:

$$M = R^{-1}(CR^e)^d \bmod n \quad \text{(pošto vrijedi } R = (R^e \bmod n)^d \bmod n)$$

Gdje je R slučajna varijabla, a R^{-1} njena inverzna vrijednost (u smislu modulo aritmetike).

4. Zaključak

Postoji nekoliko vrsta napada na RSA algoritam koji u određenim slučajevima mogu biti uspješni. Čak i napad primjenom sile, uz odgovarajuće resurse može polučiti pozitivne rezultate.
No isto tako, pravilnom uporabom algoritma, odnosno izbjegavanjem šifriranja kratkih poruka, digitalnog potpisivanja nasumičnih dokumenata, te maskiranjem kriptografskih operacija ti napadi se mogu efikasno onemogućiti. Također, adekvatnim odabirom duljine kriptografskih ključeva moguće je osujetiti i napade primjenom sile.
Može se zaključiti da RSA algoritam i gotovo tri desetljeća nakon svoje pojave i dalje predstavlja sigurno rješenje, čija uporaba uz dosad poznate tehnike napada nije ugrožena.

Dodatak A: Svojstva modulo aritmetike

1. postojanje identiteta:
   $a+0 \bmod n = 0+a \bmod n = a$
   $a*1 \bmod n = 1*a \bmod n = a$
2. postojanje inverzije:
   $a+(-a) \bmod n = 0$
   $a*a^{-1} \bmod n = 1$, ukoliko je $a \neq 0$
3. komutativnost:
   $a+b \bmod n = b+a \bmod n$
   $a*b \bmod n = b*a \bmod n$
4. asocijativnost:
   $a+(b+c) \bmod n = (a+b)+c \bmod n$
   $a*(b*c) \bmod n = (a*b)*c \bmod n$
5. distributivnost:
   $a*(b+c) \bmod n = [(a*b)+(a*c)] \bmod n$
6. reduciranje:
   $(a+b) \bmod n = [(a \bmod n) + (b \bmod n)] \bmod n$
   $(a*b) \bmod n = [(a \bmod n) * (b \bmod n)] \bmod n$

Dodatak B: Teorem o kineskim ostacima i primjena na RSA

Definicije:
Rezidua je ostatak cjelobrojnog dijeljenja s brojem koji se naziva modul.
Prikaz pomoću rezidua broja X je skup rezidua $\{r_1, r_2, \ldots, r_k\}$, obzirom na module $\{m_1, m_2, \ldots, m_k\}$.
Uz ove definicije teorem o kineskim ostacima tvrdi da, ukoliko se broj X prikaže uz pomoć skupa rezidua $\{r_1, r_2, \ldots, r_k\}$, moguće je odrediti X, ukoliko je najveći zajednički djelitelj bilo kojeg para modula 1, odnosno vrijedi $(ri, rj)=1$, $i \neq j$, tj. moduli su međusobno relativno prosti.
Posljedica ovog teorema jest da ukoliko se broj X prikaže uz pomoć skupa rezidua, operacije nad tim skupom mogu se izvoditi neovisno. Nakon izvršenja tih operacija, konačni rezultat se može rekonstruirati korištenjem teorema. Pošto su rezidue manje nego sam broj X, na taj način se značajno smanjuje vremenska složenost operacija.
Uz pomoć toga, umjesto da se izvodi eksponenciranje i $\mathrm{mod}\ n$ operacija, postupak se dijeli na dvije modulo operacije; $\mathrm{mod}\ p$ i $\mathrm{mod}\ q$. Pošto su p i q relativno prosti, zadovoljavaju zahtjeve za rekonstrukciju korištenjem teorema o kineskim ostacima, te se eksponenciranje provodi na sljedeći način:

$$M_1 = C^d \bmod p$$
$$M_2 = C^d \bmod q$$

Rezultat jesu dvije poruke M1 i M2 . Iz tih poruka moguće je rekonstruirati originalnu poruku na sljedeći način:

$$M = M_1 * (q^{-1} \bmod p)q + M_2 * (p^{-1} \bmod q)p \bmod n$$

Postupak se može još ubrzati korištenjem alternativnih metoda.