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 komplicirana, 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:

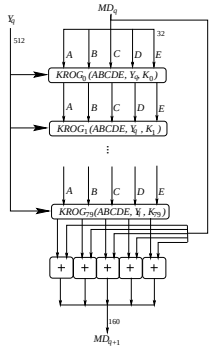
1. **SHS** privzame *big-endian* arhitekturo namesto *little-endian*.
2. **SHS** dobi 160-bitni rezultat (5 registrov).
3. **SHS** obdela 16 besed naenkrat, vendar jih najprej razširi v 80 besed, potem pa uporabi zaporedje 80-ih operacij na vsaki besedi.

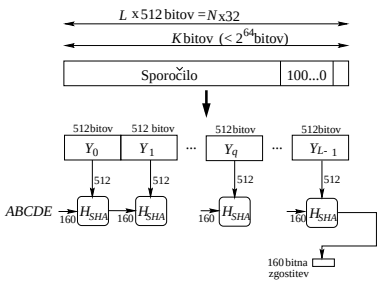
   $X[j] = X[j-3] \oplus X[j-8] \oplus X[j-14] \oplus X[j-16]$

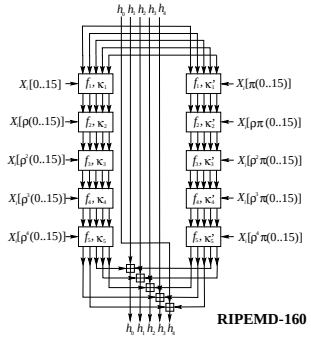
   za $16 \leq j \leq 79$.
4. **SHA-1** pa uporabi

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

   za $16 \leq j \leq 79$.







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. **Timestamper - TS**), ki je elektronski notar (angl. trusted timestamping service) oz. center zaupanja (TTP, angl. trusted third party).

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

8. poglavje

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
$$a_W = a + br_W \bmod p \quad \text{in} \quad b_W = b + cr_W \bmod p,$$
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. ■

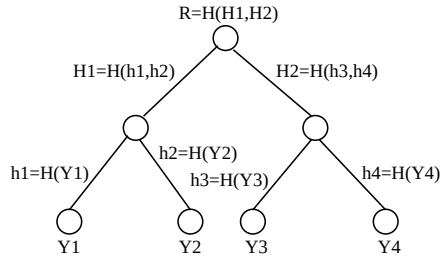
Posplošitev

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
$$f(x, y) = \sum_{i=0}^{k} \sum_{j=0}^{k} a_{ij}\, x^i y^j \mod p,$$
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 $\boldsymbol{C(U)}$, ki poveže uporabnika U z njegovim javnim ključem.

Sestavljen je iz:

- **podatkovnega dela $\boldsymbol{D(U)}$**:
  uporabnikova identifikacija, njegov javni ključ in druge informacije kot npr. veljavnost,
- **podpisanega dela $\mathbf{sig_{CA}}(\boldsymbol{D(U)})$**:
  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.

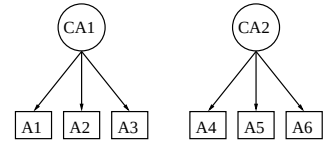
4. Identiteta osebe A je preverjena:
   - to je lahko zamudno in drago v praksi,
   - preložiti to delo na Registration Authority (RA); npr. pošto ali banko,
   - RA generira registracijski certifikat in ga prosledi CA za izdajo certifikata.
5. A-jev par ključev je preverjen:
   - CA preveri, da je javni 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

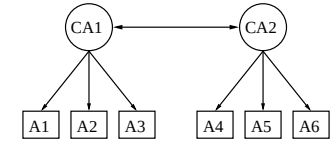
- strukturiran odnos med številnimi CA-ji.



- 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

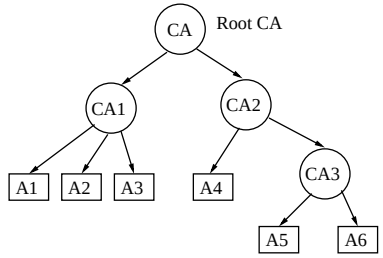


- A_1 pridobi A_5-jev overjeni javni ključ:
  - Pridobitev certifikatov CA_2 in A_5 z javnega (nezaščitenega, ne-overjenega) imenika.
  - Preveri od CA_1 podpisan certifikat CA_2 (s tem dobi overjeno kopijo javnega ključa CA_2).
  - Preveri od CA_2 podpisan certifikat A_5 (s tem dobi overjeno kopijo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

Pomisleki glede navskrižnega certificiranja

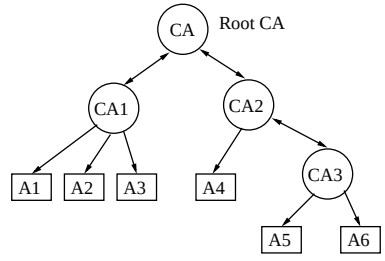
- Ali je CA_1 odgovoren osebi A_1 za varnostne probleme v domeni CA_2?
  - Potencialni problemi so lahko omejeni z izjavo v politiki CA_1 za CA_2 certifikate.
  - CA_1 mora previdno preveriti CA_2jev CPS.
  - Neodvisni pregled politike CA_2 bo pomagal.
- Ali je CA_1 odgovoren osebam iz CA_2 domene za 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



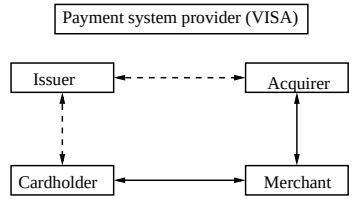
- CA lahko preveri javni ključ starševskega CA.
- Vsaka oseba prične z overjenim javnim ključem svojega CA.
- Najkrajša veriga zaupanja med A in B je pot od A do najmlajšega skupnega prednika od A in B, in nato navzdol do B.

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.

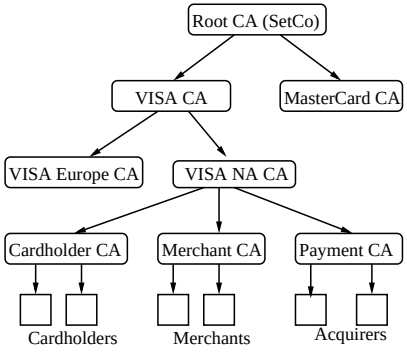
- Sodelujoči pri transakciji s kreditno kartico:
  - *Izdajatelj*: finančno podjetje, ki izdaja kreditne kartice.
  - *Lastnik kartice*: Nepooblaščen imetnik kreditne kartice holder of a credit card who is registered with the corresponding issuer.
  - *Prodajalec*: trgovec, services, or information, 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.