Kriptografija

Vesna Zakošek

28. december 2004

Kriptografija je veda o sporazumevanju v prisotnosti nasprotnikov. Njen glavni cilj je omogočiti dvema osebama sporazumevanje, tako da nasprotnik, ki prestreže sporočilo, ne razume njegove vsebine, torej iz njega ne more izvedeti nobene informacije. Osnovni model za delo v kriptografiji je *kriptosistem*. Ta je sestavljen iz:

- končne množice sporočil $\mathcal{M}$,
- končne množice kod $\mathcal{C}$,
- končne množice ključev $\mathcal{K}$,
- za vsak ključ $K \in \mathcal{K}$ obstaja kodirno pravilo $e_K : \mathcal{M} \to \mathcal{C}$ in ustrezno dekodirno pravilo $d_K : \mathcal{C} \to \mathcal{M}$, da za vsako sporočilo $M \in \mathcal{M}$ velja $d_K(e_K(M)) = M$.

Kodirna funkcija e_K mora biti injektivna, sicer dekodiranje ni enolično.

Ločimo *kriptografijo*, ki je veda o sestavljanju kriptosistemov, *kriptoanalizo*, ki je veda o razreševanju kriptosistemov in *kriptologijo*, unijo kriptografije in kriptoanalize. Pri sestavljanju kriptosistemov navadno upoštevamo *Kerckhoffovo načelo*: vedno se predpostavlja, da nasprotnik pozna kriptosistem, ki ga uporabljamo.

Pri sestavljanju kriptosistemov je klasičen cilj *varnost*. Kriptosistem je varen, če je praktično nemogoče, da bi nasprotnik, ki pozna kodo $C = e_K(M)$, a ne pozna ključa K, lahko razbral kakršenkoli del sporočila M. Včasih so bili sistemi zasnovani tako, da nasprotnik, ki je poznal le kodo C, nikakor ne bi mogel najti sporočila M (npr. Vernonov enkratni ščit, kjer je $C = M \oplus K$, $\oplus$ je XOR bitna operacija). Moderna kriptografija pa temelji na tem, da je nasprotnikova naloga računsko neobvladljiva (npr. DES in RSA). Sistem je dokazano varen, ko ima dovolj visoko spodnjo mejo računskih korakov, potrebnih za rešitev sistema. Ponavadi le primerjamo kompleksnost sistema z drugimi že znanimi NP problemi, spodnje meje pa ne znamo dokazati, ker še vedno ne vemo, ali je $P \neq NP$.

Poleg varnosti želimo v kriptosistemu zagotoviti tudi, da je prejemnik sporočilo *zanesljivo* sprejel takega, kot ga je pošiljatelj poslal (torej, da ga

nasprotnik vmes ni spremenil). Za reševanje tega problema se uporablja digitalno podpisovanje, ki temelji na podobnih principih kot kodiranje.

Ločimo *kriptosisteme s tajnim ključem* (DES) in *kriptosisteme z javnim ključem* (RSA).

1 DES

DES oz. Data Encryption Standard je kriptosistem s tajnim ključem. Od leta 1977 je bil standard za netajne aplikacije. Leta 2000 so sprejeli AES (Advanced Encryption Standard), ki naj bi ga zamenjal.

Kodiranje DES

Sporočilo $M \in \mathcal{M}$ je 64-bitni niz,
ključ $K \in \mathcal{K}$ je 56-bitnih niz,
koda $C \in \mathcal{C}$ je 64-bitnih niz.

1. M permutiramo z začetno (fiksno) permutacijo IP in dobimo $M_0 = IP(M) = L_0R_0$, kjer sta L_0 in R_0 oznaki za levih in desnih 32 bitov M_0.

2. za $i = 1, \ldots, 16$
   $$L_i = R_{i-1},$$
   $$R_i = L_{i-1} \oplus f(R_{i-1}, K_i),$$
   kjer je $\oplus$ XOR bitna operacija, f funkcija in K_i niz 48 bitov, ki jih dobimo kot izbor bitov iz ključa K.

3. $C = IP^{-1}(R_{16}L_{16})$.

Pri tem funkcija f za vhodni spremenljivki vzame 32-bitni in 48-bitni niz ter vrne 32-bitni niz. Operacije, ki se izvajajo pri računanju vrednosti funkcije f, so linearne z izjemo osmih S-škatel: 4×16 matrike, ki so nujne za varnost DES.

Dekodiranje DES

Algoritem je isti, le da ključe $K_1, \ldots, K_{16}$ jemljeno v nasprotnem vrstnem redu $(K_{16}, \ldots, K_1)$.

Problem DES

Ker so vse operacije razen S-škatel linearne in je $|\mathcal{K}|$ le 2^{56}, torej se da množico ključev s superračunalniki raziskati v dosegljivem času, DES ni ravno popolnoma varen. Ravno zato se ponavadi uporablja DES3, kjer imamo tri tajne ključe, zaradi česar je bolj varen, a trikrat počasnejši.

Uporaba DES

DES se veliko uporablja predvsem v bančništvu, npr. pri kodiranju PIN-ov (Personal Identification Code) ter pri transakcijah na računih. Drugi primeri uporabe: v vladnih organizacijah in pri kodiranju plačniške TV.

2 RSA

RSA so leta 1977 predlagali Rivest, Shamir in Adleman in je primer kriptosistema z javnim ključem. Ideja kriptosistema z javnim ključem je naslednja: Problem pri kriptosistemu s tajnim ključem je, da se morata sporazumevajoči se osebi prej varno dogovoriti glede ključa K, ki ga bosta uporabljala, česar pa včasih ni lahko doseči. Pri kripstositemih z javnim ključem lahko sporočilo kodira vsak, dekodira pa ga lahko le oseba, ki pozna določeno informacijo, bližnjico. Kodirna funkcija je *enosmerna funkcija z bližnjico*, t. j.: če poznamo kodirno funkcijo, je ob nepoznavanju bližnjice računsko neobvladljivo izračunati njen inverz (dekodirno funkcijo). Če torej A želi poslati sporočilo B, lahko A z javno kodirno funkcijo zakodira sporočilo, le B pa ga zna dekodirati. Če dobi sporočilo v roke nasprotnik, ga v dosegljivem času ne bo znal dekodirati; A in B pa se ni potrebno predhodno dogovoriti glede K.

Opomba: Primer uporabe enosmerne funkcije (brez bližnjice) f: prijava v računalniškem sistemu: namesto gesla w shranimo $f(w)$. Pravilnost gesla se ob prijavi zlahka ugotavlja (samo izračuna se $f(w)$ in preveri, če ustreza shranjenemu geslu), nihče pa ne more ugotoviti gesla (niti administrator, ker iz $f(w)$ ne zna izračunati w). To npr. uporabljamo na forumu.

Algoritem RSA
Izberemo dovolj veliki praštevili p, q in izračunamo $n = p \cdot q$.

$$\mathcal{M} = \mathcal{C} = \mathbf{Z}_n,$$
$$K = (n, p, q, a, b),$$

kjer je $a \cdot b \equiv 1 \ mod \ \varphi(n)$, ($\varphi$ je Eulerjeva funkcija - številu n priredi število elementov iz $\mathbf{Z}_n$ tujih proti n).

$$e_K(x) \equiv x^b mod \ n, x \in \mathcal{M}$$

$$d_K(y) \equiv y^a mod \ n, y \in \mathcal{C}$$

(n, b) je javni ključ, medtem ko je (p, q, a) tajni.
V primeru, da komunicirata A in B, ter B prejema sporočila, je postopek sporazumevanja sledeč:

1. B generira dve veliki praštevili p in q ter izračuna $n = pq$ in $\varphi(n) = (p-1)(q-1)$
2. B izbere naključno število $b \in \mathbf{Z}_{\varphi(n)}$, da je $gcd(b, \varphi(n)) = 1$ (ta pogoj mora biti izpolnjen, da bo b sploh imel inverz v $\mathbf{Z}_{\varphi(n)}$)
3. B izračuna $a \equiv b^{-1} mod \ \varphi(n)$ s pomočjo razširjenega Evklidovega algoritma

4. B objavi (n, b)

5. A želi poslati sporočilo $x \in \mathbf{Z}_n$, zato pošlje kodo $y \equiv x^b mod\ n$

6. B dekodira kodo y v sporočilo $x \equiv y^a mod\ n$

Za dekodiranje sporočila bi nasprotnik potreboval a, za izračun le-tega pa $\varphi(n)$. Najlažji način za izračun $\varphi(n)$ pa je faktorizacija n. Torej je e_K enosmerna funkcija z bližnjico, saj je problem računanja a enako težek kot faktorizacija n, ki pa je NP problem; bližnjica je poznavanje faktorizacije n. RSA temelji na dejstvu, da iskanje velikih praštevil teče v polinomskem času, faktorizacija števila pa teče v $\mathcal{O}(e^{c\sqrt{log(n)log(log(n))}})$. Trenutni faktorizacijski algoritmi so zmožni faktorizacije števila z največ 130 števkami. Ker se v večini realizacij RSA uporabljajo 512-bitni n, ki imajo 154 decimalk, dolgoročno gledano to ni dobra rešitev in jih bo treba posodobiti, da bo sistem še naprej varen.

Problem RSA

- je približno 1500-krat počasnejši od DES,
- za inicializacijo potrebujemo velika praštevila, za kar imamo algoritme, ki zgenerirajo naključna k-bitna praštevila in tečejo v polinomskem času:
  Monte Carlo verjetnostni algoritmi, ki se zmotijo z zelo majhno verjetnostjo in so v praksi zelo učinkoviti, saj vedno tečejo v polinomskem času,
  Las Vegas verjetnostni algoritmi, ki so vedno pravilni, a tečejo le v pričakovanem polinomskem času.

Uporaba RSA - digitalno podpisovanje

Digitalno podpisovanje je metoda podpisovanja sporočila shranjenega v digitalni obliki in naj bi, podobno kot lastnoročni podpis, enolično identificiral osebo, ki je sporočilo oddala. Z digitalnim podpisovanjem želimo zagotoviti predvsem *zanesljivost*: da je prejemnik sporočilo dobil tako, kot ga je podpisani pošiljatelj poslal. Seveda mora imeti vsak pošiljatelj svojo (tajno) *podpisovalno* funkcijo, ki za dano sporočilo vrne ustrezen podpis: $sig_A(x)$ je podpis osebe A za sporočilo x. Vsak, pa lahko za dano sporočilo in podpis preveri, ali podpis y res ustreza sporočilu x, če naj bi bil podpisani oseba A: (javna) funkcija za preverjanje $ver_A(x, y)$ mora vrniti $true$.

S pomočjo RSA lahko definiramo $sig_K := d_K$ in $ver_K := e_K$. Zdaj bo A lahko poslal sporočilo x B in se nanj podpisal z $sig_A(x)$ tako, da njegovega podpisa nihče ne bo mogel ponarediti. Poslal mu bo $z = e_B(x, sig_A(x))$. Ko bo B prejel sporočilo, ga bo najprej dekodiral $(x, y) = d_B(z)$ in potem preveril, če podpis res spada k sporočilu: ali je $ver_A(x, y) = true$?