6. Digitalni podpisi

- uvod (podpis z RSA sistemom)
- ElGamalov sistem za digitalno podpisovanje
- Digital Signature Standard
- napadi
- enkratni podpis
- podpisi brez možnosti zanikanja
- Fail-stop podpisi

Digitalni podpis je nadomestek za lastnoročni podpis pri elektronski izmenjavi in digitalnemu hranjeju podatkov.

Konceptualno se način zapisovanja informacij ni dramatično spremenil.

Medtem ko smo prej shranjevali in prenašali informacije na papirju, jih sedaj hranimo na magnetnih in drugih medijih ter jih prenašamo preko telekomunikacijskih sistemov (tudi brezžičnih).

Bistveno pa se je spremenila možnost kopiranja in spreminjanja informacij.

Zlahka naredimo na tisoče kopij neke informacije, ki je shranjena digitalno, pri tem pa se nobena ne razlikuje od originala.

Z informacijo na papirju je to precej težje.

Družba, v kateri so informacije spravljene in prenašane v digitalni obliki, mora poskrbeti za to, da ne bo varnost informacij odvisna od fizičnega medija, ki jih je zapisal ali prenesel.

Varnost informacij mora temeljiti izključno na digitalni informaciji.

Eno izmed osrednjih orodij pri zaščiti informacij je **podpis**. Le-ta preprečuje poneverjanje, je dokaz o izvoru, identifikaciji, pričanju.

Podpis naj bi bil unikat vsakega posameznika, z njim se predstavimo, potrdimo, pooblastimo.

Z razvojem digitalne informacije moramo ponovno obdelati tudi koncept podpisa.

Ni več unikat, ki enolično določa podpisnika, kajti elektronsko kopiranje podpisa je tako lahko, da je skoraj trivialno na nepodpisan dokument pripeti poljuben podpis.

Potrebujemo protokole, ki imajo podobne lastnosti kot trenutni “papirni protokoli”.

Družba ima enkratno priložnost, da vpelje nove in učinkovitejše načine, ki nam bodo zagotovili varnost informacij.

Veliko se lahko naučimo iz dosedanjih sistemov, obenem pa moramo odpraviti tudi številne pomanjkljivosti.

Primerjava digitalnega in navadnega (lastnoročnega) podpisa:

- navadni podpis je fizično del podpisanega dokumenta;
- navadni podpis preverjamo s primerjanjem, digitalnega z algoritmom, katerega rezultat je odvisen od ključa in dokumenta;
- kopija digitalnega podpisa je identična originalu;
- digitalni podpis je odvisen od dokumenta, ki ga podpisujemo.

Sistem za digitalno podpisovanje je peterka $(\mathcal{P}, \mathcal{A}, \mathcal{K}, \mathcal{S}, \mathcal{V})$, za katero velja

1. $\mathcal{P}$ je končna množica sporočil,
2. $\mathcal{A}$ je končna množica podpisov,
3. $\mathcal{K}$ je končna množica ključev,
4. $\forall$ ključ $K \in \mathcal{K}$ obstaja algoritem za podpisovanje

$$\mathrm{sig}_K \in \mathcal{S}, \quad \mathrm{sig}_K : \mathcal{P} \to \mathcal{A}$$

in algoritem za preverjanje podpisa

$$\mathrm{ver}_K \in \mathcal{V}, \quad \mathrm{ver}_K : \mathcal{P} \times \mathcal{A} \to \{\mathrm{true}, \mathrm{false}\}.$$

Funkciji sig_K in ver_K imata to lastnost, da za vsako sporočilo $x \in \mathcal{P}$ in vsak podpis $y \in \mathcal{A}$ velja

$$\mathrm{ver}_K(x, y) = \begin{cases} \mathrm{true}, & \text{če } y = \mathrm{sig}_K(x) \\ \mathrm{false}, & \text{če } y \neq \mathrm{sig}_K(x) \end{cases}$$

Zahteve:

- algoritma sig_K in ver_K imata polinomsko časovno zahtevnost
- sig_K je znan le podpisniku
- ver_K je splošno znan
- računsko mora biti nemogoče ponarediti podpis

Primer: Algoritem RSA lahko uporabimo tudi za podpisovanje. Naj bo $n = pq$, kjer sta p in q praštevili.

Če je (n, d) skriti ključ, (n, e) pa javni, pri čemer je $de \equiv 1 \pmod{\varphi(n)}$, potem definiramo:

$$\mathrm{sig}_K(x) = d_K(x) = x^d \bmod n$$

$$\mathrm{ver}_K(x, y) = \mathrm{true} \iff x = e_K(y) = y^e \bmod n$$

za $x, y \in \mathbb{Z}_n$.

Z zgornjim algoritmom je mogoče ponarediti podpis naključnih sporočil.

Ponarejevalec najprej izbere podpis y in nato izračuna

$$x \equiv y^e \pmod{n}.$$

Možnosti takega ponarejanja se izognemo z

- enosmernimi zgoščevalnimi funkcijami ali
- zahtevo, da ima sporočilo x določen pomen.

Pošiljanje podpisanih tajnih sporočil

Vrstni red šifriranja in digitalnega podpisovanja je pomemben.

1. Najprej podpisovanje:

$$x, \mathrm{sig}_{\mathrm{Anita}}(x) \to e_{\mathrm{Bojan}}((x, \mathrm{sig}_{\mathrm{Anita}}(x))).$$

2. Najprej šifriranje $z = e_{\mathrm{Bojan}}(x)$,

   potem podpis $y = \mathrm{sig}_{\mathrm{Anita}}(z)$:

   Bojan prejme (z, y), odšifrira tajnopis

   $x = d_{\mathrm{Bojan}}(z)$ ter preveri podpis $\mathrm{ver}_{\mathrm{Anita}}(z, y)$.

V drugem primeru lahko napadalec Cene zamenja Anitin podpis s svojim:

$$y' = \mathrm{sig}_{\mathrm{Cene}}(z) \quad \to (z, y') \quad \to x = d_{\mathrm{Bojan}}(z),$$

$$\mathrm{ver}_{\mathrm{Cene}}(z, y')$$

in Bojan bo mislil, da je sporočilo prišlo od Ceneta.

Zato se priporoča najprej podpisovanje in nato šifriranje.

V primeru algoritma RSA je potrebno pri zaporednem podpisovanju in šifriranju paziti na velikosti modulov (*reblocking problem*).

Če je $n_{\text{Anita}} > n_{\text{Bojan}}$, se lahko zgodi, da Bojan ne bo mogel razvozlati sporočila. Naj bo

$$\begin{aligned}(n_{\text{Anita}}, e_{\text{Anita}}, d_{\text{Anita}}) &= (62894113, 5, 37726937),\\ (n_{\text{Bojan}}, e_{\text{Bojan}}, d_{\text{Bojan}}) &= (55465219, 5, 44360237).\end{aligned}$$

Anita podpiše sporočilo $x = 1368797$ in podpis zašifrira:

1. $s = x^{d_{\text{Anita}}} \bmod n_{\text{Anita}} = 59847900$,
2. $y = s^{e_{\text{Bojan}}} \bmod n_{\text{Bojan}} = 38842235$.

Bojan izračuna

1. $\widehat{s} = y^{d_{\text{Bojan}}} \bmod n_{\text{Bojan}} = 4382681$,
2. $\widehat{x} = \widehat{s}^{e_{\text{Anita}}} \bmod n_{\text{Anita}} = 54383568$.

Ker je $s > n_{\text{Bojan}}$, je $\widehat{x} \neq x = 1368797$.

Verjetnost tega dogodka je

$$\frac{n_{\text{Anita}} - n_{\text{Bojan}}}{n_{\text{Anita}}}.$$

Delitev shem za digitalno podpisovanje

1. Podpis je dodatek (ElGamal, DSA) sporočilu - sporočilo je možno rekonstruirati iz podpisa (RSA),
2. deterministični - nedeterministični,
3. enkratni - večkratni.

Različni sistemi za digitalno podpisovanje

- RSA
- ElGamal, DSS (*Digital Signature Standard*)
- Enkratni podpisi (*one-time signatures*)
- Slepi podpisi (*blind signatures*)
- Podpisi brez možnosti zanikanja (*undeniable signatures*)
- Skupinski podpisi (*group signatures*)
- Fail-Stop podpisi

ElGamalov sistem za digitalno podpisovanje

Za razliko od algoritma RSA je ElGamalov sistem namenjen predvsem digitalnemu podpisovanju, čeprav se ga da v posebnih primerih uporabiti tudi za šifriranje.

Podpis je nedeterminističen (odvisen od naključnega števila), torej sploh ni natanko določen.

Algoritem

Naj bo p takšno praštevilo, da je v $\mathbb{Z}_p$ težko izračunati diskretni logaritem in $\alpha \in \mathbb{Z}_p^*$ primitivni element.

Naj bo še $\mathcal{P} = \mathbb{Z}_p^*$, $\mathcal{A} = \mathbb{Z}_p^* \times \mathbb{Z}_{p-1}$ in

$$\mathcal{K} = \{(p, \alpha, a, \beta) : \beta \equiv \alpha^a \pmod{p}\}.$$

Število a je skrito (zasebno),

števila p, α in β pa so javno znana.

Podpisovanje: podpisnik s ključem $K = (p, \alpha, a, \beta)$ izbere naključno skrito število $k \in \mathbb{Z}_{p-1}^*$ in določi

$$\mathrm{sig}_K(x, k) = (\gamma, \delta),$$

kjer je

$$\gamma \equiv \alpha^k \bmod p$$

in

$$\delta \equiv (x - a\gamma)k^{-1} \bmod (p-1).$$

Preverjanje podpisa: (samo z javnimi p, α in β)

$$\mathrm{ver}_K(x, \gamma, \delta) = \mathrm{true} \iff \beta^\gamma \gamma^\delta \equiv \alpha^x \pmod p.$$

Primer: Naj bo $p = 467, \alpha = 2$ in $a = 127$.
Potem je $\beta \equiv \alpha^a \bmod p = 132$. Recimo, da želimo podpisati $x = 100$, izbrali pa smo si tudi $k = 213$.
Podpis je enak (γ, δ), kjer je

$$\gamma \equiv 2^{213} \bmod 467 = 29$$

in

$$\delta \equiv (100 - 127 \cdot 29)\ 431 \bmod 466 = 51.$$

Pri preverjanju izračunamo

$$132^{29}\ 29^{51} \equiv 189 \bmod 467 \quad \text{in}$$

$$2^{100} \equiv 189 \bmod 467.$$

Zadnji vrednosti se ujemata, zato je podpis pravi.

Varnost ElGamalovega sistema za podpisovanje

Kako bi lahko ponaredili podpis, ne da bi vedeli za vrednost skritega števila a?

1. Za dano sporočilo x je potrebno najti tak par (γ, δ), da bo veljalo $\beta^\gamma \gamma^\delta \equiv \alpha^x \pmod p$, torej
   - če izberemo γ: rabimo $\delta = \log_\gamma \alpha^x \beta^{-\gamma} \pmod p$,
   - če izberemo δ: glede na γ je potrebno rešiti enačbo $\beta^\gamma \gamma^\delta \equiv \alpha^x \pmod p$,
   - hkrati računamo γ in δ (zaenkrat ni še nihče odkril hitrega postopka za reševanje zgornje enačbe).

2. Za podpis (γ, δ) je potrebno najti ustrezno sporočilo x:

$$x = \log_\alpha \beta^\gamma \gamma^\delta \pmod p.$$

3. Hkratno računanje x, γ in δ: naj bosta i in j takšni števili, da velja $0 \le i, j \le p-2$ in $D(j, p-1) = 1$. Potem števila

$$\begin{aligned} \gamma &\equiv \alpha^i \beta^j \pmod p, \\ \delta &\equiv -\gamma j^{-1} \pmod{(p-1)}, \\ x &\equiv -\gamma i j^{-1} \pmod{(p-1)} \end{aligned}$$

zadoščajo enačbi $\beta^\gamma \gamma^\delta \equiv \alpha^x \pmod p$.

Primer: Če je $p = 467, \alpha = 2$ in $\beta = 132$, lahko z izbiro $i = 99$ in $j = 179$, dobimo veljaven podpis $(117, 41)$ za sporočilo 331.

4. Ali lahko pri veljavnem podpisu (γ, δ) za x najdemo še kakšen podpis za neko drugo sporočilo x'?
   Odgovor je "DA".

   Naj bodo h, i in j takšna števila, da zanje velja $0 \le h, i, j \le p-2$ in $D(h\gamma - j\delta, p-1) = 1$.
   Potem je par (λ, μ) veljaven podpis za x', kjer je

$$\begin{aligned} \lambda &= \gamma^h \alpha^i \beta^j \bmod p, \\ \mu &= \delta\lambda(h\gamma - j\delta)^{-1} \bmod (p-1), \\ x' &= \lambda(hx + i\delta)(h\gamma - j\delta)^{-1} \bmod (p-1). \end{aligned}$$

Nevarnosti pri napačni uporabi ElGamalovega sistema

1. Če naključno število k ne ostane skrito, lahko izračunamo

$$a = (x - k\delta)\gamma^{-1} \bmod (p-1).$$

2. Število k lahko uporabimo le enkrat, sicer ga je mogoče zlahka izračunati.

Digital Signature Standard

DSS je modifikacija ElGamalovega sistema za podpisovanje.
Kot ameriški standard je bil predlagan leta 1991, sprejet pa leta 1994.

Algoritem: Naj bo p praštevilo velikosti L bitov, kjer je $512 \leq L \leq 1024$ in $64 \mid L$, q 160−bitno praštevilo, da $q \mid p-1$, ter $\alpha \in \mathbb{Z}_p^*$ $q-$ti koren enote po modulu p. Definirajmo $\mathcal{P} = \mathbb{Z}_p^*$, $\mathcal{A} = \mathbb{Z}_q \times \mathbb{Z}_q$ in

$$\mathcal{K} = \{(p, q, \alpha, a, \beta) : \beta \equiv \alpha^a \pmod{p}\}.$$

Vrednosti p, q, α in β so javne, število a pa skrito.

Podpisovanje: podpisnik izbere naključno skrito število $k \in \{1, \ldots, q-1\}$ in določi

$$\mathrm{sig}_K(x, k) = (\gamma, \delta),$$

kjer je

$$\gamma \equiv (\alpha^k \bmod p) \bmod q$$

in

$$\delta \equiv (x + a\gamma)\, k^{-1} \pmod{q}.$$

Za število δ mora veljati $\delta \not\equiv 0 \pmod{q}$.

Preverjanje podpisa: najprej izračunamo

$$e_1 \equiv x\delta^{-1} \quad \text{in} \quad e_2 \equiv \gamma\delta^{-1}.$$

Potem je

$$ver_K(x, \gamma, \delta) = \text{true}$$

$$\Updownarrow$$

$$(\alpha^{e_1}\ \beta^{e_2} \bmod p) \bmod q = \gamma.$$

Podobno kot pri ElGamalovi shemi je podpisovanje hitrejše od preverjanja (za razliko od RSA).

Prikrit kanal v algoritmu DSA

V algoritmu DSA obstaja prikrit kanal, ki omogoča:
(a) vključitev šifriranega sporočila v podpis, ki ga lahko prebere le tisti, ki pozna dodaten ključ;

(b) razkritje skritega ključa, brez vednosti njegovega lastnika.

Eno možnost za (a) si oglejmo na naslednji foliji,
točko (b) pa prihranimo za domačo nalogo.

Primer: Izberimo n tajnih praštevil $p_1, \ldots, p_n$ in poskusimo v podpis skriti binarno zaporedje $b_1, \ldots, b_n$. Naključno število k izbiramo toliko časa, da za vsak $1 \leq i \leq n$ velja

$$b_i = 1 \Longrightarrow \gamma \text{ je kvadratni ostanek po modulu } p_i,$$
$$b_i = 0 \Longrightarrow \gamma \text{ ni kvadratni ostanek po modulu } p_i,$$

kjer je $\mathrm{sig}_K(x, k) = (\gamma, \delta)$.

Napadi

Uganjevanje fraz, ki jih uporabljamo za gesla

primer	število znakov	zahtevnost	dolžina gesla	čas za razbijanje
mucka	5	25 (majhne črke)	12 bitov	40 minut
br1a9Az	7	62 (črke in številke)	24 bitov	22 let
TH,X1lb¡V+	10	95 (znaki na tipkov.)	40 bitov	nedosegljivo

Če uporabimo angleško ali slovensko besedo, dobimo
zaporedje s približno 1.3 biti entropije na en znak
(tj. prostor za besedo proti popolnoma naključnim znakom).

Napadi z grobo silo (angl. Brute Force Attack)

posameznik ima 1 PC in programsko opremo
($2^{17} - 2^{24}$ ključev/sek.),

majhna skupina, 16 PC
($2^{21} - 2^{28}$ ključev/sek.),

akademska omrežja, 256 PC
($2^{25} - 2^{32}$ ključev/sek.),

veliko podjetje z $1.000.000 za strojno opremo
(2^{43} ključev/sek.),

vojaška obveščevalna organizacija z $1.000.000.000 za strojno opremo in napredno tehnologijo
(2^{55} ključev/sek.).

Napadi z grobo silo

dolžina ključa (v bitih)	posamični napadalec	majhne skupine	razisko-valna omrežja	velika podjetja	vojaške obveščevalne službe
40	tedni	dnevi	ure	milisekunde	mikrosekunde
56	stoletja	desetletja	leta	ure	sekunde
64	tisočletja	stoletja	destletja	dnevi	minute
80	∞	∞	∞	stoletja	stoletja
128	∞	∞	∞	∞	tisočletja

Povprečen čas za napad z grobo silo

dolžina ključev (v bitih)	število možnih ključev	potreben čas za eno šifriranje/μsek.	potreben čas za 10^6 šifriranj/μsek.
32	$2^{32} = 4.3 \times 10^9$	$2^{31}\mu\text{sec} \approx 36$ min	≈ 2 milisek.
56	$2^{56} = 7.2 \times 10^{16}$	$2^{55}\mu\text{sec} \approx 1142$ let	≈ 10 ur
128	$2^{128} = 3.4 \times 10^{38}$	$2^{127}\mu\text{sec} \approx 5 \times 10^{24}$	$\approx 5 \times 10^{18}$ let

Napadi na PKS

Napadi na DSA

- Metoda Index Calculus ($p \approx 2^{1024}$)
- Pollardova ρ-metoda ($\sqrt{\pi q/2}$, $q \approx 2^{160}$)

Napadi na ECDSA

- Pollardova ρ-metoda ($\sqrt{\pi n/2}$, $n \approx 2^{160}$)

Programski napadi

MIPS računalnik lahko opravi 4×10^4 seštevanj točk na eliptični krivulji na sekundo.

(Ta ocena je precej konzervativna. Posebaj prirejeno integrirano vezje s frekvenco ure 40 MHz, ki opravlja operacije na eliptični krivulji nad obsegom $GF(2^{155})$ in lahko izvede 40.000 seštevanj na sekundo.)

Na osnovi tega zaključimo, da je število seštevanj na eliptični krivulji na $GF(2^{155})$ izvedeno na MIPS računalniku v času enega leta naslednje

$$(4 \times 10^4) \cdot (60 \times 60 \times 24 \times 365) \approx 2^{40}.$$

Spodnja tabela nam kaže kolikšno računsko moč potrebujemo za računanje problema diskretnega logaritma z uporabo Pollard ρ-methodo za različne vrednosti števila n. MIPS leto je ekvivalentno računski moči 1 MIPS računalnika, ki je na voljo eno leto.

velikost obsega (v bitih)	velikost števila n	$\sqrt{\pi n/2}$	MIPS let
155	150	2^{75}	3.8×10^{10}
210	205	2^{103}	7.1×10^{18}
239	234	2^{117}	1.6×10^{23}

Npr. če imamo na voljo 10.000 računalnikov z močjo 1.000 MIPS in je $n \approx 2^{150}$, potem je lahko problem diskretnega logaritma na eliptični krivulji rešen v 3.800 letih.

Prejšnjo tabelo je zanimivo primerjati s Odlyzko-vo tabelo, ki kaže kolikšno računsko moč potrebujemo za faktorizacijo celih števil s sedanjo verzijo splošnega NFS algoritma.

velikost števila n (v bitih)	MIPS let
512	3×10^4
768	2×10^8
1024	3×10^{11}
1280	1×10^{14}
1536	3×10^{16}
2048	3×10^{20}

Hardwarski napadi

Za bolj perspektiven napad (s strani dobro financiranega napadalca) na ECC, bi bilo potrebno narediti specializirano programsko opremo za paralelno iskanje na osnovi Pollard ρ-metode.

Van Oorschot and Wiener ocenjujeta:
za $n \approx 10^{36} \approx 2^{120}$ bi računalnik z $m = 325.000$ procesorji (cena okoli 10 milijonov USD) lahko izračunal diskretni logaritem v približno 35 dneh.

Poudariti moramo, da računanje diskretnega logaritma na $E(\mathbb{Z}_p)$ v zgoraj omenjenih napadih odkrije **en sam** *zasebni ključ.*

M. Blaze, W. Diffie, R. Rivest, B. Schneier, T. Shimomura, E. Thompson, and M. Wiener, January 1996, (`http://theory.lcs.mit.edu/rivest/publications.html`)
govorijo o minimalnih dolžinah ključev potrebnih za varen simetrični sistem (npr. DES ali IDEA):

Da bi zagotovili ustrezno zaščito proti najbolj resnim grožnjam (npr. velike komercialne ustanove in vladne agencije) mora ključ biti dolg vsaj 75 bitov. Za zaščito za naslednjih 20-let morajo ključi biti dolgi vsaj 90 bitov (pri tem upoštevamo pričakovano rast računske moči).

Če posplošimo te zaključke na eliptične kripto-sisteme, mora biti praštevilo n, ki zagotovlja kratkoročno varnost, dolgo vsaj 150 bitov, za srednjeročno varnost pa vsaj 180 bitov.

Dolžina ključev

simetrične šifre (AES)	**asimetrične** (RSA, DSA, DH)	**eliptične krivulje**
40 bitov	274 bitov	80 bitov
56 bitov	384 bitov	106 bitov
64 bitov	512 bitov	132 bitov
80 bitov	1024 bitov	160 bitov
96 bitov	1536 bitov	185 bitov
112 bitov	2048 bitov	237 bitov
120 bitov	2560 bitov	256 bitov
128 bitov	3072 bitov	270 bitov

Digitalni podpisi v $\mathbb{Z}_p$ in na EC

grupa	$\mathbb{Z}_p^*$	$E(\mathbb{Z}_p)$
elementi	množica celih števil $\{1, 2, \ldots, p-1\}$	točke (x, y), ki zadoščajo enačbi eliptične krivulje E in še točka v neskončnosti
operacija	množenje po modulu p	seštevanje točk na eliptični krivulji
oznake	elementi: g, h množenje: $g \times h$ multiplikativni inverz: h^{-1} deljenje: g/h potenciranje: g^a	elementi: P, Q seštevanje: $P + Q$ nasprotna točka: $-Q$ odštevanje: $P - Q$ skalarno množenje točke: aP
problem diskretnega logaritma	Za dana $g, h \in \mathbb{Z}_p^*$ poišči tako celo število a da je $h = g^a \bmod p$.	Za dani točki $P, Q \in E(\mathbb{Z}_p)$ poišči tako celo število a da je $Q = aP$.

Grupe

Digital Signature Algorithm (DSA), eliptični analog ECDSA

DSA	ECDSA
1. Izberi praštevili p in q velikosti $2^{1023}<p<2^{1024}$, $2^{159}<q<2^{160}$, tako da $q \mid p-1$.	1. Izberi tako eliptično krivuljo E: $y^2=x^3+ax+b$ nad $\mathbb{Z}_q$, da je število $\lvert E(\mathbb{Z}_p)\rvert$ deljivo s praštevilom $n \approx 160$-bitov.
2. $t \in \mathbb{Z}_p^*$, izračunaj $g = t^{(p-1)/q} \bmod p$, potem je $g \neq 1$ in ima red q v $\mathbb{Z}_p^*$.	2. Izberi točko P na $E(\mathbb{Z}_q)$ katere red je praštevilo n.
3. Uporabi multiplikativno grupo $\{g^0, g^1, \ldots, g^{q-1}\}$	3. Uporabi aditivno grupo $\{\mathcal{O}, P, 2P, \ldots, (n-1)P\}$

Generiranje ključa pri DSA in ECDSA

DSA	ECDSA
1. Izberi naključno celo število $x \in [2, q-2]$, tj. **zasebni ključ**	1. Izberi naključno celo število $d \in [2, n-2]$, tj. **zasebni ključ**
2. Izračunaj $y = g^x \bmod p$, **javni ključ** je (p, q, g, y).	2. Izračunaj $Q = dP$, **javni ključ** je (E, n, q, Q).

DSA	ECDSA
q	n
g	P
x	d
y	Q

Podpisovanje sporočila m

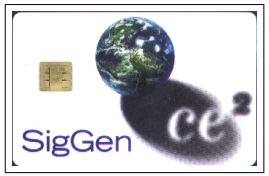
DSA	ECDSA
1. Izberi naključno celo število $k \in [2, q-2]$.	1. Izberi naključno celo število $k \in [2, n-2]$.
2. Izračunaj $g^k \bmod p$, $r = (g^k \bmod p) \bmod q$, $0 \neq s = k^{-1}(h(m) + xr) \bmod q$.	2. Izračunaj $kP = (x_1, y_1)$, $r = x_1 \bmod n$, $0 \neq s = k^{-1}(h(m) + dr) \bmod n$.
Podpis je par (r, s).	

Preverjanje podpisa (r, s) sporočila m osebe A

DSA	ECDSA
1. Preskrbi si avtentično kopijo javnega ključa osebe A: (p, q, g, y)	(E, n, q, Q)
2. Izračunaj $s^{-1} \bmod p$ in $h(m)$, $u_1 = h(m)s^{-1} \bmod q$, $u_2 = rs^{-1} \bmod q$, $v = (g^{u_1} y^{u_2} \bmod p) \bmod q$.	2. Izračunaj $s^{-1} \bmod n$ in $h(m)$ $u_1 = h(m)s^{-1} \bmod n$, $u_2 = rs^{-1} \bmod n$, $u_1 P + u_2 Q = (x_0, y_0)$ in $v = x_0 \bmod n$.
Sprejmi podpis samo in samo če je $v = r$.	

SigGen z EC

Razvita je bila v **Certicom Corp., Kanada**,
v sodelovanju s Schlumberger Smart Cards and Systems.



Uporablja Motorolin čip 68SC28:

- ROM 12.790 zlogov,
- EEPROM 8.112 zlogov,
- RAM 240 zlogov.

Vsebuje tehnologijo MULTIFLEX^{TM} ter
tehnologijo eliptičnih krivulj $(CE)^2$,
ki jo razvija podjetje Certicom Corp.

SigGen kartica je zelo prikladna za končnega uporabnika ter za proces prepoznavanja:

- je poceni,
- podpis je opravljen v pol sekunde,
- rabi samo 90 zlogov RAM-a,
- program ne zasede niti 4 KB.

Je edina pametna kartica, ki opravi digitalni podpis kar z obstoječim procesorjem.

Eliptični kripto-sistemi nudijo največjo moč glede na število bitov ključa med današnjimi javnimi kripto-sistemi.

Manjši ključi omogočajo

- manjše sistemske parametre,
- manjša potrdila z javnimi ključi,
- hitrejšo implementacijo,
- manjše zahteve po energiji,
- manjše procesorje,

itd.

Enkratni podpis

Z istim ključem lahko podpišemo le en dokument. Ponavadi algoritem temelji na enosmernih funkcijah.

Lamportova shema: $\mathcal{P} = \{0,1\}^{k\in\mathbb{N}}$, $|Y| < \infty$, enosmerna funkcija $f : Y \to Z$.

Naključno izberemo matriko $(y_{ij}) \in Y^{k\times 2}$ in določimo matriko enake velikosti z elementi $z_{ij} = f(y_{ij})$.

Ključ K sestavljata obe matriki, prva je skrita, druga pa javna.

Podpisovanje:

$$\mathrm{sig}_K(x_1,\ldots,x_k) = (y_{1,x_1},\ldots,y_{k,x_k}).$$

Preverjanje podpisa:

$$\mathrm{ver}_K(x_1,\ldots,x_k,a_1,\ldots,a_k) = \mathrm{true}$$

$$\Updownarrow$$

$$f(a_i) = z_{i,x_i}, \quad 1 \le i \le k.$$

Napadalec ne more ponarediti podpisa, saj ne more obrniti enosmerne funkcije f, da bi izračunal y-e.

Če pa bi podpisali dve različni sporočili z isto shemo, potem bi napadalec lahko poneveril podpis novih sporočil.

Primer: Naj bo $f(x) = 3^x \pmod{7879}$, ključ pa sestavljen iz matrik

$$\begin{pmatrix} 5831 & 735 \\ 803 & 2467 \\ 4285 & 6449 \end{pmatrix} \text{ in } \begin{pmatrix} 2009 & 3810 \\ 4672 & 4721 \\ 268 & 5731 \end{pmatrix}.$$

Potem je podpis za $x = (1,1,0)$ enak $(y_{1,1}, y_{2,1}, y_{3,0}) = (735, 2467, 4285)$.

Pomanjkljivost te sheme je velikost podpisa (za vsak bit čistopisa število med 1 in Y).

Spernerjeva lema

Naj bo $\mathcal{F}$ taka družina podmnožic n-elementne množice, da noben njen element ni vsebovan v kakem drugem elementu iz $\mathcal{F}$. Potem ima družina $\mathcal{F}$ največ

$$\binom{n}{\lfloor n/2 \rfloor} \text{ elementov.}$$

Bos-Chaumova shema za enkratni podpis

$\mathcal{P} = \{0,1\}^{k \in \mathbb{N}}$, $n \in \mathbb{N}$ tak, da je $2^k \leq \binom{2n}{n}$.
B je množica z $2n$ elementi in

$$\phi : \{0,1\}^k \to \mathcal{B}$$

injekcija, kjer je $\mathcal{B}$ množica n-teric iz B.

Naj bo $f : Y \to Z$ enosmerna funkcija.

Naključno izberemo vektor $\mathbf{y}= (y_i) \in Y^{2n}$.

Naj bo ključ K tajni vektor $\mathbf{y}$ in javni vektor $(f(y_i))$.

$$\mathrm{sig}_K(x_1, \ldots, x_k) = \{y_j \,|\, j \in \phi(x_1, \ldots, x_k)\}.$$

in

$$\mathrm{ver}_K(x_1, \ldots, x_k, a_1, \ldots, a_n) = \mathrm{true}$$
$$\Updownarrow$$
$$\{f(a_i) \,|\, 1 \leq i \leq n\} = \{z_j \,|\, j \in \phi(x_1, \ldots, x_k)\}.$$

Uporabili smo $2^k \leq \binom{2n}{n}$. Ocenimo binomski koeficient in dobimo

$$\binom{2n}{n} = \frac{(2n)!}{(n!)^2}$$

oziroma z uporabo Stirlingove formule $2^{2n}/\sqrt{(\pi n)}$.

Od tod dobimo

$$k \leq 2n - \frac{\log_2(n\pi)}{2}.$$

Asimptotično je torej n blizu $k/2$, zato smo dobili 50% redukcijo dolžine podpisa.

Slepi podpis

Želimo, da nam kdo podpiše dokument, hkrati pa nočemo, da bi podpisnik videl njegovo vsebino (npr. notarji, banke pri elektronskem denarju).

Algoritem (Chaum): Anita želi od Bojana podpis dokumenta x, $1 \leq x \leq n-1$, pri čemer je (n, e) Bojanov javni ključ za algoritem RSA, d pa zasebni ključ.

1. *Anita izbere takšno skrito naključno število k, da velja $0 \leq k \leq n-1$ in $D(n,k) = 1$.*

   Nato zastre dokument, tj. izračuna

   $$m = xk^e \bmod n,$$

   in ga pošlje Bojanu.
2. *Bojan podpiše zastrti dokument*

   $$s = m^d \bmod n.$$

3. *Anita odstre podpisani dokument*

   $$y = k^{-1}s \bmod n.$$

Podpisi brez možnosti zanikanja

Podpisa ni mogoče preveriti brez sodelovanja podpisnika, podpisnik pa tudi ne more zanikati, da bi že podpisani dokument res podpisal

(razen če odkloni sodelovanje pri podpisu, kar pa lahko pojmujemo kot priznanje, da je podpis v resnici ponarejen).

Primer algoritma (Chaum-van Antwerpen):

Naj bosta q in $p = 2q + 1$ praštevili, $\alpha \in \mathbb{Z}_p^*$ element reda q, $1 \leq a \leq q-1$ in $\beta = \alpha^a \bmod p$.

Grupa G je multiplikativna podgrupa reda q grupe $\mathbb{Z}_p^*$ (G sestavljajo kvadratični ostanki po modulo p).

Naj bo $\mathcal{P} = \mathcal{A} = G$ in

$$\mathcal{K} = \{(p, \alpha, a, \beta) : \beta = \alpha^a \bmod p\}.$$

Števila p, α in β so javna, vrednost a pa je skrita.

Podpisovanje (Bojan podpiše dokument $x \in G$):

$$y = \mathrm{sig}_K(x) = x^a \bmod p.$$

Preverjanje podpisa:

1. Anita izbere naključni števili $e_1, e_2 \in \mathbb{Z}_q^*$. Nato izračuna $c = y^{e_1}\beta^{e_2} \bmod p$ in ga pošlje Bojanu.
2. Bojan izračuna $d = c^{a^{-1} \bmod q} \bmod p$ in ga vrne Aniti.
3. Anita sprejme podpis kot veljaven, če je

$$d = x^{e_1}\alpha^{e_2} \bmod p.$$

Izrek. *Če je* $y \not\equiv x^a \pmod p$, *potem bo Anita*

sprejela y *za veljaven podpis čistopisa* x

z verjetnostjo $1/q$.

Poleg algoritmov za podpisovanje in preverjanje obstaja še algoritem (*disavowal protocol*), s katerim lahko podpisnik dokaže, da je ponarejen podpis res ponarejeni, hkrati pa ne more zanikati, da pravega podpisa ni napravil sam.

Primeri podpisov brez možnosti zanikanja

- *Entrusted undeniable signature*: *disavowal* protokol lahko izvede le za to določena ustanova, npr. sodišče.
- *Designated confirmer signature*: ob podpisu sami določimo, kdo bo namesto nas sodeloval pri preverjanjih podpisov. Podpišemo lahko še vedno le mi.
- *Convertible undeniable signature*: shema vsebuje skrito število. Do razkritja tega števila mora pri preverjanju podpisa sodelovati podpisnik.

  Po razkritju lahko kdorkoli preveri podpis sam

  (kot pri običajnem digitalnem podpisu).

Skupinski podpisi

Lastnosti:

- Dokumente lahko podpisujejo le člani določene skupine.
- Kdorkoli lahko preveri, da je dokument podpisal nekdo iz omenjene skupine, vendar ne more ugotoviti, kdo je to bil.
- V primeru spora je možno podpis “odpreti” in identificirati podpisnika.

Fail-stop podpisi

Če bi ponarejevalec z metodo grobe sile našel skriti ključ, bi lahko v večini sistemov za digitalne podpise podpis ponaredil. Fail-stop sistemi takšno možnost onemogočijo tako, da vsakemu javnemu ključu priredijo več skritih ključev.

Algoritem (van Heyst - Pedersen)

Generiranje ključa se razdeli med Anito in TTP (*Trusted Third Party*).

TTP izbere praštevili q in $p = 2q + 1$ (diskretni algoritem je težko izračunljiv), element $\alpha \in \mathbb{Z}_p^*$ reda q ter skrito naključno število a_0, $1 \leq a_0 \leq q - 1$ in izračuna $\beta \equiv \alpha^{a_0} \pmod{p}$. Nato Anita pošlje četverko (p, q, α, β) in izbere skrita naključna števila $a_1, a_2, b_1, b_2 \in \mathbb{Z}_q$, ki predstavljajo njen skriti ključ, ter določi svoj javni ključ $(\gamma_1, \gamma_2, p, q, \alpha, \beta)$, kjer je

$$\gamma_1 = \alpha^{a_1}\ \beta^{a_2} \pmod{p} \text{ in } \gamma_2 = \alpha^{b_1}\ \beta^{b_2} \pmod{p}.$$

Podpisovanje: $y = \text{sig}_K(x) = (y_1, y_2)$, kjer je

$$y_1 \equiv a_1 + x\ b_1 \pmod{q}$$

in

$$y_2 \equiv a_2 + x\ b_2 \pmod{q}.$$

Preverjanje podpisa:

$$\text{ver}_K(x, y_1, y_2) = \text{true} \iff \gamma_1 \gamma_2^x \equiv \alpha^{y_1} \beta^{y_2} \pmod{p}.$$

Opombe:

1. Natanko q^2 četverk (a_1', a_2', b_1', b_2'), kjer so elementi iz $\mathbb{Z}_q$, da enaki vrednosti (γ_1, γ_2) v javnem ključu.
2. Teh q^2 četverk da pri istem dokumentu x q različnih podpisov.
3. Naj bo Q_1 množica q četverk, ki da pri x enak podpis. Potem da ta množica pri drugem dokumentu q različnih podpisov.

Varnost sistema

Recimo, da želi nekdo ponarediti Anitin podpis za sporočilo x'.

1. Če ponarejevalec pozna le skriti ključ, ki pripada javnemu, je verjenost $1/q$, da je njegov podpis enak Anitinemu.
2. Ponarejevalec ima dostop do drugega sporočila x in Anitinega podpisa (y_1, y_2). Po tretji opombi je verjetnost spet $1/q$.

7. Zgoščevalne funkcije

- zgoščevalne funkcije brez trčenj
- verjetnost trčenja
- napad s pomočjo paradoksa rojstnih dnevov
- zgoščevalna funkcija z diskretnim logaritmom

Shema DSS (brez uporabe zgoščevalnih funkcij) podvoji dolžino podpisanega sporočila.

Resnejši problem nastane, ker je mogoče preurejati dele podpisanega sporočila ali pa nekatere celo izpustiti/dodati.

Celovitost podatkov ne more biti zagotovljena
izključno s podpisovanjem majhnih delov dokumenta,
zato vpeljemo **zgoščevalne funkcije** (angl. Hash Functions), ki poljubno dolgemu sporočilu priredijo kratko zaporedje bitov, ki jih potem podpišemo.

Zgoščevalne funkcije brez trčenj

(angl. Collision-free Hash Functions)

Naj bo (x, y) podpisano sporočilo, kjer je

$$y = \mathrm{sig}_K(h(x)).$$

Preprost napad: izračunamo $z = h(x)$ in nato poiščemo tak od x različen x', da je $h(x') = h(x)$.

Def: Naj bo x sporočilo. Za zgoščevalno funkcijo h pravimo, da je **šibko brez trčenj** (angl. weakly collision-free), če v doglednem času ni možno najti (izračunati) tak od x različen x', da je $h(x) = h(x')$.

Še en napad: poiščemo taka x in x', da je $x \neq x'$ in $h(x') = h(x)$ ter prisilimo Bojana, da podpiše x. Potem je (x', y) poneverjen podpis.

Def: Za zgoščevalno funkcijo h pravimo, da je **krepko brez trčenj** (angl. strongly collision-free), če v doglednem času ni možno najti (izračunati) taka x in x', da je $x \neq x'$ in $h(x) = h(x')$.

Pa še en napad: recimo, da nam je uspelo ponarediti podpis naključnega števila z, nato pa poiščemo tak x, da je $z = h(x)$.

Ta napad preprečimo z enosmernimi funkcijami.

Dokazali bomo, da so funkcije brez trčenj enosmerne. To sledi iz trditve, da je možno algoritem za računanje obrata zgoščevalne funkcije uporabiti kot podprogram Las Vegas probabilističnega algoritma, ki išče trčenja.

Izrek: *Naj bo $h : X \longrightarrow Z$ zgoščevalna funkcija,*
$|X| < \infty$ in $|X| \geq 2|Z|$ ter naj bo **A** *algoritem*
za računanje obrata zgoščevalne funkcije.
Potem obstaja Las Vegas probabilistični algoritem,
ki najde trčenja z verjetnostjo vsaj $1/2$.

Dokaz: Naj bo **B** naslednji algoritem.

1. Izberi naključen element $x \in X$,
2. izračunaj $z := h(x)$,
3. izračunaj $x_1 := \mathbf{A}(z)$,
4. **if** $x_1 \neq x$ **then** x_1 in x trčita glede na h (uspeh)
   else QUIT(neuspeh).

Izračunajmo verjetnost za uspeh. Najprej definiramo ekvivalenčno relacijo

$$x \sim x' \iff h(x) = h(x').$$

Naj bo C množica ekvivalenčnih razredov,

potem je $|C| \leq |Z|$. Velja tudi: $|Z| \leq |X|/2$.

$$P(\text{uspeh}) = \frac{1}{|X|} \sum_{x \in X} \frac{|[x]| - 1}{|[x]|} = \frac{1}{|X|} \sum_{c \in C} \sum_{x \in c} \frac{|c| - 1}{|c|}$$

$$= \frac{1}{|X|} \sum_{c \in C} (|c| - 1) \geq \frac{|X| - |Z|}{|X|} \geq \frac{1}{2} . \blacksquare$$

Verjetnost trčenja

Naj bo $h : X \longrightarrow Z$ zgoščevalna funkcija,

$$s_z = |h^{-1}(z)|$$

in

$$N = |\{\{x_1, x_2\} \,|\, h(x_1) = h(x_2)\}|,$$

tj. N je število neurejenih parov, ki trčijo pri funkciji h.

Pokazali bomo, da obstaja od nič različna spodnja meja za verjetnost P, da je $h(x_1) = h(x_2)$, kjer sta x_1 in x_2 naključna (ne nujno različna) elementa iz X.

1. $\sum_{z\in Z} s_z = |X|$, tj. povprečje s_z-ov je $\overline{s} = |X|/|Z|$.
2. $N = \sum_{z\in Z} \binom{s_z}{2} = \frac{1}{2}\sum_{z\in Z} s_z^2 - \frac{|X|}{2}$.
3. $\sum_{z\in Z} (s_z - \overline{s})^2 = 2N + |X| - |X|^2/|Z|$.
4. $N \geq \frac{|X|}{2}\left(\frac{|X|}{|Z|} - 1\right)$, pri čemer velja enakost

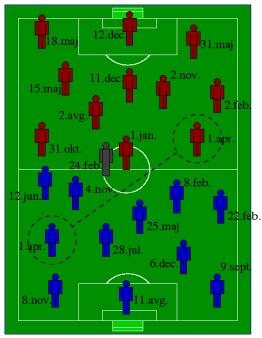
   natanko tedaj, ko je $s_z = |X|/|Z|$ za vsak $z \in Z$.
5. $P \geq 1/|Z|$, pri čemer velja enakost natanko tedaj, ko je $s_z = |X|/|Z|$ za vsak $z \in Z$.

Kakšno naključje!!! Mar res?

Na nogometni tekmi sta na igrišču dve enajsterici in sodnik, skupaj **23 oseb**.

Kakšna je verjetnost, da imata **dve osebi** isti rojstni dan?

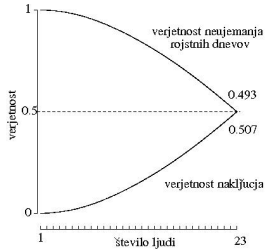
Ali je ta verjetnost lahko večja od 0.5?



Ko vstopi v sobo k-ta oseba, je verjetnost, da je vseh k rojstnih dnevov različnih enaka:

$$\frac{365}{365} \times \frac{364}{365} \times \frac{363}{365} \times \cdots \times \frac{365-k+1}{365}$$

$$= \begin{cases} 0.493, \\ 0.507, \end{cases}$$



V poljubni skupini 23-ih ljudi je verjetnost, da imata vsaj dva skupni rojstni dan $> 1/2$.

Čeprav je 23 majhno število, je med 23 osebami 253 različnih parov. To število je veliko bolj povezano z iskano verjetnostjo.

Testirajte to na zabavah z več kot 23 osebami.

Organizirajte stave in dolgoročno boste gotovo na boljšem, na velikih zabavah pa boste zlahka zmagovali.

Napad s pomočjo paradoksa rojstnih dnevov

(angl. Birthday Attack)

To seveda ni paradoks, a vseeno ponavadi zavede naš občutek.

Ocenimo še splošno verjetnost.

Mečemo k žogic v n posod in gledamo, ali sta v kakšni posodi vsaj dve žogici.

Poiščimo spodnjo mejo za verjetnost zgoraj opisanega dogodka.

Privzeli bomo, da je $|h^{-1}(x)| \approx m/n$, kjer je $n = |Z|$ in $m = |X|$ (v primeru, da velikosti praslik niso enake se verjetnost le še poveča).

$$\left(1 - \frac{1}{n}\right)\left(1 - \frac{2}{n}\right)\cdots\left(1 - \frac{k-1}{n}\right) = \prod_{i=1}^{k-1}\left(1 - \frac{i}{n}\right)$$

Iz Taylorjeve vrste

$$e^{-x} = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \cdots$$

ocenimo $1 - x \approx e^{-x}$ in dobimo

$$\prod_{i=1}^{k-1}\left(1 - \frac{i}{n}\right) \approx \prod_{i=1}^{k-1} e^{\frac{-i}{n}} = e^{\frac{-k(k-1)}{2n}}.$$

Torej je verjetnost trčenja

$$1 - e^{\frac{-k(k-1)}{2n}}.$$

Potem velja

$$e^{\frac{-k(k-1)}{2n}} \approx 1 - \varepsilon$$

oziroma

$$\frac{-k(k-1)}{2n} \approx \log(1 - \varepsilon)$$

oziroma

$$k^2 - k \approx 2n \log \frac{1}{1 - \varepsilon}$$

in če ignoriramo $-k$, dobimo končno

$$\boxed{k \approx \sqrt{2n \log \frac{1}{1 - \varepsilon}}\,.}$$

Za $\varepsilon = 0.5$ je

$$k \approx 1.17\sqrt{n},$$

kar pomeni, da, če zgostimo nekaj več kot $\sqrt{n}$ elementov, je bolj verjetno, da pride do trčenja kot da ne pride do trčenja.

V splošnem je k proporcionalen z $\sqrt{n}$.

Napad s pomočjo paradoksa rojstnih dnevov s tem določi spodnjo mejo za velikost zaloge vrednosti zgoščevalne funkcije.

40-bitna zgostitev ne bi bila varna, saj bi prišli do trčenja z nekaj več kot 2^{20} (se pravi milijon) naključnimi zgostitvami z verjetnostjo vsaj $1/2$.

V praksi je priporočena najmanj 128-bitna zgostitev in shema DSS z 160-imi biti to vsekakor upošteva.

Zgoščevalna funkcija z diskretnim logaritmom

Varnost Chaum, Van Heijst in Pfitzmannove zgoščevalne funkcije je zasnovana na varnosti diskretnega logaritma.

Ni dovolj hitra, da bi jo uporabljali v praksi, je pa zato vsaj primerna za študij varnosti.

Naj bosta p in $q = (p-1)/2$ veliki praštevili,

α in β pa dva primitivna elementa v $\mathbb{Z}_p$,

za katera je vrednost $\log_\alpha \beta$ zasebna.

Zgoščevalno funkcijo

$$h : \{0, \ldots, q-1\} \times \{0, \ldots, q-1\} \longrightarrow \mathbb{Z}_p \backslash \{0\}$$

definirajmo z

$$h(x_1, x_2) = \alpha^{x_1} \beta^{x_2} \bmod p.$$

Pokazali bomo, da je ta funkcija **krepko brez trčenj** (strongly collision-free).

Predpostavimo obratno: za $(x_1, x_2) \neq (x_3, x_4)$ velja

$$h(x_1, x_2) = h(x_3, x_4)$$

oziroma

$$\alpha^{x_1} \beta^{x_2} \equiv \alpha^{x_3} \beta^{x_4} \pmod{p}$$

ali

$$\alpha^{x_1 - x_3} \equiv \beta^{x_4 - x_2} \pmod{p}.$$

Če je $d = D(x_4 - x_2, p-1)$, potem imamo zaradi $p - 1 = 2q$ natanko štiri možnost za d:

$$\{1, 2, q, p-1\}.$$

Če je $d = 1$, definiramo

$$y = (x_4 - x_2)^{-1} \mod (p-1)$$

in dobimo

$$\beta \equiv \beta^{(x_4-x_2)y} \equiv \alpha^{(x_1-x_3)y} \pmod{p},$$

iz česar znamo izračunati diskretni logaritem

$$\log_\alpha \beta \equiv (x_1 - x_3)(x_4 - x_2)^{-1} \mod (p-1).$$

Če je $d = 2$, je $d = D(x_4 - x_2, q) = 1$, tako da lahko definiramo

$$y = (x_4 - x_2)^{-1} \mod q$$

in dobimo za $(x_4 - x_2)y = kq + 1$, kjer je $k \in \mathbb{Z}$,

$$\beta^{(x_4-x_2)y} \equiv \beta^{kq+1} \equiv (-1)^k \beta \equiv \pm\beta \pmod{p}$$

zaradi $\beta^q \equiv -1 \pmod{p}$.

Torej imamo

$$\pm\beta \equiv \beta^{(x_4-x_2)y} \equiv \alpha^{(x_1-x_3)y} \pmod{p},$$

od koder znamo izračunati diskretni logaritem

$$\log_\alpha \beta \equiv (x_1 - x_3)y \bmod (p-1)$$

ali pa

$$\log_\alpha \beta \equiv (x_1 - x_3)y + q \mod (p-1).$$

Primer $d = q$ ni možen, saj iz

$$0 \le x_2 \le q-1 \quad \text{in} \quad 0 \le x_4 \le q-1$$

sledi

$$-(q-1) \le x_4 - x_2 \le q-1.$$

Končno si poglejmo še primer $d = p - 1$, kar se lahko zgodi le za $x_2 = x_4$. Potem velja

$$\alpha^{x_1} \equiv \alpha^{x_3} \pmod{p}$$

oziroma $x_1 = x_3$ in $(x_1, x_2) = (x_3, x_4)$. Protislovje! ■

Razširitev zgoščevalne funkcije

Doslej smo študirali zgoščevalne funkcije s končno domeno.

Sedaj pa pokažimo, kako lahko razširimo zgoščevalne funkcije, ki so krepko brez trčenj in imajo končno domeno, do zgoščevalnih funkcij, ki so krepko brez trčenj in imajo neskončno domeno.

Tako bomo lahko podpisovali sporočila poljubne dolžine.

Naj bo h^* zgoščevalna funkcija za katero je $|X| = \infty$.

Naj bo zgoščevalna funkcija $h : (\mathbb{Z}_2)^m \longrightarrow (\mathbb{Z}_2)^t$, $m \ge t+1$ krepko brez trčenj.

Potem bomo za $X = \cup_{i=m}^{\infty} (\mathbb{Z}_2)^i$ definirali zgoščevalno funkcijo

$$h^* : X \longrightarrow (\mathbb{Z}_2)^t,$$

ki bo tudi krepko brez trčenj.

Elementi množice X so zaporedja bitov, $|x|$, $x \in X$, pa naj predstavlja dolžino elementa x, tj. število bitov x-a. Z $x \,||\, y$ označimo spetje zaporedij x in y.

1.primer: $m \geq t+2$. Naj bo $|x| = n > m$ in x spetje $x_1 \,||\, x_2 \,||\, \ldots \,||\, x_k$, kjer je

$$|x_1| = |x_2| = \cdots = |x_{k-1}| = m - t - 1$$

in $|x_k| = m - t - 1 - d$, pri čemer je $0 \leq d \leq m - t - 2$. Torej je

$$k = \left\lceil \frac{n}{m-t-1} \right\rceil.$$

Funkcijo $h^*(x)$ definiramo z naslednjim algoritmom:

1. **for** $i = 1$ **to** $k-1$ **do** $y_i = x_i$
2. $y_k = x_k \,||\, 0^d$
3. naj bo y_{k+1} število d v dvojiškem sistemu
4. $g_1 = h(0^{t+1} \,||\, y_1)$
5. **for** $i = 1$ **to** k **do** $g_{i+1} = h(g_i \,||\, 1 \,||\, y_{i+1})$
6. $h^*(x) = g_{k+1}$

Spetje $y_1 \,||\, y_2 \,||\, \ldots \,||\, y_{k+1}$ smo dobili tako, da smo x_k-ju na desni pripeli d ničel, zaporedju y_{k+1} pa smo pripeli ničle na levi, tako da je $|y_{k+1}| = m - t - 1$.

Izrek: *Naj bo* $h : (\mathbb{Z}_2)^m \longrightarrow (\mathbb{Z}_2)^t$, $m \geq t+2$

zgoščevalna funkcija krepko brez trčenj.

Potem je zgoraj def. zgoščevalna funkcija

$h^* : X \longrightarrow (\mathbb{Z}_2)^t$ *tudi krepko brez trčenj.*

Dokaz: Predpostavimo, da smo našli $x \neq x'$ tako, da je $h^*(x) = h^*(x')$ in pokažimo, da lahko poiščemo v polinomskem času trčenje za h. Vzamemo $|x| \geq |x'|$.

Naj bo $y(x) = y_1 || \ldots || y_{k+1}$ in $y(x') = y'_1 || \ldots || y'_{j+1}$, kjer sta x in x' dopolnjena z d ničlami po 2. koraku in so $g_1, \ldots, g_{k+1}$ in $g'_1, \ldots, g'_{j+1}$ zaporedoma vrednosti, ki jih izračunamo v korakih 4 in 5.

Če je $|x| \not\equiv |x'| \pmod{m-t-1}$, potem je $d \neq d'$ in od tod $y_{k+1} \neq y'_{j+1}$, torej dobimo trčenje iz

$$h(g_k||1||y_{k+1}) = g_{k+1} = h^*(x) = h^*(x') = g'_{j+1} = h(g'_j||1||y'_{j+1}).$$

Zato smemo sedaj privzeti, da $m - t - 1$ deli $|x| - |x'|$. Iz $y_{k+1} = y'_{j+1}$, tako kot v prejšnjem primeru, sledi

$$h(g_k||1||y_{k+1}) = h(g'_j||1||y'_{j+1}).$$

Če je $g_k \neq g'_j$, smo našli trčenje, v nasprotnem primeru pa je

$$h(g_{k-1}||1||y_k) = g_k = g'_j = h(g'_{j-1}||1||y'_j).$$

Če na ta način s postopnim vračanjem ne pridemo do trčenja, dobimo na koncu

$$h(0^{t+1}||y_1) = g_1 = g'_{j-k} = \begin{cases} h(0^{t+1}||y'_1), & \text{če je } |x| = |x'| \\ h(g'_{j-k}||1||y'_1), & \text{sicer} \end{cases}$$

Za $|x| = |x'|$ oziroma $k = j$ nam da $y_1 \neq y'_1$ trčenje, sicer pa je $y_i = y'_i$ za $1 \leq i \leq k+1$. Od tod $y(x) = y(x')$, toda potem je $x = x'$, saj je preslikava $x \mapsto y(x)$ injekcija. Dobili smo protislovje s predpostavko, da je $x \neq x'$.

Končno v primeru, ko je $m - t - 1$ deli $|x| - |x'| \neq 0$ dobimo trčenje, ker je $(t+1)$-vi bit v spetju $0^{t+1}||y_1$ enak 0, $(t+1)$-vi bit v spetju $g'_{j-k} \,||\, 1 \,||\, y'_1$ pa 1. ■

2.primer: $m = t+1$. Naj bo $|x| = n > m$ in definirajmo funkcijo f z $f(0) = 0$ in $f(1) = 01$.

Zgoščevalno funkcijo $h^*(x)$ definiramo z algoritmom:

1. $y = y_1 y_2 \ldots y_k := 11||f(x_1)||f(x_2)|| \cdots ||f(x_n)$
2. $g_1 = h(0^t \,||\, y_1)$
3. **for** $i = 1$ **to** $k-1$ **do** $g_{i+1} = h(g_i \,||\, y_{i+1})$
4. $h^*(x) = g_k$

Funkcija $x \mapsto y = y(x)$ iz prvega koraka je injekcija.

Izrek: *Naj bo* $h : (\mathbb{Z}_2)^{t+1} \longrightarrow (\mathbb{Z}_2)^t$

zgoščevalna funkcija krepko brez trčenj.

Potem je zgoraj def. zgoščevalna funkcija

$h^* : X \longrightarrow (\mathbb{Z}_2)^t$ *tudi krepko brez trčenj.*

Dokaz: Predpostavimo, da smo našli $x \neq x'$ tako, da je $h^*(x) = h^*(x')$ in naj bo $y(x) = y_1||\ldots||y_{k+1}$ in $y(x') = y'_1||\ldots||y'_{j+1}$, kjer sta x in x' dopolnjena z d ničlami po 2. koraku.

Če je $k = j$, dobimo (kot pri prejšnjem dokazu) bodisi trčenje za zgoščevalno funkcijo h bodisi $y = y'$. Slednje nam da $x = x'$, kar pa je protislovje!

Sedaj pa privzemimo, da je $k \neq j$ oziroma kar $j > k$. Če ne pride do trčenja, dobimo naslednje zaporedje enakosti:

$$y_k = y'_j, \;\; y_{k-1} = y'_{j-1}, \ldots, y_1 = y'_{j-k+1},$$

kar pa ni možno, saj za $x \neq x'$ ter poljubno zaporedje z velja $y(x) \neq z||y(x')$, kajti zaporedni enici se pojavita izključno na začetku zaporedja $y(x)$.

Od tod zaključimo, da je h^* krepko brez trčenj. ■

Za računanje funkcije h^* smo uporabili funkcijo h kvečjemu

$$\left(1 + \left\lceil \frac{n}{m-t-1} \right\rceil\right) - \text{krat} \quad \text{za} \quad m \geq t+2$$

in

$$(2n+2) - \text{krat} \quad \text{za} \quad m = t+1.$$

Zgoščevalne funkcije iz kriptosistemov

Naj bo $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ računsko varen kriptosistem in naj bo

$$\mathcal{P} = \mathcal{C} = \mathcal{K} = \mathbb{Z}_n,$$

kjer je $n \geq 128$, da bi preprečili napad z rojstnim dnevom.

Ta pogoj izključi **DES** (pa tudi DES-ov čistopis ni tako dolg kot DES-ov ključ).

Naj bo dano zaporedje

$$x_1 \,||\, x_2 \,||\, \ldots \,||\, x_k, \quad \text{kjer je } x_i \in (\mathbb{Z}_2)^n, \;\; 1 \leq i \leq k.$$

Če število bitov v zaporedju ne bi bilo večkratnik števila n, bi lahko dodali nekaj ničel...

Začnemo z neko začetno vrednostjo $g_0 = \text{IV}$ (initial value) in nato konstruiramo zaporedje

$$g_i = f(x_i, g_{i-1}),$$

kjer je f šifrirna funkcija izbranega kriptosistema. Potem je $h(x) = g_k$.

Definiranih je bilo veliko takih funkcij in mnoge med njimi so razbili (tj. dokazali, da niso varne), ne glede na to, ali je ustrezna šifra varna ali ne.

Naslednje štiri variacije pa zaenkrat izgledajo varne:

$$g_i = e_{g_{i-1}}(x_i) \oplus x_i,$$

$$g_i = e_{g_{i-1}}(x_i) \oplus x_i \oplus g_{i-1},$$

$$g_i = e_{g_{i-1}}(x_i \oplus g_{i-1}) \oplus x_i,$$

$$g_i = e_{g_{i-1}}(x_i \oplus g_{i-1}) \oplus x_i \oplus g_{i-1}.$$

Zgoščevalna funkcija MD4

Preglejmo nekaj hitrih zgoščevalnih funkcij.

MD4 je predlagal Rivest leta 1990, njeno izboljšano verzijo **MD5** pa leta 1991.

Funkcija **Secure Hash Standard** (**SHS**) iz leta 1992/93 je bolj kompli-cirana, a je zasnovana na istih principih. Njeno "tehnično napako" pa so odpravili šele leta 1994 (**SHA-1**).

Iz danega zaporedja bitov x najprej sestavimo zaporedje

$$M \;=\; M[0]\,M[1]\,\ldots\,M[N-1],$$

kjer je $M[i]$ 32-bitna beseda in je $N \equiv 0 \bmod 16$.

1. $d = (447 - |x|) \bmod 512$,
2. naj bo j binarna reprezentacija števila $x \bmod 2^{64}$, pri čemer je $|j| = 64$,
3. $M = x \,||\, 1 \,||\, 0^d \,||\, j$.

1. $A = 67452301$ (hex), $B = efcdab89$ (hex),
   $C = 98badcfe$ (hex), $D = 10325476$ (hex)
2. **for** $i = 1$ **to** $N/16 - 1$ **do**
3. **for** $j = 0$ **to** 15 **do**
4. $X[j] = M[16i + j]$.
5. $AA{=}A, \ldots, DD{=}D$.
6. 1. krog
7. 2. krog
8. 3. krog
9. $A{=}A{+}AA, \ldots, D{=}D{+}DD$.

Osnovne operacije:

$X \wedge Y$	po bitih
$X \vee Y$	po bitih
$X \oplus Y$	XOR po bitih
$\neg X$	negacija
$X + Y$	seštevanje po modulu 2^{32}
$X \lll s$	ciklični pomik v levo za s mest

V *big-endian* arhitekturi (kot npr. Sun SPARC postaja) predstavimo število na naslednji način

$$a_1 2^{24} + a_2 2^{16} + a_3 2^8 + a_4,$$

v *little-endian* arhitekturi (kot npr. Intel 80xxx), ki jo je privzela funkcija MD4 pa z

$$a_4 2^{24} + a_3 2^{16} + a_2 2^8 + a_1.$$

V 1., 2. in 3. krogu funkcije **MD4** uporabimo zaporedoma funkcije f, g, in h, definirane spodaj.

$$f(X,Y,Z) \;=\; (X \wedge Y) \vee ((\neg X) \wedge Z)$$

$$g(X,Y,Z) \;=\; (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z)$$

$$h(X,Y,Z) \;=\; X \oplus Y \oplus Z$$

1. krog

1. $A = (A + f(B, C, D) + X[0]) \lll 3$
2. $D = (D + f(A, B, C) + X[1]) \lll 7$
3. $C = (C + f(D, A, B) + X[2]) \lll 11$
4. $B = (B + f(C, D, A) + X[3]) \lll 19$

5. $A = (A + f(B, C, D) + X[4]) \lll 3$
6. $D = (D + f(A, B, C) + X[5]) \lll 7$
7. $C = (C + f(D, A, B) + X[6]) \lll 11$
8. $B = (B + f(C, D, A) + X[7]) \lll 19$

9. $A = (A + f(B, C, D) + X[8]) \lll 3$
10. $D = (D + f(A, B, C) + X[9]) \lll 7$
11. $C = (C + f(D, A, B) + X[10]) \lll 11$
12. $B = (B + f(C, D, A) + X[11]) \lll 19$

13. $A = (A + f(B, C, D) + X[12]) \lll 3$
14. $D = (D + f(A, B, C) + X[13]) \lll 7$
15. $C = (C + f(D, A, B) + X[14]) \lll 11$
16. $B = (B + f(C, D, A) + X[15]) \lll 19$

2. krog

1. $A = (A + g(B, C, D) + X[0] + 5A827999) \lll 3$
2. $D = (D + g(A, B, C) + X[4] + 5A827999) \lll 5$
3. $C = (C + g(D, A, B) + X[8] + 5A827999) \lll 9$
4. $B = (B + g(C, D, A) + X[12] + 5A827999) \lll 13$

5. $A = (A + g(B, C, D) + X[1] + 5A827999) \lll 3$
6. $D = (D + g(A, B, C) + X[5] + 5A827999) \lll 5$
7. $C = (C + g(D, A, B) + X[9] + 5A827999) \lll 9$
8. $B = (B + g(C, D, A) + X[13] + 5A827999) \lll 13$

9. $A = (A + g(B, C, D) + X[2] + 5A827999) \lll 3$
10. $D = (D + g(A, B, C) + X[6] + 5A827999) \lll 5$
11. $C = (C + g(D, A, B) + X[10] + 5A827999) \lll 9$
12. $B = (B + g(C, D, A) + X[14] + 5A827999) \lll 13$

13. $A = (A + g(B, C, D) + X[3] + 5A827999) \lll 3$
14. $D = (D + g(A, B, C) + X[7] + 5A827999) \lll 5$
15. $C = (C + g(D, A, B) + X[11] + 5A827999) \lll 9$
16. $B = (B + g(C, D, A) + X[15] + 5A827999) \lll 13$

3. krog

1. $A = (A + h(B, C, D) + X[0] + 6ED9EBA1) \lll 3$
2. $D = (D + h(A, B, C) + X[8] + 6ED9EBA1) \lll 9$
3. $C = (C + h(D, A, B) + X[4] + 6ED9EBA1) \lll 11$
4. $B = (B + h(C, D, A) + X[12] + 6ED9EBA1) \lll 15$

5. $A = (A + h(B, C, D) + X[2] + 6ED9EBA1) \lll 3$
6. $D = (D + h(A, B, C) + X[10] + 6ED9EBA1) \lll 9$
7. $C = (C + h(D, A, B) + X[6] + 6ED9EBA1) \lll 11$
8. $B = (B + h(C, D, A) + X[14] + 6ED9EBA1) \lll 15$

9. $A = (A + h(B, C, D) + X[1] + 6ED9EBA1) \lll 3$
10. $D = (D + h(A, B, C) + X[9] + 6ED9EBA1) \lll 9$
11. $C = (C + h(D, A, B) + X[5] + 6ED9EBA1) \lll 11$
12. $B = (B + h(C, D, A) + X[13] + 6ED9EBA1) \lll 15$

13. $A = (A + h(B, C, D) + X[3] + 6ED9EBA1) \lll 3$
14. $D = (D + h(A, B, C) + X[11] + 6ED9EBA1) \lll 9$
15. $C = (C + h(D, A, B) + X[7] + 6ED9EBA1) \lll 11$
16. $B = (B + h(C, D, A) + X[15] + 6ED9EBA1) \lll 15$

Zgoščevalna funkcija **MD4** še ni bila razbita, vendar pa je ni težko razbiti, če bi opustili prvi ali pa zadnji krog.

Zato zgoščevalna funkcija **MD5** uporablja 5 krogov, a je 30% počasnejša (.9Mbytes/sec na SPARC-u).

Zgoščevalna funkcija **SHA** je še počasnejša (0.2Mbytes/sec na SPARC-u).
Opisali bomo le nekaj njenih modifikacij:

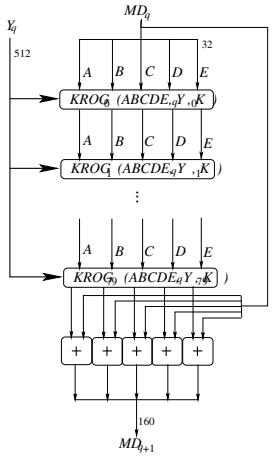
1. **SHS** privzame *big-endian* arhitekturo namesto *little-endian*.
2. **SHS** dobi 160-bitni rezultat (5 registrov).
3. **SHS** obdela 16 besed naenkrat, vendar jih najprej razširi v 80 besed, potem pa uporabi zaporedje 80-ih operacij na vsaki besedi.

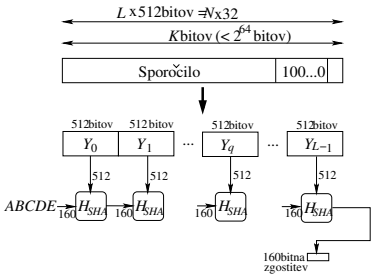
   $$X[j] = X[j-3] \oplus X[j-8] \oplus X[j-14] \oplus X[j-16]$$

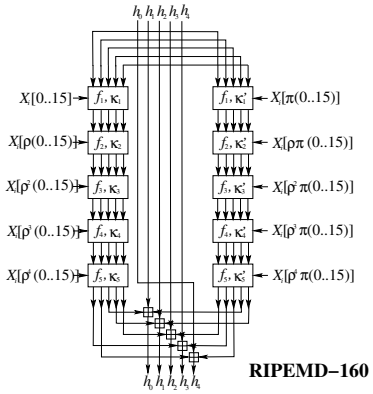
   za $16 \leq j \leq 79$.
4. **SHA-1** pa uporabi

   $$X[j] = X[j-3] \oplus X[j-8] \oplus X[j-14] \oplus X[j-16] \lll 1$$

   za $16 \leq j \leq 79$.







RIPEMD–160

HMAC

(Keyed-Hashing for Message Authentication)

Prednosti:

1. Kripto. zgoščevalne funkcije so v splošnem hitrejše v softwaru kot pa simetrične šifre (kot na primer DES).
2. Knjižnice zgoščevalnih funkcij so široko dostopne (medtem ko so bločne šifre, tudi kadar so uporabljene samo za MAC, omejene v smislu izvoznih dovoljenj).

Za design objectives v HMAC algoritmu in njegovo varnost glej:

M. Bellare, R. Canetti in H. Krawczyk, CRYPTO'96

(in `http://wwwcse.ucsd.edu/users/mihir`),

ki ga trenutno poskušajo vključiti v IETF
(Internet Engineering Task Force).

Časovne oznake/žigi (Timestamping)

Potrebujemo pričo o obstoju določenih podatkov ob določenem času, na primer na področju

- zaščite intelektualne lastnine
  (angl. intelectual property - IP), ali pa
- zanesljivega servisa za preprečevanje zanikanja

  (za dokaz, da je bil digitalni podpis generiran

  v času veljavnosti ustreznega javnega ključa).

Če želi Bojan imeti dokaz o obstoju podatkov x ob nekem določenem času, potem naredi naslednje:

1. najprej izračuna zgostitev $z = h(x)$,
2. nato še zgostitev spoja

$$z' = h(z \,||\, \text{javna\_informacija}),$$

3. rezultat podpiše $y = \text{sig}_K(z')$, in
4. naslednji dan v časopisu objavi podatke

$$(z, \text{javna\_informacija}, y).$$

Časovne oznake s TS

Časovne žige omogoča pooblaščena organizacija za podpise, (angl. **Time-stamper - TS**), ki je elektronski notar (angl. trusted timestamping service) oz. center zaupanja (TTP, angl. trusted third party).

Bojan najprej izračuna

$$z = h(x), \quad y = \mathrm{sig}_K(x)$$

in pošlje par (z, y) notarju TS,

ki doda še datum D in podpiše trojico (z, y, D).

Zgornji algoritem je varen le pod pogojem, če je notar nepodkupljiv.

Potencialno se TS sooči z enormno odgovornostjo, če so časi kompromitirani.

Na primer, uporabnik lahko zanika vse podpise, ki jih je opravil kdaj koli.

Morda lahko nastane hujša škoda kot kompromitacija CA-jevega zasebnega ključa, o katerem bomo govorili v naslednjem poglavju.

Sicer pa si pomagamo z naslednjim algoritmom:

1. TS najprej izračuna
$$L_n = (t_{n-1}, \mathrm{ID}_{n-1}, z_{n-1}, y_{n-1}, h(L_{n-1})),$$
2. nato še $C_n = (n, t_n, z_n, y_n, \mathrm{ID}_n, L_n)$,
3. rezultat podpiše $s_n = \mathrm{sig}_{\mathrm{TS}}(h(C_n)))$, ter
4. pošlje $(C_n, s_n, \mathrm{ID}_{n+1})$ osebi ID_n.

8. Upravljanje ključev

- **Distribucija ključev**
  (Blomova shema, Diffie-Hellmanova shema)
- **Certifikati**
  (avtentikacijska drevesa, certifikatna agencija, infrastruktura javnih ključev, proces certifikacije, modeli zaupanja)
- **Uskladitev ključev**
  (Kerberos, Diffie-Hellmanova shema, MTI protokoli, Giraultova shema)
- **Internetne aplikacije**
  (Internet, IPsec: Virtual Private Networks, Secure Sockets Layer, varna e-pošta)

Vprašanja

- Od kje dobimo ključe?
- Zakaj zaupamo ključem?
- Kako vemo čigav ključ imamo?
- Kako omejiti uporabo ključev?
- Kaj se zgodi, če je kompromitiran (izgubljen) zasebni ali tajni ključ? Kdo je odgovoren?
- Kako preklicati ključ?
- Kako lahko obnovimo ključ?
- Kako omogočimo servis preprečitve zanikanja?

Ta vprašanja veljajo tako za simetrične (tajne) ključe kakor tudi za javne in zasebne ključe.

Upravljanje ključev je množica tehnik in postopkov, ki podpirajo dogovor in vzdrževanje relacij ključev med pooblaščenimi strankami/sogovorniki.

Infrastruktura javnih ključev (PKI):
podporni servisi (tehnološki, pravni, komercialni, itd.), ki so potrebni, da lahko tehnologijo javnih ključev uporabimo za večje projekte.

Sistemi z javnimi ključi imajo prednost pred sistemi s tajnimi ključi, saj za izmenjavo tajnih ključev ne potrebujejo varnega kanala.

Večina sistemov z javnimi ključi (npr. RSA) je tudi do 100-krat počasnejša od simetričnih sistemov (npr. DES). Zato v praksi uporabljamo za šifriranje *daljših* besedil simetrične sisteme.

Obravnavali bomo več različnih protokolov za tajne ključe. Razlikovali bomo med *distribucijo ključev* in *uskladitvijo ključev*.

Sistem distribucije ključev je mehanizem, kjer na začetni stopnji verodostojna agencija generira in distribuira tajne podatke uporabnikom tako, da lahko vsak par uporabnikov kasneje izračuna ključ, ki je nepoznan ostalim.

Uskladitev ključev označuje protokol, kjer dva ali več uporabnikov sestavjo skupen tajni ključ, s komunikacijo po javnem kanalu. Vrednost ključa je določena s funkcijo vhodnih podatkov.

Obstaja potreba po zaščiti pred potencialnimi nasprotniki, tako pasivnimi kot tudi aktivnimi.

Pasivni sovražnik je osredotočen na prisluškovanje sporočilom, ki se pretakajo po kanalu.
Več nevšečnosti nam lahko naredi *aktivni* sovražnik:

- spreminjanje sporočil,
- shranjevanje sporočil za kasnejšo uporabo,
- maskiranje v uporabnika omrežja.

Cilj *aktivnega* sovražnika uporabnikov U in V je lahko:

- prelisičiti U in V tako, da sprejmeta neveljaven ključ kot veljaven,
- prepričati U in V, da sta si izmenjala ključ, čeprav si ga v resnici nista.

Center zaupanja

V omrežju, ki ni varno, se v nekaterih shemah pojavi agencija, ki je odgovorna za

- potrjevanje identitete,
- izbiro in prenos ključev
- itd.

Rekli ji bomo **center zaupanja** ali **verodostojna agencija** (angl. Trusted Authority – TA ali Trusted Third Party – TTP).

Uporabljali bomo oznako **TA**.

Distribucija ključev

- "Point-to-point" distribucija po varnem kanalu:
  - zaupni kurir,
  - enkratna registracija uporabnikov,
  - prenos po telefonu.
- Neposreden dostop do overjene javne datoteke:
  - avtentična drevesa,
  - digitalno podpisana datoteka.
- Uporaba "on-line" zaupnih strežnikov,
- "Off-line" certifikatna agencija (CA).

Point-to-point

Predpostavimo, da imamo

- omrežje z n uporabniki,
- agencija TA generira in preda enolično določen ključ vsakemu paru uporabnikov omrežja.

Če imamo varen kanal med TA in vsakim uporabnikom omrežja, potem dobi vsak posameznik $n-1$ ključev,
zahtevnost problema pa je vsaj $\mathcal{O}(n^2)$.

Ta rešitev ni praktična celo za relativno majhne n.

Želimo si boljšo rešitev, npr. z zahtevnostjo $\mathcal{O}(1)$.

Blomova shema

Naj bo javno p praštevilo večje od danega $n \in \mathbb{N}$ in naj bo $k \in \mathbb{N}$ za katerega velja $k \leq n-2$.

TA pošlje po varnem kanalu $k+1$ elementov $\mathbb{Z}_p$ vsaki osebi in nato si lahko vsak par $\{U, V\}$ izračuna svoj ključ $K_{U,V} = K_{V,U}$.

Število k je velikost največje koalicije, proti kateri bo shema še vedno varna.

Paul R. Halmos

"...the source of all great mathematics is the special case, the concrete example. It is frequent in mathematics that every instance of a concept of seemingly great generality is in essence the same as a small and concrete special case."

I Want to be a Mathematician, Washington: MAA Spectrum, 1985.

???

" Sometimes a research is a lot of hard work in looking for the easy way."

David Hilbert (-1900)

"The art of doing mathematics consists in finding that special case which contains all the germs of generality."

Najprej opišimo shemo v primeru, ko je $k = 1$.

- *Izberemo javno praštevilo p.*
- *TA izbere tri naključne elemente $a, b, c \in \mathbb{Z}_p$*

  (ne nujno različne) in oblikuje polinom

$$f(x, y) = a + b(x + y) + c\,xy \mod p.$$

- *Za vsakega uporabnika U izbere TA javni*

  $r_U \in \mathbb{Z}_p$, tako da so le-ti medseboj različni.

- *Za vsakega uporabnika U izračuna TA polinom*

$$g_U(x) = f(x, r_U) \bmod p$$

in mu ga pošlje po varnem kanalu.

Opozorimo, da je $g_U(x)$ linearen polinom, tako da ga lahko zapišemo v naslednji obliki

$$g_U(x) = a_U + b_U x,$$

kjer je

$$a_U = a + br_U \bmod p \quad \text{in} \quad b_U = b + cr_U \bmod p.$$

- *Za medsebojno komunikacijo osebi U in V*

  uporabita ključ

$$\begin{aligned} K_{U,V} = K_{V,U} &= f(r_U, r_V) \\ &= a + b(r_U + r_V) + c\,r_U r_V \bmod p. \end{aligned}$$

Uporabnika U in V izračunata svoja ključa $K_{U,V}$ in $K_{U,V}$ zaporedoma s

$$f(r_U, r_V) = g_U(r_V) \quad \text{in} \quad f(r_U, r_V) = g_V(r_U).$$

Izrek 1. *Blomova shema za $k = 1$ je brezpogojno varna pred posameznimi uporabniki.*

Dokaz: Recimo, da želi uporabnik W izračunati ključ

$$K_{U,V} = a + b(r_U + r_V) + c\, r_U r_V \bmod p.$$

Vrednosti r_U in r_V so javne, a, b in c pa ne.
Oseba W pozna vrednosti

$$a_W = a + b r_W \bmod p \quad \text{in} \quad b_W = b + c r_W \bmod p,$$

ker sta to koeficienta polinoma $g_W(x)$, ki ju je dobila od agencije TA.

Pokažimo, da je informacija, poznana osebi W, konsistentna s poljubno vrednostjo $\ell \in \mathbb{Z}_p$ za ključ $K_{U,V}$, tj. W ne more izločiti nobene vrednosti za $K_{U,V}$.

Poglejmo si naslednjo matrično enačbo v $(\mathbb{Z}_p)$:

$$\begin{pmatrix} 1 & r_U + r_V & r_U\, r_V \\ 1 & r_W & 0 \\ 0 & 1 & r_W \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} \ell \\ a_W \\ b_W \end{pmatrix}.$$

Prva enačba vsebuje hipotezo, da je $K_{U,V} = \ell$, drugi dve enačbi pa sledita iz definicije števil a_W in b_W.

Determinanta zgornje matrike je

$$r_W^2 + r_U r_V - (r_U + r_V) r_W = (r_W - r_U)(r_W - r_V).$$

Iz $r_W \neq r_U$ in $r_W \neq r_V$ sledi, da je determinanta različna od nič in zato ima zgornji sistem enolično rešitev za a, b in c.

Koalicija uporabnikov $\{W, X\}$ pa ima štiri enačbe ter tri neznanke in od tod zlahka izračuna a, b in c ter končno še polinom $f(x, y)$, s katerim dobi vsak ključ. ■

Posplošitev

Za splošno shemo (tj. shemo, ki je varna pred koalicijo velikosti k) je potrebna ena sama sprememba. Pri drugem koraku TA uporablja polinom $f(x, y)$ naslednje oblike

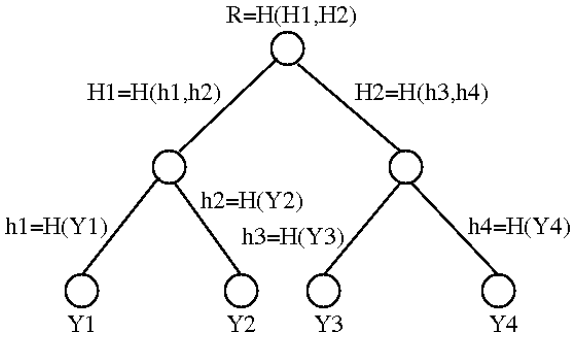
$$f(x, y) = \sum_{i=0}^{k} \sum_{j=0}^{k} a_{ij}\, x^i y^j \bmod p,$$

kjer je $a_{ij} \in \mathbb{Z}_p$ za $0 \leq i, j \leq k$ in $a_{ij} = a_{ji}$ za vsak i, j. Ostali del protokola se ne spremeni.

Avtentična drevesa

- Merkle, 1979.
- metoda za hranjenje javno dostopnih in preverljivo overjenih podatkov
- Uporaba:
  - avtentičnost velike datoteke javnih ključev,
  - servis časovnih oznak (Timestamping).

Primer: H je zgoščevalna funkcija brez trčenj.



Vzdržujemo avtentičnost korenske vrednosti R (npr. s podpisom agencije TA).

Za avtenticiranje javne vrednosti Y_2:

- sledi (natanko določeno) pot od Y_2 do korena,
- pridobi vrednosti h_1, H_2, R,
- preveri avtentičnost R,
- preveri $R = H(H(h_1, H(Y_2)), H_2)$.

Če ima drevo n javnih vrednosti, je dolžina avtenticiranja kvečjemu $\lceil \log_2 n \rceil$.

Slaba stran: dodajanje in brisanje javnih vrednosti je lahko precej zamudna.

Diffie-Hellmanova distribucija ključev

Zaradi enostavnosti bomo delali v obsegu $\mathbb{Z}_p$, kjer je p praštevilo in α generator grupe $\mathbb{Z}_p^*$.

Naj bo $\mathrm{ID}(U)$ oznaka za določeno informacijo, ki enolično identificira osebo U (npr. ime, e-pošta, telefonska številka itd).

Vsak uporabnik si izbere tajni/zasebni $a_U \in \{0, 1, \ldots, p-2\}$, in naj bo

$$b_U = \alpha^{a_U} \bmod p.$$

Agancija TA si izbere shemo za digitalni podpis z javnim algoritmom za preverjanje podpisov $\mathrm{ver}_{\mathrm{TA}}$ in tajnim algoritmom za podpisovanje $\mathrm{sig}_{\mathrm{TA}}$.

Nazadnje privzemimo še, da so vse informacije zgoščene z javno zgoščevalno funkcijo, preden jih podpišemo, vendar pa zaradi estetskih razlogov ne bomo omenjali zgoščevalne funkcije pri opisu protokolov.

Za osebo U bo agancija TA izdala naslednji **certifikat**:

$$C(U) = (\,\mathrm{ID}(U),\, b_U, \mathrm{sig}_{\mathrm{TA}}(\mathrm{ID}(U),\, b_U)\,)$$

(TA ne potrebuje zasebne vrednosti a_U).

- *Izberemo javno praštevilo p in javen primitivni element $\alpha \in \mathbb{Z}_p^*$.*
- *Oseba V izračuna*

$$K_{U,V} = \alpha^{a_U a_V} \bmod p \;=\; b_U^{a_V} \bmod p,$$

z uporabo javne vrednosti b_U iz certifikata osebe U in s svojo zasebno vrednostjo a_V.

- *Oseba U izračuna*

$$K_{U,V} = \alpha^{a_U a_V} \bmod p \;=\; b_V^{a_U} \bmod p,$$

z uporabo javne vrednosti b_V iz certifikata osebe V in s svojo zasebno vrednostjo a_U.

Podpis agencije TA preprečuje osebi W, da spreminja certifikate, torej je dovolj preprečiti pasivne napade.

Ali lahko oseba W izračuna $K_{U,V}$, če je $W \neq U, V$, tj, če poznamo $\alpha^{a_U} \bmod p$ in $\alpha^{a_V} \bmod p$ ne pa tudi a_U ali a_V, ali je mogoče izračunati $\alpha^{a_U a_V} \bmod p$?

To bomo imenovali **Diffie-Hellmanov** problem.

Očitno je **Diffie-Hellmanova distribucija ključev** varna natanko tedaj, ko je varen **Diffie-Hellmanov** problem.

Izrek 2. *Razbitje ElGamalovega kriptosistema je ekvivalentno reševanju Diffie-Hellmanovega problema.*

Dokaz: Spomnimo se, kako potekata ElGamalovo šifriranje in odšifriranje. Ključ je $K = (p, \alpha, a, \beta)$, kjer $\beta = \alpha^a \bmod p$ (a je tajni in p, α in β so javni). Za tajno naključno število $k \in \mathbb{Z}_{p-1}$ je

$$e_K(x, k) = (y_1, y_2),$$

kjer $y_1 = \alpha^k \bmod p$ in $y_2 = x\beta^k \bmod p$.

Za $y_1, y_2 \in \mathbb{Z}_p^*$ je $d_K(y_1, y_2) = y_2 (y_1^a)^{-1} \bmod p$.

Predpostavimo, da imamo algoritem A, ki reši Diffie-Hellmanov problem in podano ElGamalovo šifriranje (y_1, y_2). Z uporabo algoritma A na podatkih p, α, y_1 in β dobimo vrednost

$$A(p, \alpha, y_1, \beta) = A(p, \alpha, \alpha^k, \alpha^a) =$$

$$= \alpha^{ka} \bmod p = \beta^k \bmod p.$$

Potem odšifriranje (y_1, y_2) lahko enostavno izračunamo:

$$x = y_2(\beta^k)^{-1} \bmod p.$$

Predpostavimo, da imamo še algoritem B, ki izvrši ElGamalovo odšifriranje. Torej B vzame podatke p, α, β, y_1 in y_2 in izračuna

$$x = y_2(y_1^{\log_\alpha \beta})^{-1} \bmod p.$$

Naj bodo p, α, β in γ podatki Diffie-Hellmanovega problema. Torej je $\beta = \alpha^b$ in $\gamma = \alpha^c$ za neka $b, c \in \mathbb{N}$, ki nista poznana, pa vendar lahko izračunamo

$$(B(p, \alpha, \beta, \gamma, 1))^{-1} = (1(\gamma^{\log_\alpha \beta})^{-1})^{-1} \bmod p =$$

$$= \gamma^{\log_\alpha \beta} \bmod p = \alpha^{c \cdot b} \bmod p,$$

torej DH-ključ, kar smo tudi želeli. ■

Certifikati

Certifikatna agencija (CA) izda certifikat $C(U)$, ki poveže uporabnika U z njegovim javnim ključem.

Sestavljen je iz:

- **podatkovnega dela $D(U)$**:

  uporabnikova identifikacija, njegov javni ključ in druge informacije kot npr. veljavnost,

- **podpisanega dela $\mathrm{sig}_{\mathrm{CA}}(D(U))$**:

  CA-jev podpis podatkovnega dela.

B pridobi avtentično kopijo A-jevega javnega ključa na naslednji način:

- pridobi avtentično kopijo javnega ključa CA (npr. dobljenega z brskalnikom ali operacijskim sistemom),
- pridobi $C(U)$ (preko nezavarovanega kanala),
- preveri podpis $\mathrm{sig}_{\mathrm{CA}}(D(U))$.

Opombe: 1. CA ni potrebno zaupati uporabniških zasebnih ključev.

2. CA moramo zaupati, da ne bo izdajala ponarejenih certifikatov.

Infrastruktura javnih ključev (PKI)

Nekatere komponente:

- format certifikata,
- proces certificiranja,
- razdeljenanje certifikatov,
- modeli zaupanja,
- preklic certifikatov,
- politika certificiranja: podrobnosti o namenu in obsegu uporabe določenega certifikata.
- Izjava o prakticiranju certificiranja (CPS) (postopki in politike CA).

Format certifikata: X.509 Ver.3

- X.509 originalno predlagan za podporo X.500, ki omogoča servis imenikov na velikih računalniških mrežah.
- Ver. 1 izide leta '88;
  Ver. 2 leta '93;
  Ver. 3 pa leta '97.
- Najnovejši PKI produkti uporabljajo Ver.3.
- Dopušča precejšnjo fleksibilnost.

Podatkovna polja zajemajo:

- verzijo številke certifikata,
- certifikatovo serijsko številko,
- CA-jev podpisni algoritem ID,
- CA-jevo ime v X.500,
- rok veljave,
- uporabnikovo X.500 ime,
- uporabnikova informacija o javnem ključu,
  - algoritmov ID, vrednost javnega ključa,
- Ext. polja: omogočajo vključevanje poljubnega števila dodatnih polj. Primeri:
  - politika certifikata in politika prirejanja, pot certificiranja, omejitve.

Proces certifikacije

1. Generiranje para ključev za CA-jev podpis:
   - varnost zasebnega ključa CA je osrednja,
   - po možnosti opravljena v nepropustni napravi,
   - deljenje delov zasebnega ključa večim modulom, tako da certifikat ne more biti izdan s strani posameznega modula.
2. Generiranje para ključev osebe A:
   - bodisi s stani osebe A ali CA.
3. Zahteva za A-jev certifikat:
   - lahko, da bo CA kasneje potrebovala to zahtevo,
   - avtentičnost zahteve je potrebna.

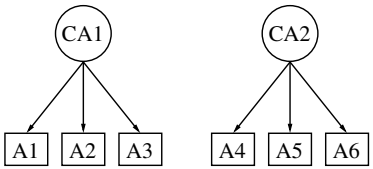
4. Identiteta osebe A je preverjena:
   - to je lahko zamudno in drago v praksi,
   - preložiti to delo na Registration Authority (RA); npr. pošto ali banko,
   - RA generira registracijski certifikat in ga prosledi CA za izdajo certifikata.
5. A-jev par ključev je preverjen:
   - CA preveri, da je javni ključ veljaven, tj. zasebni ključ logično obstaja,
   - A dokaže, da ima zasebni ključ.
6. CA naredi A-jev certifikat.
7. A preveri, da je certifikat izpraven:
   - CA lahko zahteva od A še potrdilo od prejemu.

Primer: Verisignov digitalni ID

- www.verisign.com/client/index.html
- Certifikat za javno podpisovanje in javno šifriranje.
- Certifikati so hranjeni v brskalniku ali e-poštni programski opremi.
- Brezplačni certifikati za 60-dnevno preiskusno dobo.
- Trije razredi certifikatov:
  - odgovornost prevzema Verisign (US $100, $5,000, $100,000),
  - potrditev identitete,
  - zaščita CA-jevega zasebnega ključa,
  - zaščita posameznih uporabnikovih zasebnih ključev.
- `www.versign.com/repository/index.html`

Model zaupanja

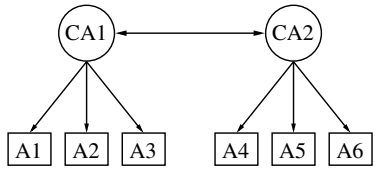
- strukturiran odnos med številnimi CA-ji.



- Stranke dobijo avtentične kopije CA-jevega javnega ključa (zunaj tekočega obsega - out-of-band, npr. med certifikacijo).
- Kako lahko A_1 preveri podpis sporočila osebe A_5? Tj. kako lahko dobi overjeno kopijo javnega ključa od A_1?
- A_1 potrebuje overjeno kopijo javnega ključa od CA_2.

Navzkrižna certifikacija

- CA-ji si lahko medsebojno overijo javne ključe

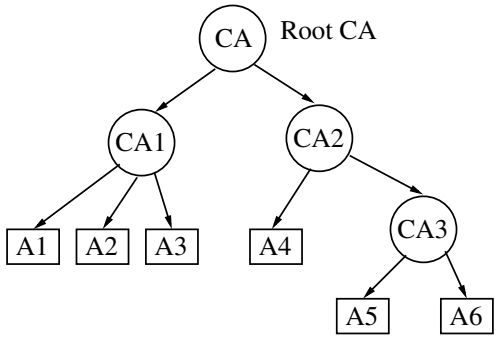


- A_1 pridobi A_5-jev overjeni javni ključ:
  - Pridobitev certifikatov CA_2 in A_5 z javnega (nezaščitenega, ne-overjenega) imenika.
  - Preveri od CA_1 podpisan certifikat CA_2 (s tem dobi overjeno kopijo javnega ključa CA_2).
  - Preveri od CA_2 podpisan certifikat A_5 (s tem dobi overjeno kopijo javnega ključa A_5).

Pomisleki glede navskrižnega certificiranja

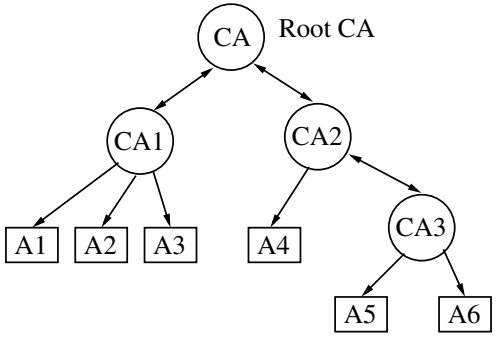
- Ali je CA_1 odgovoren osebi A_1 za varnostne probleme v domeni CA_2?
  - Potencialni problemi so lahko omejeni z izjavo v politiki CA_1 za CA_2 certifikate.
  - CA_1 mora previdno preveriti CA_2jev CPS.
  - Neodvisni pregled politike CA_2 bo pomagal.
- Ali je CA_1 odgovoren osebam iz CA_2 domene za varnostne probleme v svoji domeni?
- Vprašanje: ali bodo problemi navskrižnega certificiranja za obsežnejše aplikacije *kdaj* rešeni?

Strogo hierarhičen model



- Vsi vipis začenjajo z overjeno kopijo korenskega javnega ključa.
- Zadrški:
  - vse zaupanje je odvisno od korenskega CA,
    - rešitev: razdeli dele zasebnega ključa;
  - Certifikatne verige lahko postanejo predolge,
    - rešitev: nekatere certifikate spravimo v cache.
  - Certifikatne verige zahtevane celo za osebe znotraj iste CA,
    - rešitev: nekatere certifikate spravimo v cache.

Povratni hierarhičen model



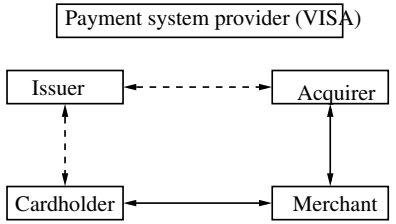
- CA lahko preveri javni ključ starševskega CA.
- Vsaka oseba prične z overjenim javnim ključem svojega CA.
- Najkrajša veriga zaupanja med A in B je pot od A do najmlajšega skupnega prednika od A in B, in nato navzdol do B.

Secure Electronic Transaction (SET)

- Standard, ki sta ga predlagala Visa in MasterCard (Feb 1996).
- Glej www.setco.org
- Cilj: varne transakcije s kreditnimi karticami prko Interneta.

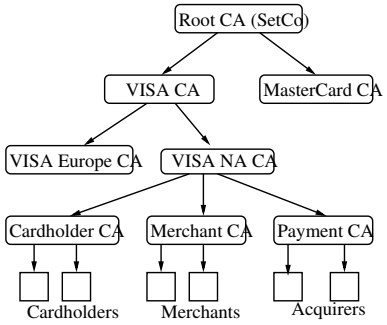
- Sodelujoči pri transakciji s kreditno kartico:
  - *Izdajatelj*: finančno podjetje, ki izdaja kreditne kartice.
  - *Lastnik kartice*: Nepooblaščen imetnik kreditne kartice holder of a credit card who is registered with the corresponding issuer.
  - *Prodajalec*: trgovec, services, or information, who accepts payment electronically.
  - *Dobavitelj*: finančna inštitucija, ki podpira prodajalca s tem, da ponuja servis za procesiranje transakcij z bančnimi karticami.

- Plačilo s kreditno kartico:



- Po Internetu: $C \longleftrightarrow M$ in $M \longleftrightarrow A$.
- Šifriranje se uporabi za zaščito številk kreditnih kartic med prenosom po Internetu; številke niso razkrite prodajalcu.
- Digitalni podpisi se uporabljajo za celovitost podatkov in overjanje udeleženih strank.

SET-ov hierarhični PKI



Preklic certifikata

- Razlogi za preklic certifikata:
  - kompromitiran ključ (redko).
  - Lastnik zapusti organizacijo.
  - Lastnik spremeni vlogo v organizaciji.
- Primer: Scotiabank tele-banking PKI:
  - Čez 90,026 certifikatov izdanih do aprila 21, 1999.
  - Čez 19,000 certifikatov preklicanih.
- Uporabnik naj bi preveril veljavnost certifikata pred njegovo uporabo.
- Preklic je enostaven v primeru on-line CA.

Certifikatne preklicne liste (CRL)

- Lista preklicanih certifikatov, ki je podpisana in periodično izdana od CA.
- Uporabnik preveri CRL predno uporabi certifikat.

Problemi z CRLs

- časovna prerioda CRL
  - Čas med preklicom in obnovitvijo CRL.
- velikost CRL
  - Delta CRL: vključuje le zadnje preklicane certifikate.
  - Groupiraj razloge za preklic.
  - Delitvene točke: revocation data is split into buckets; each certificate contains data that determines the bucket it should be placed in (patent: Entrust Technologies).
  - Uporabi avtentikacijska drevesa (komercializacija: Valicert).

Kerberos

Doslej smo spoznali sisteme, kjer vsak par uporabnikov izračuna fiksen ključ, ki se ne spreminja.
Zaradi tega je preveč izpostavljen nasprotnikom.

Zato bomo vpeljali tako imenovan sejni ključ, ki se oblikuje brž, ko se pojavita dva, ki želita komunicirati.

Tak sistem, ki uporablja simetrične sisteme, je Kerberos. Slabost tega sistema pa je zahteva po sinhronizaciji ur uporabnikov omrežja.

Določena časovna variacija je dovoljena.

Predpostavimo, da vsak uporabnik deli z agencijo TA tajni DES ključ K_U.
Tako kot prej imejmo tudi $\mathrm{ID}(U)$.

Ko dobi agencija TA zahtevo po novem sejnem ključu, si TA izbere naključni ključ K, zabeleži časovno oznako T (timestamp), določi življenjsko dobo L (lifetime) za ključ K ter vse skupaj pošlje uporabnikoma U in V.

Prenos sejnega ključa z uporabo Kerberosa

- *Uporabnik U zahteva od agencije TA sejni ključ za komunikacijo z uporabnikom V.*
- *Agancija TA izbere naključni sejni ključ K, časovno oznako T in življenjsko dobo L.*
- *TA izračuna* $m_1 = e_{K_U}(K, \mathrm{ID}(V), T, L)$ *in* $m_2 = e_{K_V}(K, \mathrm{ID}(U), T, L)$ *ter ju pošlje uporabniku U.*
- *U uporabi odšifrirno funkcijo d_{K_U}, da dobi iz m_1 K, T, L in $\mathrm{ID}(V)$. Potem izračuna $m_3 = e_K(\mathrm{ID}(U), T)$ in ga pošlje osebi V skupaj s sporočilom m_2, ki ga je dobil od agencije TA.*

- *V uporabi odšifrirno funkcijo d_{K_V}, da dobi iz m_2 K, T, L in $\mathrm{ID}(U)$. Potem uporabi d_K, da dobi T in $\mathrm{ID}(U)$ iz m_3. Preveri, da sta tako dobljeni vrednosti za T in $\mathrm{ID}(U)$ enaki prejšnjim. Če je tako, potem izračuna še*

  $$m_4 = e_K(T + 1)$$

  in ga pošlje uporabniku U.
- *U odšifrira m_4 z uporabo e_K in preveri, ali je rezultat enak $T + 1$.*

V tem protokolu se prenašajo različne funkcije sporočil.

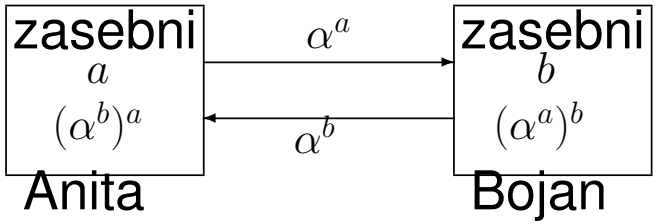
Sporočili m_1 in m_2 poskrbita za tajnost pri prenosu sejnega ključa K.

Sporočili m_3 in m_4 se uporabljata kot potrdilo sejnega ključa K tako, da se U in V prepričata, da imata res isti sejni ključ K.

Diffie-Hellmanova uskladitev ključev

Naj bo p praštevilo in α generator multiplikativne grupe $\mathbb{Z}_p^*$. Naj bosta oba javno poznana
(ali pa naj ju oseba U sporoči osebi V).

1. *Oseba U izbere naključen a_U,* $0 \le a_U \le p - 2$,
   izračuna $\alpha^{a_U} \bmod p$ *in ga pošlje osebi V.*
2. *Oseba V izbere naključen a_V,* $0 \le a_V \le p - 2$,
   izračuna $\alpha^{a_V} \bmod p$ *in ga pošlje osebi U.*
3. *Osebi U in V izračunata zaporedoma*
   $K = (\alpha^{a_V})^{a_U} \bmod p$ *in* $K = (\alpha^{a_U})^{a_V} \bmod p$.



Anita in Bojan si delita skupni element grupe:

$$(\alpha^a)^b = (\alpha^b)^a = \alpha^{ab}.$$

Edina razlika med tem protokolom in pa Diffie-Hellmanovim protokolom za distribucijo ključev je, da si izberemo nova eksponenta a_U in a_V uporabnikov U in V zaporedoma vsakič, ko poženemo ta protokol.

Varnost Diffie-Hellmanovega protokola

Protokol ni varen pred aktivnim napadalcem,
ki prestreže sporočila in jih nadomesti s svojimi.
Ta napad bomo imenovali **napad srednjega moža**.

$$\xrightarrow{\alpha^{a_U}} \qquad \xrightarrow{\alpha^{a'_U}}$$

$$\xleftarrow[\alpha^{a'_V}]{U} \qquad W \qquad \xleftarrow[\alpha^{a_V}]{V}$$

Na koncu sta osebi U in V vzpostavili z napadalcem W zaporedoma ključa $\alpha^{a_U a'_V}$ in $\alpha^{a'_U a_V}$.

Tako bo zašifrirano sporočilo osebe U odšifriral napadalec W ne pa oseba V.

Uporabnika U in V bi bila rada prepričana, da ni prišlo namesto medsebojne izmenjave sporočil do izmenjave z napadalcem W.

Potrebujeta protokol za medsebojno avtentikacijo (predstavitev).

Dobro bi bilo, če bi potekala avtentikacija istočasno z uskladitvijo ključev, saj bi s tem onemogočili aktivnega napadalca.

Overjena uskladitev ključev

Diffie, Van Oorschot in Wiener so predlagali protokol **uporabnik-uporabniku** (station-to-station - STS), ki je protokol za *overjeno uskladitev ključa* in je modifikacija Diffie-Hellmanove uskladitve ključev.

Vsak uporabnik ima **certifikat (potrdilo)**

$$C(U) = \Big(\mathrm{ID}(U), \mathrm{ver}_U, \mathrm{sig}_{\mathrm{TA}}\big(\mathrm{ID}(U), \mathrm{ver}_U\big)\Big),$$

kjer je shranjena njegova identifikacija $\mathrm{ID}(U)$.

Poenostavljen protokol uporabnik-uporabniku

1. *Oseba U izbere naključen $a_U \in \{0, \ldots, p-2\}$,*
   izračuna $\alpha^{a_U} \bmod p$ *in pošlje osebi V.*

2. *Oseba V izbere naključen $a_V \in \{0, \ldots, p-2\}$,*
   izračuna $\alpha^{a_V} \bmod p$,
   $$K = (\alpha^{a_U})^{a_V} \bmod p \quad \text{in} \quad y_V = \mathrm{sig}_V(\alpha^{a_V}, \alpha^{a_U}),$$
   ter pošlje potrdilo $(C(V), \alpha^{a_V}, y_V)$ *osebi U.*

3. *Oseba U izračuna* $K = (\alpha^{a_V})^{a_U} \bmod p$ *ter*
   preveri podpis y_V *z uporabo* ver_V *in*
   potrdilo $C(V)$ *z* ver_{TA}.

   Nato izračuna $y_U = \text{sig}_U(\alpha^{a_U}, \alpha^{a_V})$
   in pošlje potrdilo $(C(U), y_U)$ *osebi* V.

4. *Oseba V preveri podpis* y_U *z uporabo* ver_U *in*
   potrdilo $C(U)$ *z uporabo* ver_{TA}.

Varnost protokola STS

Uporabnika U in V si izmenjata naslednje informacije (izpustimo potrdila):

$$\xrightarrow{\quad\alpha^{a_U}\quad}$$
$$U \xleftarrow{\quad\alpha^{a_V},\ \text{sig}_V(\alpha^{a_V}, \alpha^{a_U})\quad} V$$
$$\xrightarrow{\quad\text{sig}_U(\alpha^{a_U}, \alpha^{a_V})\quad}$$

Kaj lahko naredi napadalec W (mož na sredini):

$$\xrightarrow{\quad\alpha^{a_U}\quad} \qquad \xrightarrow{\quad\alpha^{a'_U}\quad}$$
$$U \xleftarrow{\quad\alpha^{a'_V},\ \text{sig}_V(\alpha^{a'_V}, \alpha^{a_U})=?\quad} W \xleftarrow{\quad\alpha^{a_V},\ \text{sig}_V(\alpha^{a_V}, \alpha^{a'_U})\quad} V$$
$$\xrightarrow{\quad\text{sig}_U(\alpha^{a_U}, \alpha^{a'_V})\quad} \qquad \xrightarrow{\quad\text{sig}_U(\alpha^{a'_U}, \alpha^{a_V})=?\quad}$$

Poenostavljeni STS protokol je torej varen pred napadom srednjega moža.

Tako oblikovan protokol ne vsebuje potrditve ključa, kakor je slučaj v Kerberosovi shemi.

Protokol, v katerem je vključena potrditev ključa:

$$y_V = e_K(\text{sig}_V(\alpha^{a_V}, \alpha^{a_U})), \quad y_U = e_K(\text{sig}_U(\alpha^{a_U}, \alpha^{a_V}))$$

se imenuje STS protokol.

MTI protokoli

Matsumoto, Takashima, Imai so modificirali Diffie-Hellmanovo uskladitev ključev, tako da uporabniki U in V ne potrebujejo podpisov.

Kadar moramo izmenjati dve pošiljki, pravimo, da gre za **protokole z dvema izmenjavama**.

Predstavili bomo en njihov protokol.

Osnovne predpostavke so enake kot pri Diffie-Hellmanovi uskladitvi ključev: praštevilo p in generator α multiplikativne grupe $\mathbb{Z}_p^*$ sta javna.

Vsak uporabnik U ima svoj *zasebni* eksponent a_U ($0 \le a_U \le p-2$) in *javno* vrednost $b_U = \alpha^{a_U} \bmod p$.

Agencija TA ima shemo za digitalni podpis, z *javnim* algoritmom ver_{TA} in *tajnim* algoritmom sig_{TA}.

Vsak uporabnik U ima svoj certifikat:

$$C(U) = (\text{ID}(U), b_U, \text{sig}_{\text{TA}}(\text{ID}(U), b_U)).$$

1. *Oseba U izbere naključen $r_U \in \{0, \ldots, p-2\}$,*

   izračuna $s_U = \alpha^{r_U} \bmod p$ *in*

   pošlje osebi V $(C(U), s_U)$.

2. *Oseba V izbere naključen $r_V \in \{0, \ldots, p-2\}$,*

   izračuna $s_V = \alpha^{r_V} \bmod p$ *in*

   pošlje osebi U $(C(V), s_V)$.

3. *Osebi U in V izračunata zaporedoma*

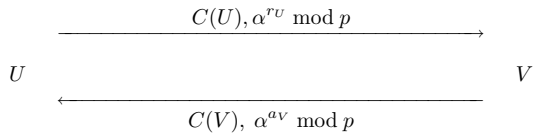
$$K = s_V^{a_U} b_V^{r_U} \bmod p \quad \text{in} \quad K = s_U^{a_V} b_U^{r_V} \bmod p,$$

   kjer sta b_V in b_U zaporedoma iz $C(V)$ in $C(U)$.

Varnost protokola MTI

Ta MTI protokol je enako varen pred pasivnimi sovražniki kot Diffie-Hellmanov protokol.

Varnost pred aktivnimi sovražniki je bolj vprašljiva. Brez uporabe podpisnega algoritma nismo varni pred napadom srednjega moža.



Ključ uporabnikov, ki komunicirata, je težko izračunati, ker je v ozadju težko izračunljiv diskretni logaritem.

Tej lastnosti pravimo **implicitna overitev ključev**.

Uskladitev ključev s ključi, ki se sami overijo

Giraultova shema ne potrebuje certifikatov, saj uporabnike razlikujejo že njihovi javni ključi in identifikacije.

Vsebuje lastnosti RSA sheme in diskretnega logaritma.

Uporabnik naj ima identifikacijo $\mathrm{ID}(U)$.
Javni ključ za osebno overitev dobi od agencije TA.

Naj bo $n = p\,q$, kjer je $p = 2p_1 + 1$, $q = 2q_1 + 1$, in so p, q, p_1, q_1 velika praštevila. Potem je

$$(\mathbb{Z}_n^*, \cdot) \sim (\mathbb{Z}_p^* \times \mathbb{Z}_q^*, \cdot).$$

Največji red poljubnega elementa v $\mathbb{Z}_n^*$ je najmanjši skupni večkratnik elementov $p-1$ in $q-1$ oziroma $2p_1q_1$.

Naj bo α generator ciklične podgrupe v $\mathbb{Z}_p^*$ reda $2p_1q_1$, problem diskretnega logaritma v tej podgrupi pa naj bo računsko prezahteven za napadalca.

Javni ključ za osebno overitev

Naj bosta števili n, α javni,

števila p, q, p_1, q_1 pa naj pozna samo agencija TA.

Število e je javni RSA šifrirni eksponent in ga izbere agancija TA,
$d = e^{-1} \bmod \varphi(n)$ *pa je tajni odšifrirni eksponent.*

1. *Oseba U izbere tajni eksponent a_U,*

   izračuna $b_U = \alpha^{a_U} \bmod n$ *in*

   izroči a_U ter b_U agenciji TA.
2. *Agencija TA izračuna*

   $p_U = (b_U - \mathrm{ID}(U))^d \bmod n$ *ter ga izroči osebi U.*

Giraultov protokol za uskladitev ključev

1. *Oseba U izbere naključen zasebni r_U, izračuna*

$$s_U = \alpha^{r_U} \bmod n$$

ter pošlje ID*(U), p_U in s_U osebi V.*

2. *Oseba V izbere naključen zasebni r_V, izračuna*

$$s_V = \alpha^{r_V} \bmod n$$

ter pošlje ID*(V), p_V in s_V osebi U.*

3. *Osebi U in V izračunata ključ K zaporedoma z*

$$s_V^{a_U}\,(p_V^e + \mathrm{ID}(V))^{r_U} \bmod n, \quad s_U^{a_V}\,(p_U^e + \mathrm{ID}(U))^{r_V} \bmod n.$$

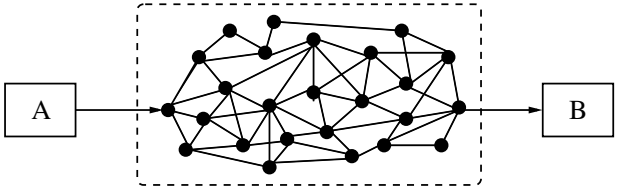
Varnost Giraultovega protokola

Ključ za osebno overitev varuje pred sovražniki.

Protokol implicitno overi ključe, zato napad srednjega moža ni možen.

Agencija TA je prepričana, da uporabnik pozna vrednost števila a predno izračuna ključ za osebno overitev.

Internetne aplikacije

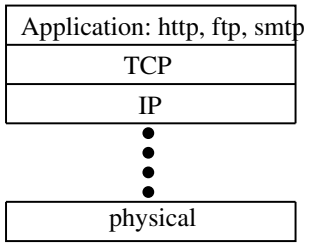


- ftp: File Transfer Protocol
- http: HyperText Transfer Protocol
- smtp: Simple Mail Transfer Protocol

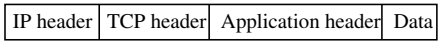
TCP – Transport Control Protocol
IP – Internet Protocol

TCP/IP

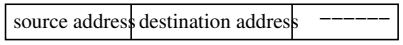
Protokolov sklad:



TCP/IP paket:



IP header:



Nekateri napadi

- **IP address spoofing** (slov. ponarejanje naslovov)
  rešitev: overi glavo IP paketa
- **IP packet sniffing** (slov. vohljanje za IP paketi)
  rešitev: zašifriraj IP payload (vse kar se prenaša)
- **Traffic analysis** (slov. Analiza prometa)
  rešitev: zašifriraj pošiljateljev in prejemnikov naslov

Varnost znotraj TCP/IP

Varnostni protokoli so prisotni na različnih nivojih TCP/IP sklada.

1. IP nivo: IPsec.
2. Transportni nivo: SSL/TLS.
3. Aplikacijski nivo: PGP, S/MIME, SET, itd.

Internet Engineering Task Force (IETF)

- Sprejema standarde za razvoj Internetne arhitekture in omogoča nemoteno delovanje Interneta.
- Odprta za vse zainteresirane posameznike: `www.ietf.org`
- Delo, ki ga opravljajo delovne skupine povezane z varnostjo (Security Area) pokrivajo:

– IP Security Protocol (IPsec)

– Transport Layer Security (TLS)

– S/MIME Mail Security

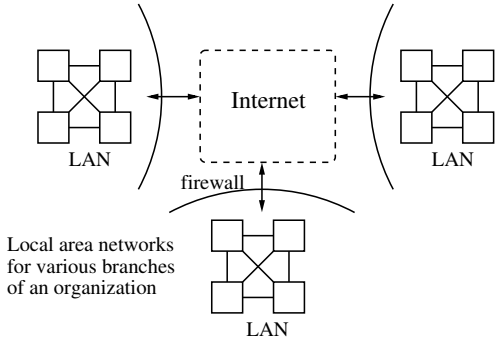
– Odprto specifikacijo za PGP (OpenPGP)

– Secure Shell (secsh)
(Nova verzija ssh protokola, ki omogoča varno prijavo na oddaljene šifre in varen prenos datotek.)

– X.509 Public-Key Infrastructure (PKIX)

IPsec: Virtual Private Networks (VPNs)

Omogočajo šifriranje in overjanje
(overjanje izvora podatkov, celovitost podatkov) na IP layer.



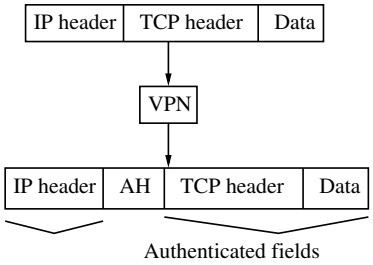
Gradniki IPsec

- Security Association (SA):
  - upravlja algoritme in ključe med sogovorniki,
  - vsaka glava IPsec se nanaša na Security Association preko Security Parameter Index (SPI).
- Upravljanje s ključi:
  - dogovor o ključu z Diffie-Hellmanovo shemo (OAKLEY),
  - kreira ključe za Security Association,
  - upravljanje z javnimi ključi, ki ni pokrito v IPsec.
- Trije načini IPsec servisov:
  - AH: overjanje,
  - ESP: šifriranje + overjanje.

IPsec glava za overjanje (AH)

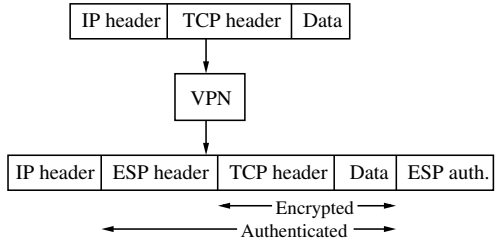
– Podpira MACs: HMAC-MD5-96, HMAC-SHA-1-96.

– Transportni način:



IPec ESP glava

- Encapsulating Security Payload.
- Podprti šifrirni algoritmi: 3-DES, RC5, IDEA, ...
- Transportni način:

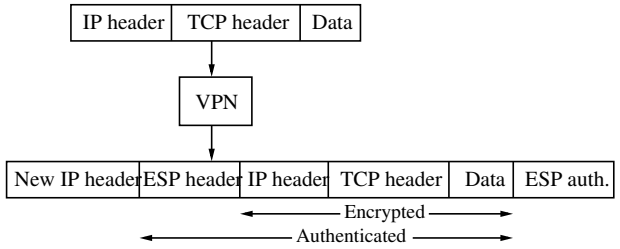


- Opomba: analiza prometa je še vedno možna (ker IP glave niso šifrirane).

ESP v tunelskem načinu

– Požarni zid vključi novo IP glavo
(IP naslov pošiljateljevega požarnega zidu in
IP naslov prejemnikovega požarnega zidu).

– Možna je samo zelo omejena analiza prometa.



Secure Sockets Layer (SSL)

- SSL je naredil Netscape.
- TLS (Transport Layer Security)
je IETF-ova verzija SSL-a.
- SSL uporabljamo v brskalnikih (npr. Netscape)
za zaščito mrežnih transakcij.
- Osnovne komponente SSL/TLS:

  handshake protocol: dopusti strežniku in klientu, da se overita in dogovorita za kriptografske ključe,

  record protocol: uporabljan za šifriranje in overjanje prenašanih podatkov.

Upravljanje z javnimi ključi v SSL/TLS

- Korenski CA ključ je vnaprej inštaliran v brskalnik.
  – Klik na “Security” in nato na “Signers”, da najdete seznam ključev korenskih CA v Netscape-u.
- Mrežnim strežnikom certificirajo javne ključe z enim izmed korenskih CA-jev (seveda brezplačno).
  – Verisign-ov certification business za mrežne strežnike `www.verisign.com/server/index.html`

- Klienti (uporabniki) lahko pridobijo svoje certifikate.
Večina uporabnikov trenutno nima svojih lastnih certifikatov.
  – Če klienti nimajo svojih certifikatov,
potem je overjanje samo enostransko
(strežnik se avtenticira klientu).
  – Obiščite varno internetno stran kot npr.
webbroker1.tdwaterhouse.ca in kliknite na
“padlock” v Netscapu, da si ogledate
informacijo o strežnikovem certifikatu.

SSL/TLS handshake protocol

Na voljo so naslednji kriptografski algoritmi:

- MAC: HMAC-SHA-1, HMAC-MD5.
- šifriranje s simetričnimi ključi: IDEA, RC2-40,
DES-40, DES, Triple-DES, RC4-40, RC4-128.
- Osnovne sheme za dogovor o ključu so:

– RSA transport ključev: deljeno skrivnost izbere klient in jo zašifrirana s strežnikovim javnim RSA ključem.

– Fixed Diffie-Hellman: strežnikov Diffie-Hellman-ov javni ključ g^x je v njegovem certifikatu.
Klient ima lahko g^y v svojem certifikatu,
ali generira enkratno vrednost g^y.

– Ephemeral Diffie-Hellman: Strežnik izbere enkratni Diffie-Hellman-ov javni ključ g^x in ga podpiše s svojim RSA ali DSA ključem za podpise.
Klient izbere enkratni g^y in ga podpiše če in samo če ima certifikat.

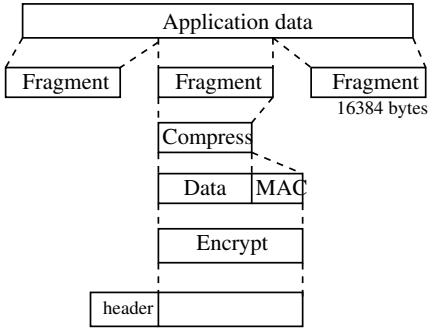
– MAC in šifrirni ključi so izpeljani iz skupne skrivnosti.

SSL/TLS handshake protokol (2)

1. faza: Določi varnostne zmožnosti.
   - Verzija protokola, način kompresije, kriptografski algoritmi,...
2. faza: Strežnikovo overjanje in izmenjava ključev.
   - Strežnik pošlje svoj certifikate, in (morda še) parametre za izmenjavo ključev.
3. faza: Klientovo overjanje in izmenjava ključeve.
   - Klient pošlje svoj certifikat (če ga ima) in parametre za izmenjavo ključev.
4. faza: Zaključek.

SSL/TLS record protocol

Predpostavimo, da klient in strežnik delita MAC tajnega ključa in sejni šifrirni ključ:



10 zapovedi komercialne varnosti
(Adi Shamir 30th Aug. 1995)

1. Ne meri na perfektno varnost!
2. Ne rešuj napačnih problemov!
3. Ne prodajaj na glavo obrnjene varnosti (problem vodstva)!
4. Ne uporabljaj kriptografskih kanonov za muhe!
5. Ne kompliciraj po nepotrebnem!
6. Ne postavljaj dragih rešitev!
7. Ne uporabljaj ene same linije obrambe!
8. Ne pozabi na “misteriozen napad”!
9. Ne zaupaj sistemom!
10. Ne zaupaj ljudem!