DO-LU-X-22-1297

Ajatempli protokollid, turvavajadused ja tehnilised nõuded

Ahto Buldas, Peeter Laud, Helger Lipmaa, Jan Villemson

{ahtbu,helger}@cyber.ee, {peeter_l,jan}@cs.ut.ee

Küberneetika AS
Akadeemia tee 21
EE0026 Tallinn
Estonia

22. detsember 1997. a.

Töö on tellinud Eesti Informaatikakeskus seoses digitaalsignatuure reguleeriva seaduse väljatöötamisega.

Sisukord

Peatükk 1

Sissejuhatus

Üks olulisemaid turvaeesmärke (kõrvuti konfidentsiaalsusega) on autentsus, mille saavutamiseks hajussüsteemides kasutatakse tavaliselt digitaalsignatuure ([DH76], [RSA78]). Autentsust on vaja tõestada väga erinevates situatsioonides, nii väga lühiajaliselt (automaatsetes autentimisprotokollides) kui ka väga pikaajaliselt (arhiivides säilitatavad elektrondokumendid, muuhulgas ka elektroonilisel kujul lepingud). Teatud digitaalsignatuuride (soovitud) eluiga võib ulatuda kümnetesse aastatesse ning on vaja (nii organisatoorseid kui matemaatilisi) vahendeid ja meetodeid, mis võimaldaksid signatuuri korrektsuse verifitseerimist dokumendi eluea jooksul. Järgnevas valgustame neid probleeme ning pakume neist mõningatele ka lahenduse.

1.1 Ajatemplid ning nende linkimine

Üks neist probleemidest on aja mõiste sisse tulemine. Digitaalsignatuur eraldi võetuna võimaldab meil tõestada, et keegi on kunagi midagi öelnud, kuid ei võimalda tõestada, **millal** öeldi. Aja vajalikkusest võib tuua järgmise näite. Olgu meie käsutuses string $\{X\}_{K_A}$ (üldjuhul tähendab $\{X\}_K$ salajase võtmega K^{-1} signeeritud stringi X), kus K_A^{-1} on Alice'i salajane võti ning X kujutab endast lepingut, millega Alice'i kohustub meile maksma 3 miljonit dollarit. Alice võib hoiduda lepingut täitmast, avaldades meedias salajase võtme K_A^{-1} ning väites, et leping oli signeeritud kellegi kolmanda poolt **pärast** tema võtme avalikuks tulekut. Võltsimaks ülaltoodud meetodile sarnaseid vastutusest hoidumisi, peab olema võimalik tõestada ka seda, **millal** keegi midagi ütles. Signatuuris peab sisalduma ka ütlemise aeg

$$A \xrightarrow{1} \mathrm{TSS} : (y_n, \mathrm{ID}_n)$$

$$\mathrm{TSS} \xrightarrow{2} A : \{n, t_n, y_n, \mathrm{ID}_n, L_{n-1};\ L_n\}_{K_{\mathrm{TSS}}^{-1}}$$

Siin $L_n = H(n, t_n, y_n, \mathrm{ID}_n, L_{n-1})$ (protokolli seletus on toodud peatükis 2). Järgmise päringu saamisel saadab TSS suuruse ID_{n+1} Alice'ile.

Joonis 1.1: Ajatempliprotokoll

ehk teatega X seotud **ajatempel**.

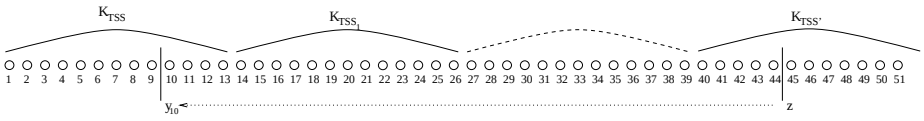
Lihtne on tõestada, et midagi on öeldud **pärast** teatud ajahetke (pannes ajatemplisse sisse talse seisu, korvpalli tulemused vms); hoopis raskem on tõestada, et midagi oli öeldud **enne** mingit ajahetke. Enne ütlemise tõestamist võimaldavate ajatemplite moodustamiseks on olemas mitmeid erinevaid meetodeid (mõningane ülevaade on toodud artiklis [MQ97]), kõik need meetodid eeldavad nn kolmanda usaldatava osapoole olemasolu. Üks kõige levinumaid ning perspektiivsemaid meetodeid on järgmine. Kolmas osapool ehk **ajatempli server** (servereid võib olla mitu) vastab erinevate osapoolte poolt tulnud päringutele, lisades esitatud stringile ajatempli ning signeerides tulemuse. Ajatempli server peab meeles kõik tema poolt siiamaani signeeritud ajatemplid, et võimaldada hilisemat ajatemplite kontrolli. Võltsimise (ajatemplite hilisema lisamise, muutmise või/ja kustutamise) vältimiseks peavad kõik ajatemplid serveri mälus olema teineteisega seotud (ehk **lingitud**) ahelasse viisil, mis välistab üksikute ajatemplite kustutamist, muutmist või lisamist ahelasse. Nii näiteks võib igasse järgmisse ajatemplisse kirja panna eelmise ajatempli põhjal arvutatud sõnumilühendi. Seega, kontrollides üle kõik ajatemplid alates viimati signeeritust ning lõpetades nõutud ajatempliga ning vigu mitte leides, saaksime "ideaalsel" juhul kindlad olla selles, et leitud ajatempel tõepoolest korrektne on.

Tegelikkuses on lood palju keerulisemad. Ajatempli server TSS signeerib ajatempli y_n oma salajase võtmega K_{TSS}^{-1}, mis on kehtiv signeerimise hetkel t_n. Ajatempli väljastamisel võib kasutada järgmist meetodit.

Alice (A), soovides saada ajatemplit suurusele y, käivitab ajatempli serveriga TSS joonisel 1.1 toodud protokolli. Kunagi tulevikus, ajahetkel $\mathfrak{E}(K_{\mathrm{TSS}}^{-1})$, $\mathfrak{E}(K_{\mathrm{TSS}}^{-1}) > t_n$, muutub võti K_{TSS}^{-1} kehtetuks (kas tuleb avalikuks või täitub tema eluiga) ning avalikustakse tühistusnimekirjas. Ahela järgmised elemendid signeeritakse juba uue võtmega $K_{\mathrm{TSS}_1}^{-1}$. Ajahetkel t', mil kehtiv ajatempli serveri võti $K_{\mathrm{TSS}'}^{-1}$, on tarvitusel alates hetkest $\mathfrak{B}(K_{\mathrm{TSS}'}^{-1})$, soovitakse veri-

fitseerida hetkel x signeeritud andmeid. Tol ajahetkel saab suvaline osapool signeerida suvalisi andmeid ajatempli serveri suvalise **enne** võtit $K^{-1}_{\mathrm{TSS}'}$ tarvitusel olnud võtmega.

Lähtudes eeltoodust, liigutakse ajatempli verifitseerimiseks mööda ajatemplite ahelat tagurpidi, alustades mõnest ajahetkest $\mathfrak{B}(K^{-1}_{\mathrm{TSS}'})$ *hiljem* väljastatud ajatemplist ning lõpetades ajatempliga x, kontrollides kõikide vahepealsel teel esinevate sõnumilühendite korrektsust (joonis 1.2).



Joonis 1.2: Liikumine mööda ajatemplite ahelat

Kui kasutada protokollis toodud linkimisviisi, kus iga element sisaldab räsi eelmise elemendi linkinformatsioonist L_{n-1}, siis tuleb ajatemplite vahemiku $[m, n]$ läbimisel verifitseerida lõigu pikkuse suhtes lineaarne arv sõnumilühendeid, reaalses situatsioonis (dokumente 10^7 või enam) kulub ühe ajatempli kehtivuse kontrolliks lineaarse ahela puhul liiga kaua aega. On selge, et kõik teised moodused ahela kokku linkimiseks, kus L_n-s on vaid ühele eelmisele elemendile vastav sõnumilühend (st L_{n-1} asemel on $L_{n-f(n)}$ suvalise funktsiooni f, $f(n) \neq n-1$, jaoks), ei ühenda ajatemplite ahela kõiki elemente ning seega on antud kontekstis kasutud.

Tekib loomulik küsimus, kas on võimalik ajatemplite verifitseerimiseks kuluvat aega oluliselt vähendada, konstrueerides ahela, milles tagurpidi liikumine võtab lineaarsest väiksema aja (näiteks $\sqrt{x}$ või $\log x$), kui linkida iga ajatempel otseselt $r > 1$ erineva eelneva ajatempliga. Tekkivatele probleemidele pakume lahenduse peatükis 3.

1.2 Kasutatavad primitiivid

Lahendanud küsimuse ajatemplite linkimisest, tuleb tähelepanu pöörata ajatemplite juures kasutatavatele krüptograafilistele primitiividele. Konkreetselt kasutatakse ajatempli serveris digitaalsignatuure ning räsifunktsioone. Seda, kuidas toimida signatuuride kehtivuse kaotamisel me juba vaatasime. Lühike ülevaade olemasolevatest räsifunktsioonidest ning räsifunktsioonide

vastu mõeldud rünnetest on toodud peatükis 4. Peatükis 5 pakume välja konkreetse lahenduse, mida teha siis, kui räsifunktsioon lahti murtakse.

1.3 Ajatempli serverite hierarhia

Tekkiva koormuse (signeerimispäringutele vastamine, ajatemplite alleshoidmine) jaotamiseks on mõistlik tekitada teatav ajatemplite server hierarhia, kus on olemas üks tipmine server ning temale alluvad serverid. Hierarhia moodustamise problemaatikale on pühendatud peatükk 6, kus käsitletakse muuhulgas ka erinevate serverite turvapoliitikaid.

1.4 Viited senitehtule

Üks ülevaatlikumaid ilmunud materjale on Belgia TIMESEC töörühma aruanne [MQ97]. Ajatempli teenuse osutamisega seonduvaid formaate on püütud standardiseerida ([AZ97], [ACPZ97]). Reaalselt pakuvad ajatempli teenust Surety Technologies (`http://www.surety.com`) ning I. T. Consultancy Limited (`http://www.itconsult.co.uk/stamper.htm`). Viimane ajatempliteenus (nagu ka `http://www.eskimo.com/ weidai/timestamp.html`) põhinevad PGP-l. USA-s on registreeritud vähemalt neli ajatemplialast patenti ([HS92a], [HS92b], [HS94], [HS95]), millest ükski ei paku aktsepteeritavat lahendust.

Peatükk 2

Ajatempli protokoll

Sissejuhatuses oli toodud ajatempli süsteemides kasutatav protokoll (joonis 1.1) ning seletatud, miks on vaja ajatempleid arhiveerida ning neid ahelasse siduda. Järgnevas vaatleme ajatempli protokolli lähemalt.

Protokolli algatab suvaline ajatempli serveris sertifitseeritud klient. Protokoll algab kliendi autentimisega serveri poolt, kasutades mõnda turvalist autentimisprotokolli (vt peatükk 6). Seejärel esitab klient serverile tembeldamiseks sõnumi y_n (antud protokolli samm peab olema turvaliselt seotud autentimisprotokolli eelmiste sammudega. Joonisel 1.1 tähistasime identivat osa lihtsalt kui ID_n), millele server lisab ajatempli ning saadab signeeritud kujul tagasi kliendile. Serveri vastus on kujul

$$\{n, t_n, y_n, \mathrm{ID}_n, L_{n-1};\ L_n\}_{K_{\mathrm{TSS}}^{-1}}.$$

Sõnumi signeerimine ajatempli serveri poolt võimaldab teatud aja (signatuuri eluea) jooksul tõestada, et sõnum oli öeldud antud serveri poolt. Signeeritud andmetest on ajatempli hilisemaks verfitseerimiseks vajalikud

1. y_n kui tembeldatud sõnum,
2. t_n kui tembeldamisaeg ning
3. ID_n kui tembeldamist soovinud osapoole identifikaator.

Kolmiku $(t_n, y_n, \mathrm{ID}_n)$ olemasolu on vajalik väitmiseks, et keegi (ID_n) on millalgi (t_n) midagi (y_n) öelnud.

Sissejuhatuses oli räägitud vajadusest ajatempleid ahelasse linkida. Ahela läbimise jaoks on soovitatav linkimisprotseduuri otsene sõltuvus ajatempli

järjekorranumbrist n, mis peab sisalduma seega ilmutatud kujul ajatemplis endas. Võimaldamaks täielikku verifitseerimist, peab ajatempel sisaldama ka linkimisinformatsiooni L_n, mille põhjal oleks kindlaks tehtav ahela eelmine element, muuhulgas ka eelmise elemendi linkimisinformatsioon L_{n-1} (peatükis 3 vaadeldakse ajatemplite ahelate kiiremat läbimist võimaldavaid linkimisviise, kus L_n sisaldab informatsiooni mitme erineva eelneva ajatempli kohta), mille üldkuju on järgmine:

$$L(n) = H(n, t_n, y_n, \mathrm{ID}_n, L_{n-1}),$$

kus H on teatud omadusi rahuldav funktsioon. Samuti peab H rahuldama järgmist omadust: on raske leida paari $x \neq y$, mille korral $H(x) = H(y)$. Peale selle peab H olema nn sõnumilühendifunktsioon, mille korral funktsiooni tulemi $H(x)$ pikkus ei sõltu argumendist x (vastasel korral pikeneksid ajatemplid piiramatult). Vaadeldatud omadusi rahuldavaid funktsioone nimetatakse räsifunktsioonideks, räsifunktsioone käsitletakse pikemalt peatükis 4.

Peatükk 3

Ajatemplite linkimisviisid

Käesolevas paragrahvis vaatleme ajatemplite erinevaid linkimisviise, eesmärgiga leida praktikas vastuvõetav (eelkõige, kiire) meetod ahelas suvalise kahe punkti vahel liikumiseks.

Definitsioon 1 *Kui punktist n viib link punkti m, $n \geq m$, kirjutame $n \to m$. Lingi $n \to m$* pikkuseks *nimetame lõigu $[m, n]$ pikkust.*

Olgu antud ajatemplite ahel ning funktsioon $F(m, n) : \mathbb{Z}^2 \to \{\bot, \top\}$, $0 \leq m \leq n$, nii et link $n \to m$ leidub parajasti siis kui $F(m, n) = \top$. Olgu m ja n, $m < n$, kaks naturaalarvu. Defineerime järgmised suurused:

1. $\mathrm{lw}(m, n)$ *(*lühima tee pikkus *arvust m arvuni n üle linkide).*

2. $\ell(m, n) := \max_{n_1 - m_1}\{m \leq m_1 \leq n_1 \leq n \wedge F(m_1, n_1) = \top\}$ *(pikim link lõigus $[m, n]$).*

Märkus 1 *Suurused* $\mathrm{lw}(m, n)$ *ja* $\ell(m, n)$ *sõltuvad konkreetsest linkimisviisist. Igas järgnevas paragrahvis toome erineva linkimisviisi ning* $\mathrm{lw}(m, n)$ *ja* $\ell(m, n)$ *vastavad siis antud konkreetsele linkimiviisi suurusele.*

Antud peatükis vastame sellele küsimusele, tõestades, et (edaspidi on logaritmi aluseks vaikimisi 2)

1. juhul $r = 2$ leidub linkimisviis, kus $\mathrm{lw}(m, n) \leq \frac{\lceil \log n \rceil^2 + \lceil \log n \rceil}{2}$ ning

2. juhul $r = \log n$ leidub linkimisviis, kus $\mathrm{lw}(m, n) \leq 1 + \log(n - m)$

3.1 Homogeenne ahel

3.1.1 Definitsioon

Definitsioon 2 *Olgu p mingi algarv ning $m \in \mathbb{Z}^+$. Kui $p^k \mid m$ ning $p^{k+1} \not| m$, siis kirjutame* $\mathrm{ord}_p\, m := k$. *Kui $p = 2$, jätame alaindeksi märkimata.*

Algoritm 1 *Defineerime*

$$f(n) := 2^{\mathrm{ord}\, n+1},$$

L_n on lingitud kahe elemendiga: L_{n-1} ning $L_{n-2^{\mathrm{ord}\, n+1}}$. Juhul $n < 2^{\mathrm{ord}\, n+1}$ (n on kahe aste) teist linki ei moodustata. Joonisel 3.1 on toodud näide lõplikust vahemikust.

Kuna $1 = 2^0$, on suvalise lingi pikkus kahe mingi aste.

3.1.2 Ülemtõke

Lemma 1 $\min_{m<n} \frac{n-m}{\ell(m,n)} < 4$.

Tõestus. Fikseerime suuruse $\ell(m,n) = 2^k$. Juhul $n - m > 2^{k+2} - 2$ leiduks vahemikus $[m,n]$ vähemalt kaks erinevat naturaalarvu x ja y, $x < y$, nii et $\mathrm{ord}\, x = \mathrm{ord}\, y = k$ ning $2^{k+1} \mid y - x$. Seega leiduks vahemikus $[m,n]$ link $y \to x$ pikkusega 2^k. Vastuolu, järelikult $n - m \leq 2^{k+2} - 2$. Siit saame

$$\min_{m<n} \frac{n-m}{\ell(m,n)} = \min_{m<n} \frac{2^{k+2}-2}{2^k} = \min_{m<n}(4 - \frac{1}{2^{k-1}}) < 4.$$

Kuna nii $n - m$ kui ka $\ell(m,n)$ on naturaalarvud, siis $\ell(m,n) \geq \frac{n-m}{4} + \frac{1}{4}$. □

Lemma 2 *Olgu $\ell(m,n) = 2^k$. Kui lõik $[m,n]$ sisaldab ainult ühte linki pikkusega 2^k, siis $n - m \leq 3 \cdot 2^k - 2$.*

Tõestus. Kui $n - m > 3 \cdot 2^k - 2$, siis on lõigus $[m,n]$ vähemalt kolm punkti, mille järk alusel 2 on $k - 1$. □

Joonis 3.1: Näiteahel

Näide 1 *Vaatame joonist (3.1). Juhul* $\ell(m,n) = 2^2 = 4$ *saame lemmast 1, et* $n - m \leq 2^4 - 2 = 14$. *See juht realiseerub lõigul* $[5, 19]$. *Lemmast 2 saame, et kui lõigus* $[p, q]$ *on ülimalt üks link pikkusega* 4, *siis* $q - p \leq 10$. *Antud juht on joonisel realiseeritud lõigul* $[23, 33]$.

Algoritm 2 (Ahne algoritm) *On antud lõik* $[m, n]$ *ning*

$$k := \log \ell(m, n).$$

Lemmast 1 saame, et $n - m \leq 2^{\ell(m,n)+2} - 2$.

1. *Kui* $n-m > 3\cdot 2^k - 2$, *siis leidub lõigus kaks järjestikust linki* $[p_1, p_2]$ *ning* $[p_2, p_3]$ *pikkusega* 2^k. *Olgu rekursiivselt* S_1 *punktide* n *ning* p_3 *vaheline tee ja* S_2 *punktide* p_1 *ning* m *vaheline tee. Tagasta vastuseks* $S_1 \to p_3 \to p_2 \to p_1 \to S_2$.

2. *Kui* $n-m \leq 3\cdot 2^k - 2$, *siis leidub lõigus täpselt üks link* $[p_1, p_2]$ *pikkusega* 2^k. *Olgu rekursiivselt* S_1 *punktide* n *ning* p_2 *vaheline tee ja* S_2 *punktide* p_1 *ning* m *vaheline tee. Tagasta vastuseks* $S_1 \to p_2 \to p_1 \to S_2$.

Olgu $f(t) := \max_m \mathrm{lw}(m, m + t)$. Lemmast 2 järeldub, et esimesel juhul saame ahne algoritmiga ülemtõkke (eeldusel, et f on monotoonselt kasvav)

$$f(t) \leq 2 + 2f(t/4) \Rightarrow f(t) = O(\sqrt{t}).$$

Lemmast 1 saame teise juhu asümptootilise ülemtõke

$$f(t) \leq 1 + 2f(t/3) \Rightarrow f(t) = O(t^{\log_3 2})$$

ehk kokku,

$$f(t) = O(t^{\log_3 2} + \sqrt{t}) = O(t^{\log_3 2}).$$

n	$\sum$	$=$
0	1	1
1	2	2
2	$1 + 2\,\mathrm{lw}(0, 2^0)$	3
3	$1 + 2\,\mathrm{lw}(1, 2^1)$	5
4	$1 + 2\,\mathrm{lw}(1, 2^2)$	7
5	$3 + 2\,\mathrm{lw}(1, 2^2)$	9
6	$3 + 2\,\mathrm{lw}(1, 2^3)$	13
7	$7 + 2\,\mathrm{lw}(1, 2^3)$	17

Joonis 3.2: $\mathrm{lw}(0, 2^k)$ homogeense ahela korral

Näide 2 *Ahne algoritmi kohaselt*

$$\mathrm{lw}(0, 2^{2k}) \leq 2 \cdot 2^k - 1$$
$$\mathrm{lw}(0, 2^{2k+1}) \leq 3 \cdot 2^k - 1$$

Seega, umbes $2 \cdot 10^6$ *dokumendi olemasoleku korral tuleb läbida umbes 3000 linki, pakutud lahendus ei ole piisavalt praktiline (*10^9 *dokumendi korral läbitaks üle 50000 lingi).*

On kerge näha, et ahne algoritm pole optimaalne. Tõsi, on selge, et kui optimaalse algoritmi poolt lõigus $[m, n]$ läbitavate linkide maksimaalne pikkus on 2^{k-r}, $k \geq r \geq 0$, siis läbib algoritm ka kõik teised selles lõigus olevad antud pikkusega lingid. Laskumata päris sügavustesse, leiame järgnevalt ülemtõkke suurusele $\mathrm{lw}(0, 2^n)$. Lähtudes eelnevast jutust, on optimaalse algoritmi kandidaatide poolt leitud tee pikkus $\mathrm{lw}(0, 2^n) := \min_i(2^{i+1} - 1 + 2\,\mathrm{lw}(0, 2^{n-2-i}))$. Vasak pool on minimaalseim siis, kui 2^{i+1} ning $2\,\mathrm{lw}(0, 2^{n-2-i}))$ on samas suurusjärgus, st $\mathrm{lw}(0, 2^n) = 4\,\mathrm{lw}(0, 2^{n-2-i}))$. Lahendades viimase võrrandi, saame et $\mathrm{lw}(0, 2^n) = 2^{2n/(i+2)}$. Suurused 2^{i+1} ning $2 \cdot 2^{(2n-4-2i)/(i+2)}$ on samas suurusjärgus parajasti siis kui $i \approx \sqrt{n}$.

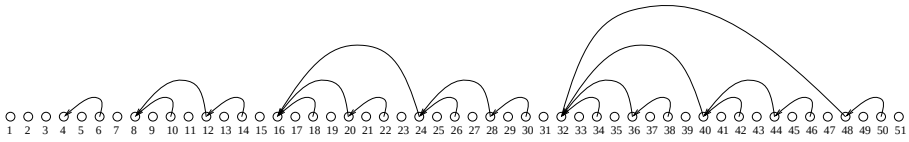
Juhul $i = \lfloor\sqrt{n}\rfloor$ saame, et $\mathrm{lw}(0, 2^n) = O(2^{2\sqrt{n}})$. Kui $n = 34$ ($2^{34} \approx 1.7 \cdot 10^{10}$), siis $\mathrm{lw}(0, 2^n) \approx 3300$. Võrreldes ahne algoritmi on aja kokkuhoid umbes kahekümnekordne, kuid siiski mitte piisav.

3.2 Binomiaalahel

Algoritm 3 *Olgu ahela esimene element seekord* 1 *(mitte* 0*). Ahela iga element* n *on ühendatud kahe elemendiga:* $n-1$ *ning* $n-f(n)$*, kus*

$$f(n) := 2^{\operatorname{ord} n}.$$

Juhul $f(n) = 1$ *või* $n - f(n) < 1$ *(*n *on paaritu või kahe aste) teist linki ei moodustata. Teiselt poolt, antud ahelat saab kirjeldada ka lähtudes linkide lõpppunktist. Punkti* m *sisenevad lingid punktidest* $m+1$*,* $m+2$*, . . . ,* $m + 2^{\operatorname{ord} m-1}$*. Lõik ahelast on kujutatud joonisel 3.3.*



Joonis 3.3: Binomiaalahel

Optimaalseim tee saadakse seekord ahne algoritmiga: leidakse pikim link $n \to n_1$, nii et $n_1 \geq m$ ning väljastatakse tee $n \to n_1 \to S$, kus S on lõigu $[m, n_1]$ lühim tee. Ahne algoritmi optimaalsus tuleb faktist, et antud ahela korral lingid ei ristu: kui jäetakse valimata pikim link, pikeneb tee automaatselt. Vastav algoritm on toodud joonisel 3.4. Siinjuures kasutame asjaolu, et suuruse $n - \operatorname{ord} n$ arvutamine on triviaalne: n-i kõige madalam ühene bitt tuleb nullida, bititasemel sobib selleks operatsioon $n_1 := n\&(n-1)$.

proc *findway*(m, n) $\equiv$
  $n_1 := n - 1$
  if $n\&n_1 \geq m$ **then** $n_1 := n\&n_1$ **fi**
  if $n_1 < m$ **then exit fi**
  print(n_1)
  findway(m, n_1)**.**

Joonis 3.4: Ahne algoritm binomiaalahela jaoks

Ülemtõkke tõestus käib järgmiselt:

1. Näitame et suvalise lõigu $[m, n]$ korral saab leida teise lõigu $[p, p + t]$, mille lühim tee on sama pikk ning kus sisaldub mingi kahe aste.

2. Anname valemi suuruse $\mathrm{lw}(p, n)$ arvutamiseks, juhul kui selles lõigus sisaldub kahe aste.

3. Tõestame saadud valemite jaoks ülemtõkked.

Lemma 3 *Olgu $[m, n]$ selline lõik, mis ei sisalda kahe astmeid ($\lfloor \log_2 m \rfloor = \lfloor \log_2 n \rfloor = x$). Siis $\mathrm{lw}(m, n) = \mathrm{lw}(m - 2^x, n - 2^x)$.*

Tõestus. Triviaalne, kuna lõigus $[m - 2^x, n - 2^x]$ olevate elementide järgud on võrdsed lõigu $[m, n]$ elementide järkudega ning seega leiduvad lõigus $[m - 2^x, n - 2^x]$ täpselt samad lingid mis lõigus $[m, n]$. □

Lemma 4 *Olgu $[m, n]$ lõik, mis ei sisalda kahe astmeid. Siis leiduvad arvud p ning q nii, et $\mathrm{lw}(m, n) = \mathrm{lw}(p, q)$, kus $n - m = q - p$, $p \leq m$ ning vahemikus $[p, q]$ on vähemalt üks kahe aste.*

Tõestus. Rakendame rekursiivselt lemmat 3, kuni saame viimaks kätte lõigu $[p, q]$, milles ei ole kahe astmeid. □

Definitsioon 3 *Olgu $n = \sum n_i 2^i$, $n_i \in \{0, 1\}$, arvu n kahendesitus. Arvu n Hammingi kaaluks $w_h(n)$ nimetatakse summat $\sum n_i$; analoogia põhjal tähistame summat $\sum i n_i$ sümboliga $w_g(n)$.*

Lemma 5 *Olgu $n > 0$. Siis $\mathrm{lw}(2^{\lfloor \log n \rfloor}, n) = w_h(n) - 1$.*

Tõestus. Ahne algoritm võtab n-ist järjest maha kõige vasakpoolsemaid bitte, kuni saab lõpuks kätte arvu $2^{\lfloor \log n \rfloor}$. Mahavõetud bitte on $w_h(n) - 1$. □

Lemma 6 *Olgu $m > 0$, $x = \lfloor \log m \rfloor$ ning $r := \mathrm{ord}\, m$. Siis*

$$\mathrm{lw}(m, 2^{x+1} - 1) = \frac{x^2 + 3x + 2}{2} - \frac{r^2 - r}{2} - w_g(m) - w_h(m)$$

Tõestus. Olgu $m = \sum im_i$. Vaatleme juhtu $r \geq 1$ (m on paaris). Siis

$$\begin{aligned}
\mathrm{lw}(m, 2^{x+1} - 1) &= \sum_{i=r-1}^{x} (1 - m_i)(i + 1) \\
&= \sum_{i=r}^{x+1} i - \sum_{i=r}^{x} m_i(i + 1) \\
&= \frac{x^2 + 3x + 2}{2} - \frac{r^2 - r}{2} - \sum_{i=r}^{x} m_i - \sum_{i=r}^{x} im_i \\
&= \frac{x^2 + 3x + 2}{2} - \frac{r^2 - r}{2} - w_h(m) - w_g(m).
\end{aligned}$$

Juhul $r = 0$ (m on paaritu)

$$\begin{aligned}
\mathrm{lw}(m, 2^{x+1} - 1) &= \sum_{i=0}^{x} (1 - m_i)(i + 1) \\
&= \sum_{i=0}^{x} (i + 1) - \sum_{i=0}^{x} m_i(i + 1) \\
&= \frac{x^2 + x}{2} + x + 1 - \sum_{i=0}^{x} m_i - \sum_{i=0}^{x} im_i \\
&= \frac{x^2 + 3x + 2}{2} - w_h(m) - w_g(m).
\end{aligned}$$

□

Lemma 7 *Olgu $[m, n]$ lõik, mis sisaldab vähemalt ühte kahe astet (st $x = \lfloor \log m \rfloor < \lfloor \log n \rfloor = y$). Siis*

$$\mathrm{lw}(m, n) = \frac{y^2 + y}{2} - \frac{r^2 - r}{2} + w_h(n) - w_h(m) - w_g(m).$$

Juhul $\lceil \log m \rceil = \log m$ (m on kahe aste) kehtib

$$\mathrm{lw}(m, n) = \frac{y^2 + y}{2} - \frac{x^2 + x}{2} + w_h(n) - 1.$$

Sealjuures,

$$\mathrm{lw}(1, n) = \frac{y^2 + y}{2} + w_h(n) - 1.$$

Tõestus. Ahne algoritm läbib teekonna moodustamisel kõiki lõigus $[m, n]$ olevaid kahe astmeid. Seega

$$\begin{aligned}
\mathrm{lw}(m,n) =& \mathrm{lw}(m, 2^{x+1}) + \mathrm{lw}(2^{x+1}, 2^y) + \mathrm{lw}(2^y, n) \\
=& \mathrm{lw}(m, 2^{x+1}) + \sum_{i=x+2}^{y} i + w_h(n) - 1 \\
=& \mathrm{lw}(m, 2^{x+1}) + \frac{y^2+y}{2} - \frac{(x+1)^2+x+1}{2} + w_h(n) - 1 \\
=& \mathrm{lw}(m, 2^{x+1}-1) + 1 + \frac{y^2+y}{2} - \frac{x^2+3x+2}{2} + w_h(n) - 1 \\
=& \frac{x^2+3x+2}{2} - \frac{r^2-r}{2} - w_g(m) - w_h(m) + 1 \\
&+ \frac{y^2+y}{2} - \frac{x^2+3x+2}{2} + w_h(n) - 1 \\
=& \frac{y^2+y}{2} - \frac{r^2-r}{2} + w_h(n) - w_h(m) - w_g(m).
\end{aligned}$$

Kui $m = 2^x$, siis $\mathrm{lw}(m, 2^{x+1}) = x + 1$. Kui $m = 1 = 2^0$, siis $x = 0$. □

Lemma 8 *Olgu $m \geq 1, t \geq 0, n \geq m$ ning $t := \lfloor \log(m - n + 1) \rfloor$. Siis*

$$\mathrm{lw}(m,n) \leq \frac{\lceil \log(n-m) \rceil^2 + \lceil \log(n-m) \rceil}{2}.$$

Tõestus. Lemma 4 tõttu võime eeldada, et vahemikus $[m, n]$ on vähemalt üks kahe aste. Olgu $x := \lfloor \log m \rfloor$ ning $y := \lfloor \log n \rfloor$ ning $t := \lfloor \log(n-m) \rfloor$. On vaja näidata, et

$$\mathrm{lw}(m,n) \leq \frac{\lfloor \log(n-m) \rfloor^2 + 3\lfloor \log(n-m) \rfloor + 2}{2}.$$

Vaatleme eraldi järgmisi juhte.

1. $m = 1$. Siis lemmast 7 saame, et

$$\mathrm{lw}(m,n) = \frac{\lfloor \log(n-m+1) \rfloor^2 + \lfloor \log(n-m+1) \rfloor}{2} + w_h(n) - 1.$$

Kui $\lfloor\log(n-m+1)\rfloor = \lfloor\log(n-m)\rfloor$, siis

$$\begin{aligned}\mathrm{lw}(m,n) \leq & \frac{\lfloor\log(n-m+1)\rfloor^2 + 3\lfloor\log(n-m+1)\rfloor}{2} - 1 \\ = & \frac{\lfloor\log(n-m)\rfloor^2 + 3\lfloor\log(n-m)\rfloor}{2} - 1.\end{aligned}$$

Kui $\lfloor\log(n-m+1)\rfloor \neq \lfloor\log(n-m)\rfloor$, siis $\lfloor\log(n-m+1)\rfloor = \lfloor\log(n-m)\rfloor + 1$ ning $n = 2^y$. Sel juhul

$$\begin{aligned}\mathrm{lw}(m,n) = & \frac{\lfloor\log(n-m)\rfloor^2 + 3\lfloor\log(n-m)\rfloor + 2}{2} + w_h(2^z) - 1 \\ = & \frac{\lfloor\log(n-m)\rfloor^2 + 3\lfloor\log(n-m)\rfloor}{2} + 1.\end{aligned}$$

2. Lõigus $[m,n]$ on parajasti 1 kahe aste ning $m \neq 1$.

   Kui $m = 2^x$, siis $\mathrm{lw}(m,n) = H(n) - 1 \leq \lfloor\log(n-m)\rfloor$, tõestatud.

   Kui m ei ole kahe aste, siis $y = x+1$. Kui $x < 2$, siis teoreem kehtib, olgu $x \geq 2$. Olgu arvu n kahendesitus $1 + 0^\ell + \alpha$, $0 \leq \ell \leq x$. Siis $\mathrm{lw}(2^{x+1}, n) = H(n) - 1 \leq x - \ell$. Samal ajal kehtib ka $x - \ell = \lceil\log(n - (2^{x+1}-1))\rceil \leq \lfloor\log(m-n)\rfloor + 1$.

   Olgu arvu m kahendesitus $1^{\ell_2} + \beta$, kus $1 \leq \ell_2 \leq k$. Kui $\beta \neq \lambda$, siis β algab nulliga. Kuna $\mathrm{lw}(m, 2^{x+1}) = \mathrm{lw}(\beta, 2^{x+1-\ell_2}) \leq \frac{(x+1-\ell_2)^2+x+1-\ell_2}{2} - 2$ ning $\ell_2 := x + 1 - \lfloor\log(2^{x+1} - m)\rfloor$, siis $x + 1 - \ell_2 = \lfloor\log(2^{x+1} - m)\rfloor \leq \lfloor\log(m-n)\rfloor$. Seega antud juhul kehtib

$$\mathrm{lw}(m,n) \leq \frac{\lfloor\log(m-n)\rfloor^2 + 3\lfloor\log(m-n)\rfloor}{2} - 1.$$

3. Kolmandaks, lõigus $[m,n]$ on $s \geq 2$ kahe astet ($y = x+s$) ning $m \neq 1$.

Sellisel juhul $y \in \{t, t+1\}$. Kui $y = t$, siis

$$\begin{aligned}
\mathrm{lw}(m,n) =& \frac{y^2+y}{2} - \frac{r^2-r}{2} + w_h(n) - w_h(m) - w_g(m) \\
\leq& \frac{y^2+y}{2} - \frac{r^2-r}{2} + y - w_h(m) - w_g(m) \\
=& \frac{t^2+3t}{2} - \frac{r^2-r}{2} - w_h(m) - w_g(m) \\
\leq& \frac{t^2+3t}{2} - \frac{r^2-r}{2} - w_h(m) - w_g(m) \\
\leq& \frac{t^2+3t}{2} - 1.
\end{aligned}$$

Kui $y = t+1$, siis $n < 2^y + m$. Kuna $m \leq 2^{x+1} - 1$, siis $2^y \leq n \leq 2^y + 2^{x+1} - 2$ ning seega $w_h(y) \leq x$. Kui m on kahe aste ($r = x$), siis $w_h(n) - w_g(m) \leq x - x = 0$. Kui m ei ole kahe aste, siis $w_h(n) - w_g(m) \leq x - (x+r) = -r$, seega

$$\begin{aligned}
\mathrm{lw}(m,n) =& \frac{y^2+y}{2} - \frac{r^2-r}{2} + w_h(n) - w_h(m) - w_g(m) \\
\leq& \frac{y^2+y}{2} - \frac{r^2-r}{2} - w_h(m) \\
=& \frac{t^2+3t+2}{2} - \frac{r^2-r}{2} - w_h(m) \\
\leq& \frac{t^2+3t}{2},
\end{aligned}$$

sest $w_h(m) \geq 1$.

Tõestatud. □

Näide 3 *Juhul $n = 10^{10}$ on tee pikkus umbes 600 (sõltuvalt räsifunktsiooni kasutusviisist tuleb tee iga lingi läbimisel teha $1 \ldots 2$ kontrolli). Juhul $n = 10^{11}$ on tee pikkus umbes 700. Arvestades, et $10^{11} \approx 2^{36.5}$, võib sellise andmemahu juures kitsaskoht juba kuskilt mujalt (mälu maht, ...) tulla.*

Märkus 2 *Siin artiklis toodud ahelate ilu on nende homogeensuses. Binomiaalahelale saab panna lisalinke (näiteks $2^k \to 2^{k-1}$), mis teatud olukordades kiirendaksid ahela läbimist, kuid ülemtõkke jätaksid paigale.*

Võrreldes eelmise ahelaga on binomiaalahela heaks omaduseks ka see, et antud ahela teatud elementide verifitseerimise tõenäosus on suvalise ajatempli verifitseerimise protsessis tunduvalt suurem kui teistel. Neid nn sõlmpunkte, mille korral $\operatorname{ord} n$ on suur, on väga tulus hoida pidevalt vahemälus, mis võimaldab veelgi kiirendada ahela läbimist.

3.3 Logaritmilise arvu linkidega ahel

Algoritm 4 (Logaritmiline arv linke) *Olgu antud arv n. Ahela elemendist n väljuvad lingid $n - 2^0$, $n - 2^1$, ... , $n - 2^{\lfloor \log n \rfloor}$.*

Lemma 9 $\operatorname{lw}(m, n) \leq 1 + \log(n - m)$, $m < n$.

Tõestus. Olgu $k := \lfloor \log(n-m) \rfloor$. Linkide konstruktsiooni põhjal leidub link $n \to n_1$, kus $n_1 = n - 2^k$; k valiku tõttu $n - 2^{k+1} < m$ ning seega oleme ühe lingiga läbinud rohkem kui poole vahemaast $n - m$ (ehk $2(n_1 - m) < n - m$). Induktsiooni baas: $\operatorname{lw}(m, m + 1) = 1 \leq 1 + \log 1$. Induktsiooni eelduse tõttu $\operatorname{lw}(m, n_1) \leq 1 + \log(n_1 - m)$, seega

$$\begin{aligned} \operatorname{lw}(m, n) &\leq 2 + \operatorname{lw}(m, n_1) \leq 2 + \log(n_1 - m) \\ &= 1 + \log 2(n_1 - m) < 1 + \log(n - m) \end{aligned}$$

□

Optimaalseim tee saadakse alati kätte ahne algoritmiga, mis vahemiku $[m, n]$ korral väljastab rekursiivselt tee $n \to n_1 \to S$, kus $n_1 := n - 2^{\lfloor \log(n-m) \rfloor}$ ning S on lühim tee lõigus $[m, n_1]$. Aega kulub logaritm vahemikust $n - m$, kuid iga ahela element n sisaldab viidet $\log n$ eelnevale elemendile.

Näide 4 *Juhul $n = 10^{10}$ on elemendiga n seotud 33 elementi. Mingi eelneva elemendini jõudmiseks tuleb läbida ülimalt 34 linki. Juhul $n = 10^{11}$ elemendiga n seotud 36 linki ning tuleb läbida ülimalt 37 linki (ning verifitseerida ülimalt $36 \cdot 37$ sõnumilühendit).*

3.4 Veel üks ahel

Algoritm 5 *Olgu n üks element ahelas. Seostame n-i elementiga $n - 1$ ning elementidega $\lceil \frac{n}{\log n} \rceil$, $\lceil \frac{n}{\log^2 n} \rceil$, ... , $\lceil \frac{n}{\log^k n} \rceil$ kus k on suurim arv, mille korral $\frac{n}{\log^k n} > \log n$ ehk $k := \lfloor \frac{\log n}{\log \log n} \rfloor - 1$.*

Seega on linke kokku $\lfloor \frac{\log n}{\log \log n} \rfloor$.

Lemma 10 $\mathrm{lw}(m, n) = O(\log n - \log m)$.

Peatükk 4

Räsifunktsioonid

4.1 Motivatsioon

Peatükis 2 oli sõnastatud paar linkimisinformatsioonis kasutatavale funktsioonile H kohustuslikku omadust ja öeldud, et neid nõudeid rahuldavaid funktsioone nimetatakse räsifunktsioonideks.

Definitsioon 4 *Paari (x, x'), kus $x \neq x'$, kuid $H(x) = H(x')$ nimetame räsifunktsiooni H kollisiooniks.*

Definitsioon 5 *Kui etteantud suuruse x jaoks on arvutuslikult raske leida suurust $x' \neq x$ nii, et $H(x) = H(x')$, nimetame räsifunktsiooni H nõrgalt kollisioonivabaks argumendi x suhtes.*

Definitsioon 6 *Kui on raske leida suurusi x ja x', et $x \neq x'$ ja $H(x) = H(x')$, nimetame räsifunktsiooni H rangelt kollisioonivabaks.*

Definitsioon 7 *Räsifunktsiooni H nimetame ühesuunaliseks, kui antud suuruse z jaoks on raske leida sellist suurust x, et $H(x) = z$.*

On lihtne tõestada, et räsifunktsiooni rangest kollisioonivabadusest järeldub nõrk kollisioonivabadus suvalise argumendi x jaoks ning ka ühesuunalisus.

Vastavalt kollisioonivabaduse kahele määratlusele võib rääkida ka räsifunktsiooni murdmisest kahes mõttes, st olukorrast, kus varem raskeks peetud vastavate suuruste leidmise ülesanne muutub kergeks.

4.2 Räsifunktsioonide tüübid

Räsifunktsioonide väljatöötamisel lähtutakse tavaliselt ühest alltoodud kahest ideest.

1. Paljude krüptograafiliste primitiivide turvalisus põhineb teatud (arvutuslikult) raskeks peetavate ülesannete kasutamisel. Sellisteks ülesanneteks kvalifitseeruvad näiteks suurte arvude teguriteks lahutamine ning diskreetse logaritmi leidmine, mille jaoks on juba sajandeid püütud kiireid algoritme leida, kuid senini edutult. Loodetakse, et sellised algoritme polegi või vähemasti ei leita neid lähitulevikus. Nii konstrueeritud räsifunktsioone iseloomustab kõrge turvalisus, kuid samas vähene efektiivsus.

2. Oluliselt kiiremad on bititeisendustel põhinevad räsifunktsioonid. Nende idee on teisendada sisendit bititasemel, kasutades erinevaid kahendloogikast tuntud operatsioone. Vastava funktsiooni arvutamise kiirus saavutatakse just tänu võimalusele selliseid operatsioone arvutis edukalt teostada. Samas ei saa me kindlad olla saadavate räsifunktsioonide turvalisuses — osalt seetõttu, et bitioperatsioonide kaudu funktsioone konstrueerides on raske ette näha saadavate funktsioonide kõiki omadusi, teisalt aga seetõttu, et nende funktsioonide süstemaatiline teoreetiline käsitlus pole veel piisaval määral arenenud.

4.2.1 Rasketel ülesannetel põhinevad räsifunktsioonid

Chaum-van Heijst-Pfitzmanni räsifunktsioon

Üks tuntumaid rasketel ülesannetel põhinevaid räsifunktsioone on **Chaum-van Heijst-Pfitzmanni räsifunktsioon** ([CvHP92]). Selle konstrueerimiseks valitakse suur algarv p nii, et ka $q = \frac{p-1}{2}$ on algarv; lisaks valitakse korpuse $\mathbb{Z}_p$ kaks primitiivset elementi α ja β. Funktsioon H_{CHP} defineeritakse võrdusega

$$H_{CHP}(x_1, x_2) = \alpha^{x_1}\beta^{x_2} \mod p.$$

Saab tõestada järgmise teoreemi.

Teoreem 1 *Kui kirjeldatud funktsiooni H jaoks õnnestub leida kollisioon (st kaks argumendipaari $(x_1, x_2) \neq (x_3, x_4)$, et $H_{CHP}(x_1, x_2) = H_{CHP}(x_3, x_4)$), siis võib efektiivselt leida diskreetse logaritmi $\log_\alpha \beta$.*

Kuna diskreetse logaritmi leidmine on üks tuntumaid klassikalisi raskeks peetavaid ülesandeid, mille lahendamiseks pole efektiivset eeskirja leitud sajandeid (alates probleemi püstitamisest), siis võib Chaum-van Heijst-Pfitzmanni räsifunktsiooni pidada küllaltki turvaliseks. Samas tuleb lõivu maksta arvutuste kiirusele, põhilise aja nõuab astendamine suure mooduli p järgi.

Goldreich-Goldwasser-Halevi räsifunktsioon

Goldreich-Goldwasser-Halevi räsifunktsiooni (vt [GGH96]) konstrueerimiseks valitakse kõigepealt juhusliku $n \times m$ maatriksi M üle ringi $\mathbb{Z}_q$. Olgu antud sisendbitijada $s_1 s_2 \dots s_m \in \{0,1\}^m$, räsifunktsiooni H väärtus sellel avaldub kujul

$$H_{GGH}(s) = Ms \mod q = \sum_{i=1}^{m} s_i M_i,$$

kus M_i tähistab maatriksi M i-ndat veergu.

Selle räsifunktsiooni turvalisus tugineb võre mõistele eukleidilises ruumis.

Definitsioon 8 *Olgu antud hulk $B = \{b_1, \dots, b_n\}$, mis koosneb n-ist lineaarselt sõltumatust vektorist ruumis $\mathbb{R}^n$. Hulgale B vastavaks võreks nimetatakse hulka*

$$L(B) = \left\{ \sum_{i=1}^{n} a_i b_i : \forall\ i\ a_i \in \mathbb{Z} \right\}.$$

Hulka B nimetatakse võre $L(B)$ baasiks.

Vaatleme selles võres järgmist ülesannet. Olgu võre L jaoks antud mingi baas $B = \{b_1, \dots, b_n\}$. Leida võrest selline hulk $C = \{c_1, \dots, c_n\}$, mis koosneb lineaarselt sõltumatutest vektoritest ja mille korral hulga C pikkus

$$|C| = \max\{c_1, \dots, c_n\}$$

on ülimalt polünomiaalselt (n järgi) suurem kui lühima n-elemendilise lineaarselt sõltumatu vektorite hulga pikkus.

Hetkel parim teadaolev seda ülesannet lahendav algoritm töötab ajas $(1+\epsilon)^n$, kus ϵ on suvaline fikseeritud positiivne arv. Saab tõestada, et kollisiooni leidmine funktsiooni H_{GGH} jaoks annab selle ülesande lahenduseks

efektiivse meetodi. Järelikult võib ka funktsiooni H_{GGH} hetkel piisavalt turvaliseks pidada.

Funktsiooni H_{GGH} eeliseks funktsiooni H_{CHP} ees on tema suhteliselt suur kiirus (ainus vajalik operatsioon on modulaarne liitmine). Mõlema räsifunktsiooni ühiseks puudujäägiks on räsitava bitijada fikseeritud pikkus, kuid leidub protseduur suvalise räsifunktsiooni iteratiivseks laiendamiseks suvalise pikkusega sisenditele kaotamata seejuures turvalisust.

4.2.2 Bititeisendustel põhinevad räsifunktsioonid

Ka seda tüüpi räsifunktsioonid põhinevad iteratiivsel tehnikal, kus teatud operatsiooni rakendatakse sisendi mingile osale, siis lisatakse sisendist uus osa ja korratakse sama operatsiooni analoogiliselt sisendi lõpuni.

Formaalsemalt, olgu räsifunktsiooni sisend M, jagame selle etteantud pikkusega plokkideks $M_1, \ldots, M_n$ (lisades vajadusel liigseid bitte, kui M pikkus ei jagu ploki pikkusega). Samuti olgu antud nn *raundifunktsioon* f, mille abil arvutatakse $i = 1, \ldots, n$ jaoks suurused

$$X_{i+1} = f(X_i, M_i).$$

Funktsiooni f rakendamiseks tuleb fikseerida algväärtus $X_1 = IV$, protsessi väljundiks loeme väärtust X_{n+1}.

Ühe raundi arvutamine funktsiooni f abil seisneb tavaliselt teatud hulga bitioperatsioonide (Boole'i funktsioonid) sooritamises. Tänu efektiivsele realiseeritavusele riistvara tasemel on seda liiki räsifunktsioonid suure töökiirusega.

Bititeisendustel põhineva räsifunktsiooni turvalisus sõltub oluliselt valitud raundifunktsiooni turvalisusest. Näiteks kollisioon räsifunktsiooni i-ndas raundis (sellised $M_i \neq M_i'$, et $f(X_i, M_i) = f(X_i, M_i')$) annab kohe kollisiooni kogu räsifunktsioonile ([Sti95]). Vahel piisab ka nõrgemast, nn pseudokollisioonist, ehk paaridest $(X_i, M_i) \neq (X_i', M_i')$, mille korral $f(X_i, M_i) = f(X_i', M_i')$, et kollisioonini jõuda.

Samuti nõrgendavad räsifunktsiooni tunduvalt raundifunktsiooni püsipunktid, st suurused X_i, mille korral $X_i = f(X_i, M_i)$. Nende ilmnemisel võib sisendis bloki M_i vahetada suvalise bloki M_i' vastu.

Seda liiki räsifunktsioonide turvalisuse tõstmiseks on välja töötatud rida heuristilisi nõudeid nende räsifunktsioonide disainile. Iga väljundbitt peaks sõltuma kõikidest sisendbittidest ning iga sisendbiti muutus peab võrdse

tõenäosusega esile kutsuma muutuse mistahes väljundbitis, et hoida ära sisend-väljundvoogude võrdlevale statistilisele analüüsile tuginevaid ründeid. Samas on arusaadav, et diskreetses maailmas ning piiratud ressurssidega (funktsiooni f ei saa arvutada liiga kaua) pole ühtlast jaotust võimalik saavutada. Tänu sellele ei õnnestu kunagi täielikult vältida statistilisel analüüsil põhinevaid ründeid. Näiteks kulus suhteliselt levinud räsifunktsiooni MD4 murdmiseks umbes 5 aastat. On pakutud mitmeid MD4 parandusi (MD5, RIPEMD, RIPEMD-160, SHA-1), kuid arvestades räsifunktsioonide konstrueerimise hetketeadmust, aga samas ka räsifunktsioonide krüptoanalüüsi suhteliselt kiiret arengut, ei saa ühtegi olemasolevat bititeisendustel põhinevat räsifunktsiooni pidada pikaajaliselt turvaliseks.

4.3 Räsifunktsioonide ründamise meetodid

4.3.1 Räsifunktsioonist sõltumatud ründed

Oletades, et räsifunktsioon on pseudojuhuslik, võib ründaja kasutada mõningaid üldisi meetodeid kollisioonide leidmiseks. Need meetodid ei sõltu konkreetse räsifunktsiooni leidmise algoritmist, vaid ainult sõnumilühendi pikkusest.

Sünnipäevarünne

Ründeidee põhineb nn sünnipäevaparadoksil, mis seisneb selles, et juhuslikult valitud vähemalt 23-st inimesest koosnevas rühmas leidub tõenäosusega $> 50\%$ kaks liiget, kes peavad sünnipäeva samal päeval. Mistahes r-bitise väljundiga räsifunktsiooni abil arvutatud erinevate sõnumilühendite arv on $n = 2^r$.

Kui võtta kaks (juhuslikult genereeritud) sõnumilühendite hulka võimsustega vastavalt x_1 ja x_2, siis tõenäosust, et need hulgad sisaldavad ühiseid elemente, võib hinnata järgmiselt:

$$P \approx 1 - e^{-\frac{x_1 x_2}{n}}.$$

Seega kui x_1 ja x_2 on mõlemad suurusjärgus $\sqrt{n}$, siis ühiste elementide leidumise tõenäosus on umbes $1 - \frac{1}{e} \approx 0,63$. Lisaks sellele muudab vaadeldavate sõnumilühendite hulkade võimsuste väike suurendamine ühise elemendi leidmise tunduvalt tõenäolisemaks.

Praktikas püüab ründaja esitada omakoostatud sõnumit "õige" sõnumi asemel. Kui "õigeid" sõnumeid on x_2 tükki, siis tõenäosus, et "vale" sõnumi lühend langeb kokku mõne "õige" sõnumi lühendiga, on umbes $1 - e^{-\frac{x_2}{n}}$.
Selle ründe ebaefektiivseks muutmiseks peaks räsifunktsioonist saadav sõnumilühend olema vähemalt 128, soovitavalt isegi 160 bitti pikk.

Juhuslik rünne Selle ründe puhul valib ründaja mingi sõnumi või selle osa ja loodab, et vastav sõnumilühend langeb kokku "õige" sõnumi lühendiga. Kui sõnumilühendi pikkus on r bitti, siis selle ründe õnnestumise tõenäosus $P = \frac{1}{2^r}$.
Juhusliku ründe ebaefektiivseks muutmiseks peaks räsifunktsioonist saadav sõnumilühend olema vähemalt 64 bitti pikk.

Toodud rünnete õnnestumise tõenäosust saab suurendada, leides eelnevalt suure hulga sõnumite lühendid.

4.3.2 Algoritmist sõltuvad rünne

Saame-keskel-kokku-rünne

See variatsioon sünnipäevaründest on rakendatav pööratavat raundifunktsiooni kasutavate räsifunktsioonide vastu ning lubab ründajal konstrueerida etteantud lühendiga sõnumi. Ründaja valib sõnumi esimese poole jaoks x_1 ja tagumise poole jaoks x_2 võimalikku väärtust. Seejärel liigub ta oletatavast sõnumist edasi ja sõnumilühendi väärtusest tagasi; tõenäosus, et keskmised väärtused langevad kokku, on

$$P \approx 1 - e^{-\frac{x_1 x_2}{2^r}},$$

kus r on sõnumilühendi pikkus.

Plokiparandusrünne

Siin kasutab ründaja olemasolevat teate ja sõnumilühendi paari, üritades muuta sõnumi ühte või enamat plokki nii, et sõnumilühend ei muutuks.

Selle ründe ebaefektiivseks muutmiseks tuleks saavutada sõnumilühendi iga biti tugev sõltuvus algteksti igast bitist.

Kasutatava krüptosüsteemi nõrkustel põhinevad ründed

Plokkšifrite teatavad omadused, mis šifreerimisel ei oma praktilist tähendust, võivad oluliselt nõrgendada sellel šifril baseeruvat räsifunktsiooni. Niisugused omadused on näiteks:

- Täiendiomadus, s.t. $Y = f(X, M) \Leftrightarrow \bar{Y} = f(\bar{X}, \bar{M})$, kus $\bar{X}$ tähistab X-i täiendit bitthaaval. Seljuhul on lihtne leida sõnumite paare, mille sõnumilühendid erinevad ettemääratud viisil.

- Nõrgad võtmed, s.t. olukord, kus $f(f(X, M), M) = X$ kõigi X-de korral. See omadus, nii nagu ka

- püsipunktid – $f(X, M) = X$ – lubavad leida raundifunktsiooni püsipunkte.

- Võtmekollisioonid – $f(X, M_1) = f(X, M_2)$ – aitavad leida raundifunktsiooni kollisioone.

Erinevuste krüptoanalüüs

Selle ründe puhul otsitakse statistilisi korrelatsioone räsifunktsiooni sisendi ja väljundi vahel, st kuidas muudab mõne sisendbiti vahetamine väljundit. Räsifunktsioonide korral huvitab meid erijuht, kus algteksti muutmisel jääb sõnumilühend samaks.

Erinevuste krüptoanalüüs on rakendatav peaaegu kõigile krüptosüsteemidele ning tänu pidevalt arenevale uurimistööle selles vallas pole tema vastu just kerge midagi konkreetset ette võtta. See ongi põhimeetod, tänu millele ei saa bititeisendustel põhinevaid räsifunktsioone väga kaua usaldada.

Lineaarne krüptoanalüüs

Jaapani krüptoloogi Matsui poolt 1993. aastal välja pakutud lähenemine [Mat94] on eelmisest isegi võimsamaks osutunud (saab tõestada, et kui süsteem on kindel lineaarse analüüsi vastu, siis ei saa teda murda ka diferentsiaalse analüüsiga). Meetod töötati algselt välja ründeks algoritmi DES vastu ja tema idee seisnes DESi mittelineaarsete S-kastide lähendamises lineaarvormidega. Hetkel pole lineaarse analüüsi rakenduste uurimisega räsifunktsioonide korral veel eriti kaugele jõutud, aga vastavaid tõsiseid tulemusi on

oodata juba lähiajal. Et ka selle rünnaku vastu universaalset rohtu ilmselt leida ei õnnestu, tuleb räsifunktsioone kasutavate protokollide välja töötamisel räsifunktsiooni murdumisega lihtsalt arvestada.

4.4 Kokkuvõtteks

Nagu ülalkirjutatust nähtub, on räsifunktsiooni murdumine teatud määral paratamatu. Samas tähendab see kogu ajatempli ahela ebaturvaliseks muutumist, järelikult peavad TSSi protokollid nägema ette vastavaidmeetmeid sellise olukorra ennetamiseks.

Üheks võimalikuks ja praktikas kasutatavaks lahenduseks on rakendada paralleelselt kaht erinevat räsifunktsiooni. Kuni ükski neist turvalisust pole kaotanud, võib kogu ahelat veel usaldada, kui aga mõlemad murduvad, tundub üsna ebatõenäoline võimalus leida neile ühiseid kollisioone. Kui võtta linkinformatsiooniks näiteks

$$L_n = H_1(t_n, \mathrm{ID}_n, y_n, L_{n-1})H_2(t_n, \mathrm{ID}_n, y_n, L_{n-1}),$$

tuleks ahela n-nda lüli andmete võltsimiseks leida kollisioonid üheaegselt mõlemale räsifunktsioonile. Samas võib funktsiooonide H_1 ja H_2 konkatenatsiooni vaadelda ühe räsifunktsioonina $H = H_1H_2$, mis on oma komponentidest aga oluliselt kohmakam ja aeglasem. Lisaks on sellise konkateneeritud funktsiooni kasutusele võtmisel väga raske veenduda, et komponendid on teineteisest sõltumatud. Sõltuvuse leidmine aga võib oluliselt kergendada funktsiooni H krüptoanalüüsi.

Seega – kas eelistada räsimeetodit, mille arvutamine võtab küll rohkem aega, kuid mille tulemus on turvalisem, või kiiremat meetodit, mis tõenäoliselt lähitulevikus murdub? Vastus peitub ajatempli ideoloogia lätetes. On ju ajatempleid vaja eelkõige dokumentidele, mille kehtivus huvitab meid teatava perioodi jooksul (lepingud, võlakirjad,notariaalpaberid). Vastavalt dokumendi iseloomule tulekski valida dokumendi signeerimise meetod – trammipilet võib maksvuse kaotada poole tunni pärast, testament aga peab tõestatav olema ka paarikümne aasta möödudes. Kuna loodav ajatemplite kontseptsioon näeb ette hierarhilise TSS-ide struktuuri (Peatükk 6), on ilmselt otstarbekas varustada kõrgema taseme server kindlama räsifunktsiooniga, andes talle töötlemiseks pikemaealiseid dokumente. Samas võivad madalama taseme serverid tembeldada lühema kehtivusajaga materjale, tehes seda kiiremini ja odavamalt.

Kuna teoreeme sõnastusega "Räsifunktsioon H püsib turvaline vähemalt aastani 2010" ei õnnestu ilmselt tõestada, tuleb kasutatavat funktsiooni perioodiliselt profülaktiliselt vahetada. Et varem signeeritud dokumendid räsi murdumisel ei tühistuks, on vaja nad (koos vana räsiga) uuesti signeerida. Vastavat problemaatikat käsitlebki järgmine peatükk.

Peatükk 5

Ajatemplite aegumisest ja pikendamisest

5.1 Ajatempli aegumine

Olgu $\tau_0, \tau_1, \ldots$ ajahetked. Kasutagu TSS_1 ajatemplite linkimiseks ajavahemikus τ_{i-1} kuni τ_i räsifunktsiooni H_i. Olgu τ'_i ajahetk, millal räsifunktsioon H_i arvatakse veel mitte lahti murtud olevat (seejuures peab kehtima $\tau'_i \geq \tau_i$).

Ajatempli serveri TSS_1 ajahetkel t_n, kus $\tau_{i-1} < t_n < \tau_{i+1}$, väljaantaval templil on aegumistähtaeg, mis ei ole suurem kui τ'_i. Seega, kui Alice'ile antakse hetkel t_n välja n-s ajatempel, siis on see kujul

$$S_n = \{(n, t_n, y_n, \mathrm{ID}_n;\ L_n;\ t'_n)\}_{K^{-1}_{TSS_1}},$$

kus

- ID_n on Alice'i identifikaator
- $y_n = \{h(x_n)\}_{K_A^{-1}}$, kus x_n on tembeldatav dokument ja h on funktsioon, mis seab dokumendile vastavusse tema sõnumilühendi
- L_n on n-nda templi linkimisinformatsioon, mille koosseisu kuulub mingite eelnevate templite linkimisinformatsioonide konkatenatsiooni sõnumilühend räsifunktsiooniga H_i
- t'_n on n-nda ajatempli aegumistähtaeg (seejuures $t'_n \leq \tau'_i$).

See ajatempel loetakse kehtetuks, kui tema aegumistähtaeg on möödunud või kui räsifunktsioon H_i on lahti murtud (kui τ'_i valik on õnnestunud, siis on templi aegumistähtaeg enne räsifunktsiooni murdumishetke). Kehtetu ajatempliga S_n ei saa tõestada y_n-i olemasolu enne hetke t_n.

5.2 Ajatempli kehtivusaja pikendamine

Kui n-s ajatempel on kehtiv ajahetkel $t_m > t_n$ ja ajahetkel t_m väljaantava ajatempli kehtivusaeg võib olla suurem kui t'_n (s.t. olgu $\tau_{j-1} < t_m < \tau_j$ ja $\tau'_j > t'_n$), siis võib igaüks, esitada TSS_2-le päringu S_n kehtivusaja pikendamiseks (siin võib, kuid ei pea olema $TSS_1 = TSS_2$), seejuures

1. TSS_2 kontrollib TSS_1 käest S_n kehtivust ja leiab selle järgi t_n ja y_n
2. TSS_2 väljastab pärijale uue ajatempli (olgu selle järjekorranumber näiteks m)

$$S_m = \{(m, t_m, ID_m, (t_n, y_n);\ L_m;\ t'_m)\}_{K^{-1}_{TSS_2}},$$

   kus ID_m on pärija identifikaator.

Soovides sedasama templit veelkord pikendada, näiteks ajahetkel $t_{m'}$ (sel hetkel peab S_m kehtima), antakse pikendamispäringu esitajale (identifikaatoriga $ID_{m'}$) välja tempel (olgu selle järjekorranumber näiteks m')

$$S_{m'} = \{(m', t_{m'}, ID_{m'}, (t_n, y_n);\ L_{m'};\ t'_{m'})\}_{K^{-1}_{TSS_k}},$$

kus TSS_k on ajatempli server, millele päring esitati. Selle ajatempliga saab tõestada, et y_n eksisteeris ajahetkel t_n.

5.3 Pikendamise korrekstsuse põhjendus

Olgu ajatempliserveri TSS_k välja antud ajatempel S_m, mis väidab y_n olemasolu ajahetkel t_n, kehtiv. Sisuliselt väidab S_m seda, et ajahetkel t_m kontrollis TSS_k, kas y_n eksisteeris ajahetkel t_n, ja sai positiivse vastuse. Tõepoolest, olgu S_{m_0} esialgne ajatempel, mis väidab y_n olemasolu hetkel t_n, ja olgu $S_{m_1}, S_{m_2}, \ldots, S_{m_l}$ selle pikendused. Siis S_{m_1} väljastamise ajal kontrolliti S_{m_0} kehtivust ning S_{m_1} väidab, et tema väljastamise ajal oli S_{m_0} kehtiv, s.t. tema

väljastamise ajal võis väita, et y_n eksisteeris hetkel t_n. Kuna viimase väite tõesus ei sõltu tema ütlemise ajast, siis väidab S_{m_1}, et y_n eksisteeris hetkel t_n. Analoogiliselt saame, et $S_{m_2}, \ldots, S_{m_l}$ väidavad, et y_n eksisteeris hetkel t_n.
Kuna S_m väidab, et y_n eksisteeris hetkel t_n ja S_m on kehtiv, siis pidi y_n eksisteerima hetkel t_n.

Peatükk 6

Ajatemplite süsteemi infrastruktuur

Ajatempli teenust võib osutada ka üksainus asutus, kuid see pole soovitav järgmistel põhjustel:

- Liiga suur töökoormus. Üksainus asutus ei pruugi toime tulla kõikide dokumentidega, mis nõuavad ajatemplit.
- Töökindluse nõuetest tingitud kõrge hind. Mida vähem on sertifitseerimiskeskusi, seda suuremad on talituse katkematuse tagamiseks tehtavad kulutused.

Edaspidi eeldame, et ajatempli teenust osutavad asutused/keskused moodustavad kaheastmelise hierarhia, mis langeb kokku sertifitseerimiskeskuste hierarhiaga. Iga sertifitseerimiskeskus võib osutada ajatempli teenust kõigile neile, kellele ta on väljastanud sertifikaadi.

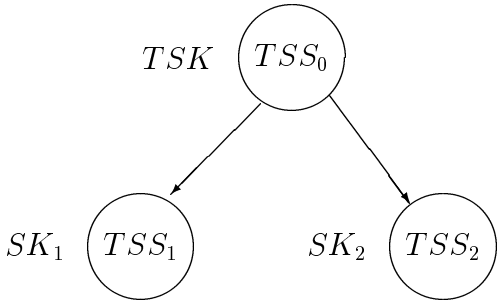
Ajatempli teenuse osutamine tähendab järgmiste tegevuste sooritamist:

- mingitele andmetele X käesoleva ajahetke t juurdelisamist TSS-i poolt ja **ajatempli**, st signeeritud infokogumi

  $$s_n(t, X) = \{TSS, n, t, X;\ L_n\}_{K_{\mathrm{TSS}}^{-1}}$$

  välja andmist, kus n on välja antava ajatempli number.
- ajatempli linkimist teatud hulga eelnevate ajatemplitega räsifunktsiooni abil. Selleks moodustatakse linkimisinfo:

  $$L_{n+1} = (t_n, X, h(L_n)),$$



Joonis 6.1: Ajatempli teenuse pakkujate hierarhia.

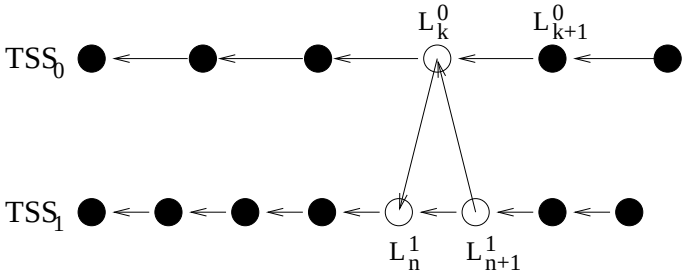
mida kasutatakse analoogilisel viisil järgneva ajatempli arvutamisel.

6.1 Ahelate sidumine

Kasutajaid teenindavad TSS-d on kohustatud teatava aja jooksul, mis on paika pandud sertifitseerimispoliitikas, võtma ajatempleid TSS_0-lt oma viimati välja antud ajatemplist, et kindlustada tõestatav seos teiste ajatempli teenustega.

Joonisel 6.2 on kujutatud olukord, kus ajatempli server TSS_1 küsib ajatemplit serverilt TSS_0 ajatemplile L_n^1. Siin

$$L_k^0 = H(k, t_k^0, L_n^1, \mathrm{TSS}_1, L_{k-1}^0) \text{ ja}$$
$$L_{n+1}^1 = H(n+1, t_{n+1}^1, L_k^0, \mathrm{TSS}_0, L_n^1).$$



Joonis 6.2: Ajatempli ahelate sidumine.

6.2 Erinevate turvapoliitikatega serverid

Ajatempli serveritel võivad olla erinevad turvapoliitikad, milles on määratletud järgmised näitajad:

- dokumentide eluea piirangud,
- ahelate sidumise sagedus (mida sagedamini ahelaid sünkroniseeritakse tipmise ajatempli serveriga,?seda turvalisem antud keskuse teenus on),
- dokumentide rahalise väärtuse ülempiir,
- kasutatav kliendi autentimise protokoll,
- kasutatavad räsifunktsioonid ja räsifunktsioonide uuendamise sagedus,
- kasutatav signatuuriskeem (algoritm, võtme pikkus),
- kõik tingimused, mis peavad olema sertifitseerimiskeskuse turvapoliitikas.

Näiteks võib üks ajatempli server väljastada ainult lühikese elueaga dokumentide ajatempleid. Selline server võib kasutada kiireid, kuid vähem turvalisi krüptograafilisi primitiive, sealjuures ei pruugi antud server toetada ajatemplite pikendamist. Nõrgema turvapoliitika tasakaalustavad selle serveri madalamad teenuse hinnad.

6.3 Ajatempli teenuse tasulisus

Ajatempli teenuse tasustamine on loomulik abinõu *denial of service* tüüpi rünnete vastu. Kõige efektiivsem tasumisviis on siin teenuse eest ette maksmine. Ajatempli teenuse osutaja peab andmebaasi

nr	nimi	summa
1	Alice	200
2	Carol	300
...	...	...

kus iga kliendi jaoks on kirjas tema ettemakstud summa. Iga ajatempli eest vähendatakse andmebaasis olevat summat. Kui see summa muutub nulliks, siis ei teenindata vastavat klienti nii kaua, kuni ta on uuesti maksnud. Teenustasu suurus sõltub sertifitseerimiskeskusest. On kahte liiki tasulisi teenuseid:

- Uutele andmetele ajatempli saamine, millest ülalpool juttu oli, ja
- Vanade ajatemplite uuendamine, st vanadele ajatemplitele

$$X = (m, H(m, t_m, y_m, \mathrm{ID}_m, L_m))$$

ajatempli saamine. Seejuures antakse välja ajatempel

$$s_n = \{n, t_n, X, \mathrm{TSS}, L_n\}_{K_{\mathrm{TSS}}^{-1}}.$$

Siin on oluline, et igaüks saaks ilmutada initsiatiivi mistahes teise (ka teise TSS-i poolt välja antud) ajatempli uuendamiseks. Seda on vaja näiteks selliste lepingute ajatemplite uuendamiseks, millest kõik osapooled enam huvitatud ei ole. Näiteks laenulepingud vms. Siin tekib aga tõsine raskus, sest iga sertifitseerimiskeskus peab andmebaasi üksnes omaenda klientidest, mistõttu on tasu võtmine oluliselt raskendatud. Selle raskuse leevendamiseks/kõrvaldamiseks on kaks teed:

- Lepingu tüüpi[1] dokumentidele võtavad mõlemad osapooled ajatempli omaenda keskusest ajatempli ka vastaspoole signatuurile. Seega kaob vajadus teise keskuse ajatempleid uuendada. Piisab, kui lepingu uuendamisest huvitatud osapool pikendab ajatemplit omaenda keskuses.
- Tasumine võib toimuda tipmise keskuse kaudu.

Ajatempli tasuliseks muutmine eeldab **kliendi autentimist** enne ajatempli andmist. Autentimine võib toimuda mingi standardse autentimisprotokolli järgi.

[1]st sellistele dokumentidele, mille uuendamisest/säilimisest võivad olla huvitatud vaid kaks osapoolt.

Peatükk 7

Koondajatemplid

Sageli pole otstarbekas tembeldada dokumente eraldi, vaid moodustada neile nn. koondajatempel. See teguviis on kasulik just sellistel juhtudel, kus lühikese aja jooksul moodustatakse palju erinevaid dokumente, mille omavaheline ajaline järgnevus pole oluline.

Koondajatempli moodustamiseks kasutatakse räsifunktsioone, mingisugust salajast infot selleks vaja ei ole. Joonisel 7.1 on esitatud puukujuline skeem, mis selgitab, kuidas moodustada koondajatemplit kaheksale erinevale dokumendile y_A, ..., y_H.

Esmalt arvutatakse neli vahepealset väärtust

$$H_1 = H(y_A, y_B), \quad H_2 = H(y_C, y_D), \quad H_3 = H(y_E, y_F), \quad H_4 = H(y_G, y_H),$$

millest omakorda moodustatakse kaks väärtust

$$H_5 = H(H_1, H_2), \quad H_6 = H(H_3, H_4).$$

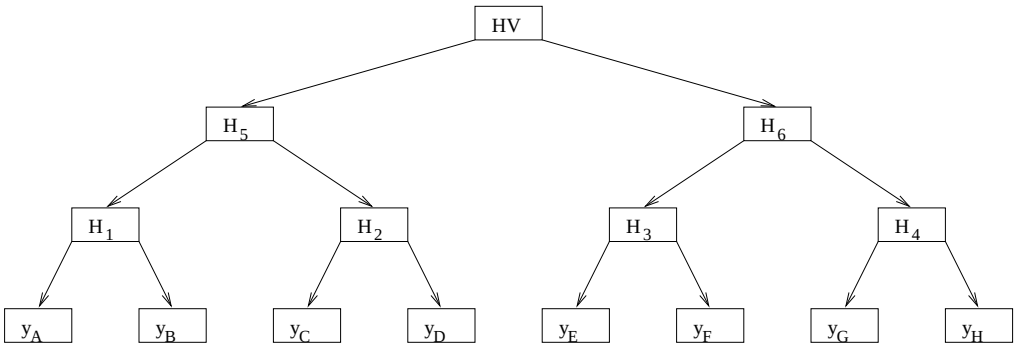
Nendest kahest väärtusest saadakse väärtus

$$HV = H(H_5, H_6),$$

mis lastakse tembeldada TSS-s. Kirjeldatud systeem võimaldab omistada dokumentidele ka individuaalseid ajatempleid. Niteks dokumendi y_B ajatempel oleks jada

$$(y_A, \text{right}), (H_2, \text{left}), (H_6, \text{left}).$$

Seega on võimalik tekitada usaldatav ajatemplite süsteem ainult ühe päringuga ajatempli serverist. Seda muidugi tingimusel, et dokumentide omavahelised ajalised suhted pole olulised.



Joonis 7.1: Koondajatempli moodustamine

Toodud skeem on sobilik rakendamiseks pankadevaheliste maksete tembeldamisel, kus ajaühikus võib laekuda palju maksekorraldusi, millele eraldi ajatempli võtmine läheks liiga kulukaks.

Peatükk 8

Kokkuvõte

Käesoleva projekti esimese osa raames töötati välja ajatemplite süsteemi üldpõhimõtted, mis on vajalikud tehnilise ülesande püstitusel süsteemi realiseerimiseks vajaliku tarkvara loomiseks. Töö käigus on leitud lahendused mitmetele seni maailma mastaabis lahendamata probleemidele (ahelate kiired linkimisviisid, ajatemplite ületembeldamine, ajatempli serverite hierarhiline struktuur, ajatempli ahelate sidumine ning *denial of service* ründe vältimine teenuse tasustamisega). Pakutud lahendused ei lõpliku iseloomuga, lahenduste täiustamiseks on vajalik koostöö teiste riikide spetsialistidega.

Kirjandus

[ACPZ97] C. Adams, P. Cain, D. Pinkas, and R. Zuccherato. Time Stamp Protocols. Technical report, Internet Draft, November 1997. draft-adams-time-stamp-00.txt.

[AZ97] C. Adams and R. Zuccherato. Notary Protocols. Technical report, Internet Draft, November 1997. draft-adams-notary-00.txt.

[CvHP92] D. Chaum, E. van Heijst, and B. Pfitzmann. Cryptographically strong undeniable signatures, unconditionally secure for the signer. In *Advances in Cryptology — CRYPTO '91*, pages 470–474, 1992. Lecture Notes in Computer Science No. 576.

[DH76] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Trans. Inform. Theory*, IT-22:644–654, November 1976.

[GGH96] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Collision–Free Hashing from Lattice Problems. Technical report, Electronic Colloquium on Computational Complexity, 1996. TR96-042.

[HS92a] S. Haber and W.-S. Stornetta. Digital document time-stamping with catenate certificate. Technical report, 1992. US-patent no 5,136,646.

[HS92b] S. Haber and W.-S. Stornetta. Method for secure timestamping of digital documents. Technical report, 1992. US-patent no 5,136,647.

[HS94] S. Haber and W.-S. Stornetta. Method of extending the validity of a cryptographic certificate. Technical report, 1994. US-patent no 5,373,561.

[HS95] S. Haber and W.-S. Stornetta. Method of extending the validity of a cryptographic certificate. Technical report, 1995. US-patent no RE. 34,954.

[Mat94] M. Matsui. Linear cryptanalysis method for DES cipher. In T. Helleseth, editor, *Advances in Cryptology — Eurocrypt '93*, volume 765 of *Lecture Notes in Computer Science*, pages 386–397, Berlin, 1994. Springer-Verlag.

[MQ97] H. Massias and J.-J. Quisquater. Time and Cryptography. Technical report, Université catholique de Louvain, March 1997. TIMESEC Technical Report WP1.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[Sti95] Douglas Stinson. *Cryptography Theory and Practice*. CRC Press, 1995.