UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Simon Kozina

Digitalno časovno žigosanje

SEMINARSKA NALOGA
pri predmetu
KRIPTOGRAFIJA IN TEORIJA KODIRANJA 2

Mentor: prof. dr. Aleksandar Jurišić

Ljubljana, 2010

Kazalo

Povzetek

V današnjem svetu je vse več pomembnih dokumentov v elektronski obliki. Pri nekaterih je pomembno kdaj so nastali in kdo je njihov avtor. Vsi pa vemo, da so to podatki, ki jih je enostavno spremeniti pri digitalnih zapisih. Potrebujemo torej metodo, s katero bomo lahko zasidrali dokument v času in brez dvoma določili njegovega lastnika.

Govorimo torej o digitalnem časovnem žigosanju. Predstavljenih je več metod centraliziranega in distribuiranega modela žigosanja. Kako ločimo dobre metode od slabih in na kaj moramo paziti pri časovnem žigosanju. Omenjenih je tudi nekaj napadov na nekatere izmed metod, ki se najbolj uporabljajo v današnjih časih. Za konec so predstavljeni še resnični sistemi, ki zagotavljajo storitve digitalnega časovnega žigosanja.

Ključne besede:

digitalno časovno žigosanje, elektronski notariat, time-stamping authority

Poglavje 1

Uvod

V dobi informacijske družbe in elektronskega poslovanja se vedno več informacij pojavlja v elektronski obliki in vedno več papirnatih dokumentov prevzema novo podobo in obliko. Danes so v večini primerov dokumenti v elektronski obliki že ob samem nastajanju, kajti pisalni stroji in magnetni zvočni in video zapisi se sploh več ne uporabljajo. Ob razcvetu interneta in njegove najpopularnejše storitve elektronske pošte se vsi ti dokumenti vedno pogosteje tudi prenašajo po elektronski poti. Zaradi vse bolj uveljavljenega mednarodnega poslovanja je prenos dokumentov na dolge razdalje postal vsakdanje opravilo.

Mnogi od elektronskih dokumentov (na primer poslovne pogodbe) so lahko vir tožb in pravd na sodiščih, podobno kot v tradicionalnih okoljih. Seveda elektronski dokumenti niso verodostojni sami po sebi, saj jih je enostavno kopirati in spreminjati brez možnosti razkritja takih posegov. Sami znaki so med seboj enaki in iz njih ni možno prepoznati pisca, kot pri lastnoročnem pisanju na papir. Tudi podpis dokumenta (odtipkana ime in priimek na koncu dokumenta) ni prepoznaven, predvsem pa ni fizično povezan z vsebino dokumenta (medtem ko lastnoročni podpis obstaja na istem papirju kot tekst in je s tem z njim neločljivo povezan). Tudi čas nastanka elektronskega dokumenta lahko poljubno spreminjamo, medtem ko imamo pri navadnih dokumentih, ki so podpisani s strani notarja, čas fiksno določen. Potrebujemo torej metodo s katero bomo lahko elektronske dokumente brez dvoma uvrstili v pravilen čas in jim določili pravilnega lastnika. Trenutno najoptimalnejša rešitev se kaže v uporabi časovnih žigov. Začetnika na tem področju sta Haber in Stornetta, ki sta se s tem začela ukvarjati že leta 1991. Eden glavnih razlogov za njune raziskave so bili patenti, katere lahko enostavno zaščitimo s časovnim žigosanjem in hkrati ne izdamo njihove vsebine. Po njunem mnenju mora imeti vsaka metoda digitalnega časovnega žigosanja naslednji dve lastnosti [1]:

- Podatke moramo žigosati tako, da se ne zanašamo na lastnosti medija na katerem so podatki shranjeni. Nemogoče mora biti spreminjanje tudi enega samega bita informacije.

- Žigosanje s časom, ki ni enak dejanskemu, bi moralo biti nemogoče.

Ker je v današnjem času vse več pomembnih dokumentov v digitalni obliki, je postalo digitalno časovno žigosanje pomemben člen pri ugotavljanju časa nastanka oz. časa zadnje spremembe nekega dokumenta.

V drugem poglavju sta opisana dva kriptografska primitiva, zgoščevalne funkcije in digitalne sheme, ki sta osnovi časovnega žigosanja. V tretjem poglavju so predstavljene dosedanje rešitve na tem področju. Vse te rešitve so odvisne od agencije, ki nam omogoča žigosanje dokumentov. V četrtem poglavju je predstavljenih več metod, s katerimi se znebimo odvisnosti od agencije za žigosanje. Tukaj smo odvisno samo od ostalih uporabnikov s katerimi skupaj žigosamo dokumente. V petem poglavju sta predstavljeni dve glavni vrsti napadov na protokol časovnega žigosanja. Šesto poglavje opisuje trenutne sisteme digitalnega časovnega žigosanja, njihovo uporabo in varnost. V zaključku je podan povzetek seminarske naloge in diskusija o smiselnosti uporabe centraliziranega časovnega žigosanja.

Poglavje 2

Osnove

V tem poglavju je opisanih nekaj osnovnih kriptografskih primitivov, ki jih potrebujemo za definiranje digitalnega časovnega žigosanja.

2.1 Zgoščevalne funkcije brez trčenj

Zgoščevalna funkcija brez trčenj [2] je funkcija H, ki ima naslednje lastnosti:

- *kompresija* - H priredi vhodu x, ki je poljubne dolžine, kratko končno zaporedje bitov $H(x)$ dolžine ℓ. Opravimo torej transformacijo $\{0,1\}^* \rightarrow \{0,1\}^\ell$.
- *enostavno računanje* - če imamo dano funkcijo H in vhod x, lahko enostavno izračunamo $H(x)$.
- *krepko brez trčenj* - v doglednem času ni možno najti (izračunati) dveh različnih vhodov x in x', ki ju funkcija H preslika v enak izhod $H(x) = H(x')$.

Za razbitje varnosti zgoščevalne funkcije, ki je krepko brez trčenj, lahko napadalec uporabi napad z generiranjem naključnih sporočil in poskusom izdelave povzetka določene vrednosti (ki je verjetno enaka vrednosti povzetka dokumenta, ki ga želi napadalec "ponarediti"). Za T poskusov in dolžino povzetka n bitov je verjetnost uspeha $\frac{T}{2^n}$. Zgoščevalne funkcije so izpostavljene napadu "birthday attack"(algoritem iskanja kolizij, na podlagi katerega obstaja verjetnost vsaj ene kolizije večja od 0,5 med $1,17 \cdot \sqrt{A}$ vrednosti od vseh A vrednosti; če je povzetek dolg 256 bitov je možno dobiti kolizijo z verjetnostjo 0,5 med

nekaj manj kot $4 \cdot 10^{38}$ povzetki. V praksi so ti napadi neuspešni, če je dolžina povzetka dovolj velika. V današnjih časih je temu tako, saj se giblje med 160, 256 in 512 biti.

Dandanes je obstoj zgoščevalnih funkcij brez trčenj še vedno pomemben problem. Trenutno je najbolj uporabljana funkcija SHA-2, vendar pa že načrtujejo SHA-3, ki naj bi svojo predhodnico zamenjala že leta 2012.

2.2 Digitalni podpisi

Še eden od kriptografskih primitivov, ki jih potrebujemo pri digitalnem časovnem žigosanju, so digitalni podpisi. **Digitalni podpis** je definiran kot $\text{sig}_A(M)$, kjer je M sporočilo, ki ga je podpisala oseba A. Protokoli digitalnega časovnega žigosanja potrebujejo sheme digitalnih podpisov, kjer je možno podpisovati poljubne podatke. Zato morajo biti te sheme varne proti napadom z izbranim besedilom. Najbolj varne in trenutno najbolj uporabljane sheme digitalnega podpisovanja so trenutno RSA, ElGamal in DSS.

Poglavje 3

Timestamping authority (TSA)

Obstajata dva popolnoma različna načina digitalnega časovnega žigosanja. Prvi način temelji na načelu centralizacije. Imamo agencijo za digitalno časovno žigosanje (TSA), katera nam dokument žigosa in pri kateri lahko kasneje nekdo drug preveri kdaj je bil ta dokument ustvarjen oz. nazadnje spremenjen. Drugi način vključuje distribucijo odgovornosti žigosanja med množico uporabnikov vključenih v sistem digitalnega časovnega žigosanja. Ta način bo predstavljen v naslednjem poglavju.

3.1 Naivna rešitev

Naivna rešitev bo služila kot ogrodje za vse naslednje izboljšave. Imamo agencijo, kateri uporabnik pošlje dokument, ki ga želi žigosati. Ta shrani dokument in si zapomni še datum in čas, ko je dokument prejela. Uporabnik dobi samo potrditev, da je bil njegov dokument žigosan. Če je potrebno preveriti integriteto dokumenta, se ga pošlje agenciji. Ta ga lahko primerja s kopijo, ki jo ima shranjeno. V primeru enakosti velja, da dokument ni bil spremenjen po datumu, ki ga hrani agencija. Vidimo, da so izpolnjeni osnovni pogoji za digitalno časovno žigosanje. Vendar se tukaj pojavi več problemov [1]:

Zasebnost 3.1.1 Ta metoda ogroža zasebnost dokumenta na dva načina. Napadalec bi lahko prisluškoval med prenašanjem dokumenta; dokument je vedno na voljo TSA. Torej mora klient skrbeti, ne samo za varnost dokumentov, ki jih hrani sam, ampak tudi dokumentov, ki jih hrani TSA.

Pasovna širina in prostor 3.1.2 Tako čas prenosa kot tudi količina prostora, katero potrebuje TSA za shranjevanje dokumenta, sta odvisna od

velikosti dokumenta. Torej je žigosanje velikih datotek[1] lahko časovno potratno in zelo drago opravilo.

Možnost napak 3.1.3 Pride lahko do napake med prenosom datoteke, lahko se zgodi, da je datoteka nepravilno žigosana ob prihodu k TSA, ali pa se datoteka poškoduje med hrambo pri TSA. Vsak od teh dogodkov bi povzročil neveljavno preverjanje časovnega žiga.

Zaupanje 3.1.4 Glavni problem je seveda zaupanje. TSA lahko spremeni časovni žig našega dokumenta (po žigosanju) in tako postane naš žig nepravilen.

V nadaljevanju bodo opisane rešitve prvih treh prej navedenih težav. Problem z zaupanjem se izkaže kot težji in bomo zaenkrat predpostavljali, da zaupamo naši agenciji. Seveda pa bo kasneje predstavljena tudi rešitev, ki izboljša naše zaupanje v TSA.

3.2 Zaupanja vredna TSA

Recimo, da ima TSA določeno varnostno politiko v skladu s standardom ISO/IEC 27001. Splošna zahteva standarda je, da podjetje vzpostavi, vpelje, izvaja, spremlja, pregleduje, vzdržuje in izboljšuje dokumentiran sistem upravljanja varovanja informacij v okviru vseh poslovnih procesov podjetja in tveganj, ki ga ogrožajo. Sedaj, ko za varnost ne rabimo skrbeti, se lahko osredotočimo na ostale tri probleme iz prejšnjega poglavja. Naivno rešitev bomo popravili z dvema izboljšavama.

Prva je uporaba zgoščevalnih funkcij. Namesto da bi poslali TSA celotno datoteko x, ji raje pošljemo samo njen povzetek $H(x) = y$. Uporaba povzetka $y = H(x)$ reši problema 3.1.1 in 3.1.2. Problem 3.1.2 je trivialen zaradi same definicije zgoščevalne funkcije H, saj ima ta lastnost kompresije. Torej so vsi povzetki dolgi točno ℓ bitov in jih tako lahko učinkovito shranjujemo in prenašamo preko interneta. Tudi problem 3.1.1 je trivialen, saj uporabljamo zgoščevalno funkcijo brez trčenj.

Trditev 3.2.1 *Časovno žigosanje povzetka $H(x) = y$ je enakovredno žigosanju dokumenta x zaradi uporabe zgoščevalne funkcije brez trčenj.*

[1] Vedeti moramo, da lahko digitalno časovno žigosamo vse vrste digitalnih zapisov, kot so recimo dokumenti, slike ali celo video vsebine. Tako so lahko datoteke velike tudi nekaj Gb.

Dokaz Recimo, da zgornja trditev ne velja. To bi pomenilo, da lahko najdemo takšen dokument $x' \neq x$, da bi veljalo $y = H(x) = H(x')$. Če bi sedaj žigosali x in x' bi dobili dva različna časovna žiga, če pa bi žigosali povzetek y bi za oba dokumenta dobili isti časovni žig. Vendar pa ob predpostavki, da uporabljamo zgoščevalno funkcijo brez trčenj, ne moremo najti takšnega dokumenta x', da bi veljalo $H(x) = H(x')$.

■

Druga izboljšava je uporaba digitalnih podpisov. TSA dobi povzetek y od klienta, kateremu pripne datum in čas, vse skupaj podpiše in pošlje nazaj klientu. Ta lahko nato preveri podpis in s tem preveri, da je TSA pravilno časovno žigosal dokument in da je dobil pravilni y. S tem odpravimo tudi problem 3.1.3 na strani TSA in ji omogočimo, da sama več ne hrani nobenih podatkov o žigosanju.

3.3 Povezovanje

Do sedaj opisani metodi nam omogočajo digitalno časovno žigosanje, vendar še vedno ne moremo zaupati TSA, saj nimamo mehanizma, ki bi ji preprečeval izdajo lažnega žiga, tj. napačnega datuma. Želimo si torej mehanizem, ki bi prisilil TSA k zagotavljanju izdajanja pravilnih časovnih žigov, četudi bi nam želela izdati napačnega.

Rešitev lahko najdemo v zaporedju zahtevkov klientov. Tega ne moremo poznati vnaprej, saj ne vemo kateri klient bo naslednji zahteval storitve TSA in katero sporočilo bo želel žigosati. Če bi torej v žig dodali še informacije o žigosanju prejšnjih zahtevkov, bi vedeli, da je bil naš dokument žigosan za vsemi ostalimi. Ta rešitev nam omogoča omejevanje časa tudi v drugo smer, saj TSA ne more izdati naslednjih žigov, če še ni obdelala našega zahtevka.

3.3.1 Linearno povezovanje (2-povezovanje)

Opravka imamo s povezovanjem dveh zaporednih zahtevkov. Informacije predhodnega zahtevka uporabimo pri izdaji žiga trenutnega zahtevka. Zahtevek klienta za žigosanje je torej sestavljen iz povzetka y dolžine ℓ-bitov in klientove identifikacijske številke ID. TSA izdaja zaporedno oštevilčene časovne žige. Ko dobi n-ti zahtevek v zaporedju (y_n, ID_n) od klienta naredi dve stvari:

1. TSA pošlje klientu podpisan žig $s = \mathrm{sig}_{\mathrm{TSA}}(C_n)$. Žig je v obliki

$$C_n = (n, t_n, \mathrm{ID}_n, y_n; L_n) \tag{3.1}$$

kjer je n zaporedna številka, t_n čas in datum žigosanja, ID_n identifikacijska številka klienta in y povzetek datoteke. L_n sestavljajo povezovalne informacije, ki so določene s predhodnim zahtevkom:

$$L_n = (t_{n-1}, \mathrm{ID}_{n-1}, y_{n-1}, H(L_{n-1})) \tag{3.2}$$

2. Ko TSA obdela naslednji zahtevek pošlje klientu identifikacijsko številko naslednjega zahtevka, torej ID_{n+1}.

Ko klient prejme s in ID_{n+1}, lahko preveri, če sta podpis in žig pravilna. Preveri torej, ali je C_n v pravilni obliki in ali vsebuje pravilni čas t.

Če želi nekdo (recimo stranka) kasneje preveriti časovni žig dokumenta x, najprej preveri, da sta (s, ID_{n+1}) v pravilni obliki. Nato preveri, če je vrednost $y' = H(x)$ enaka vrednosti y_n, ki se nahaja v podpisanem žigu s. Vendar preverjevalec še vedno nima zagotovila, da TSA in klient ne sodelujeta pri ponarejanju časovnega žigosanja. Ker pa vemo, kateri klient je naslednji zahteval časovno žigosanje (ID_{n+1}), ga lahko povprašamo naj nam dostavi svoj časovni žig (s', ID_{n+2}). Ta vsebuje podpisan žig

$$s' = \mathrm{sig}_{\mathrm{TSA}}(n+1, t_{n+1}, \mathrm{ID}_{n+1}, y_{n+1}; L_{n+1})$$

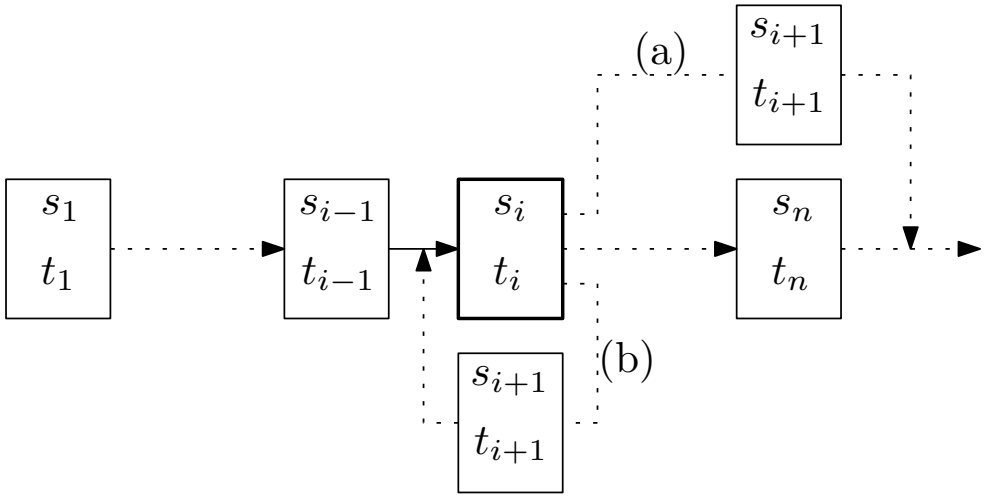
in znotraj njega tudi povezovalno informacijo L_{n+1}, ki vsebuje povzetek y_n. S pomočjo L_n iz originalnega žiga, lahko izračunamo $H(L_n)$ in preverimo, če se ujema s tistim v L_{n+1}. S tem zaključimo preverjanje. Če še vedno obstaja dvom (recimo, da oba uporabnika sodelujeta), lahko sedaj preverimo še klienta z identifikacijsko številko ID_{n+2} in tako naprej dokler želimo. Na podoben način lahko preverimo tudi časovne žige izdane pred našim, kjer začnemo s preverjanjem klienta ID_{n-1}.

Zakaj to onemogoča, da bi TSA izdajala napačne žige? Na začetku omeni dejstvo, da lahko časovni žig ponaredimo samo s pomočjo TSA. Sami ga ne moremo, saj nimamo dostopa da zasebnega ključa agencije, da bi lahko žig podpisali.

Trditev 3.3.1 *TSA ne more izdati časovnega žiga s poznejšim datumom od trenutnega.*

Trditev 3.3.2 *TSA ne more izdati žiga s predhodnim datumom.*

Dokaz Na sliki 3.1 imamo verigo časovnih žigov in njihove čase. Velja $t_1 < \ldots < t_{i-1} < t_i < \ldots < t_n$. Trenutno smo obdelali zahtevek z zaporedno



Slika 3.1: Primera, kako bi časovni žig z zaporedno številko $i+1$ dodali naprej v verigi žigov (a), ali nazaj (b).

številko i. Nato prejmemo dokument, katerega bi hoteli napačno časovno žigosati. Njegov časovni žig bi imel obliko

$$s_{i+1} = \mathrm{sig}_{\mathrm{TSA}}(i+1, t_{i+1}, \mathrm{ID}_{i+1}, y_{i+1}; L_{i+1})$$

Če bi hoteli, da ima časovni žig dokumenta $i+1$ poznejši datum, ga moram vstaviti za nek kasnejši časovni žig, recimo s_n. Tako bi veljalo $t_{i+1} > t_n$. Njegova povezovalna informacija L_{i+1} bi sedaj morala vsebovati podatke o žigu s_n (glej enačbo 3.2). Ugotovimo, da bi za napačno žigosanje dokumentov, pri katerih bi želeli datum prestaviti naprej, potrebovali podatke časovnih žigov, katerih še nismo izdali. Trditev 3.3.1 torej velja.

Poskušajmo sedaj žig s_{i+1} vstaviti v že obstoječo verigo žigov, recimo med žiga s_{i-1} in s_i (identično bi bilo vstavljanje s_{i+1} med katerakoli druga zaporedna žiga). Želimo, da velja $t_{i+1} < t_i$. Časovni žig s_i, vsebuje informacijo o žigu L_{i-1} (3.1). Tako je v resnici s_{i+1} že določen in mora biti enak s_{i-1}, saj bi drugače takoj zaznali nepravilno žigosanje. Tako poznamo tudi povzetek $y_{i+1} = y_{i-1}$. Najti moramo torej tak dokument x_{i+1}, da bo veljalo $H(x_{i+1}) = y_{i+1}$. Trivialna rešitev bi bila $x_{i-1} = x_{i+1}$, vendar s tem ne bi nič pridobili, saj bi imeli 2 identična žiga. Preostane nam še $x_{i+1} \neq_{i-1}$, katerega pa ne moremo najti, saj uporabljamo zgoščevalno funkcijo brez trčenj. Velja tudi trditev 3.3.2.

■

3.3.2 k-povezovanje

Pri zgornji shemi morajo vsi klienti shranjevati svoje časovne žige, da jih lahko kasneje posredujejo različnim strankam. To je nujno, saj imamo linearno verigo, torej sta vedno povezana po 2 zaporedna dokumenta.

To omejitev bomo tukaj malo ublažili. Sedaj ne povezujemo več samo prejšnjega dokumenta, ampak povezujemo k prejšnjih dokumentov. Kako bi sedaj TSA odgovorila na n-ti zahteveh žigosanja:

1. Kot pri linearnem povezovanju imamo sedaj časovni žig v obliki $C_n = (n, t_n, \mathrm{ID}_n, y_n; L_n)$. Spremeniti moramo vsebino naše povezovalne informacije L_n, ki sedaj zgleda tako:

$$L_n = [(t_{n-k}, \mathrm{ID}_{n-k}, y_{n-k}, H(L_{n-k})), \ldots, (t_{n-1}, \mathrm{ID}_{n-1}, y_{n-1}, H(L_{n-1}))].$$

2. Ko TSA obravnava naslednjih k zahtevkov, klientu pošlje seznam $(\mathrm{ID}_{n+1}, \ldots, \mathrm{ID}_{n+k})$.

Stranka sedaj pri preverjanju časovnega žiga na začetku preveri, ali je ta v pravilni obliki. Nato lahko vpraša kateregakoli izmed k naslednjih klientov, recimo ID_{n+i}, da mu posreduje svoj časovni žig. Kot pri linearnem povezovanju, vsebuje njegov časovni žig povezovalno informacijo L_{n+i}. Ta vsebuje tudi informacije o časovnem žigu C_n. Pravilnost C_n potrdimo, če se ujemata izračunana povzetek klientove povezovalne informacije $H(L_n)$ in povzetek, ki je naveden pri informaciji o C_n (v L_{n+i}). Časovni žig, ki ga dobimo od klienta ID_{n+i}, vsebuje tudi informacije o klientih $(\mathrm{ID}_{n+i+1}, \ldots, \mathrm{ID}_{n+i+k})$. Od tega je zadnjih i klientov novih, torej lahko naša stranka preverja tudi njihove časovne žige in to nadaljuje kolikor časa pač hoče.

Poleg tega, da smo s k-povezovanjem ublažili pogoj o hranjenju časovnih žigov posameznega klienta, smo še zmanjšali verjetnost izdaje ponarejenega časovnega žiga. Sedaj bi za pravilno uvrstitev dokumenta v časovno vrsto morali hkrati izračunati k trčenj funkcije H, namesto samo enega.

3.4 Akumulirani povezovalni model

V prejšnjih opisanih modelih, ki linearno ali k-povezujejo vse časovne žige, je potrebno ogromno sodelovanja med klienti. Dolgo lahko traja, preden pridemo do časovnega žiga v verigi, ki mu res lahko zaupamo. Kako torej odpravimo tako veliko potrebo po sodelovanju in zmanjšamo število preverjanj? Glavna ideja je, da žigosanje razdelimo v cikle [3][4]. Na koncu vsakega cikla se

izračuna super povzetek, ki je odvisen od vseh zahtevkov v tistem ciklu in super povzetka prejšnjega cikla. Mera za čas je sedaj dolžina cikla, ki jo moramo izračunati glede na uporabo sistema in potrebe aplikacije. Ločimo dva načina s katerimi določimo dolžino cikla. Ta je lahko fiksna glede na število oddanih zahtevkov (zaključi se, ko TSA dobi N zahtevkov), ali pa je določena s časovnim intervalom t (cikel se konča, ko preteče čas t). Splošni model žigosanja deluje na sledeči način:

1. Klient ID_i pošlje TSA povzetek dokumenta: $y_i = H(x_i)$.
2. TSA zbere vse zahteve, ki jih je prejela v trenutnem ciklu (y_i za $1 \leq i \leq m$) in izračuna super povzetek na podlagi vseh zahtevkov tega cikla in super povzetka prejšnjega cikla.
3. TSA nato pošlje vsem klientom super povzetek cikla in potrebne podatke, da si super povzetek tudi sami izračunajo. Te podatki in dokument skupaj predstavljajo digitalni časovni žig dokumenta.

TSA super povzetek cikla objavi skupaj z datumom in časom cikla recimo na internet, v časopis ali celo kakšen drug, zaupanja vreden vir. Časovni žig lahko preverimo tako, da izračunamo povzetek dokumenta in skupaj s podatki izračunamo super povzetek cikla. Nato samo še preverimo ali je ta pravilen (ga je TSA sploh kdaj izdala?) in dobimo podatek o tem kdaj ga je izdala.

3.5 Akumulirani drevesni povezovalni model

Drevesni model je eden od načinov kako lahko učinkovito povezujemo in združujemo povzetke istega cikla. Recimo, da v istem ciklu dobimo m zahtevkov, ki jih sestavimo v binarno drevo. Listi drevesa so povzetki dokumentov y_i. Super povzetek cikla i je RP_i in ga dobimo tako, da združimo korenski povzetek $P_{1,m}$ skupaj s super povzetkom prejšnjega cikla RP_{i-1}. Kako torej združimo dva povzetka in dobimo staršа? To lahko naredimo na naslednji način:

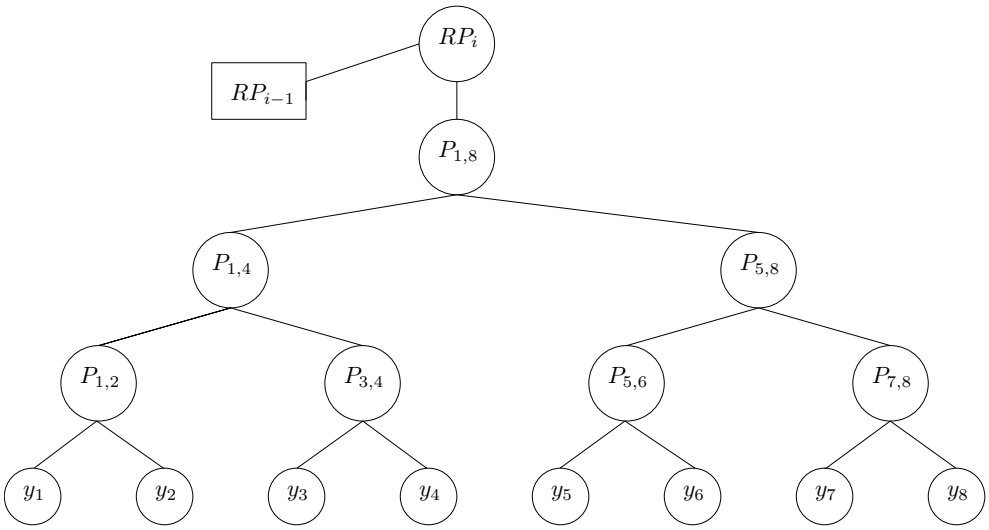
$$P_{\ell,r} = H(P_{\ell,\lfloor \frac{\ell+r}{2} \rfloor} | P_{\lceil \frac{\ell+r}{2} \rceil,r})$$

ℓ je oznaka najbolj levega lista v poddrevesu z vozliščem $P_{l,r}$, r pa oznaka najbolj desnega. Ta postopek ponavljamo dokler ne pridemo do korena drevesa. Če sta sinova vozlišča tudi lista drevesa uporabimo naslednjo formulo:

$$P_{\ell,r} = H(y_\ell | y_r).$$

Preostane nam še izračun super povzetka trenutnega cikla:

$$RP_i = H(RP_{i-1}|P_{1,m}).$$



Slika 3.2: Izgradnja binarnega drevesa v ciklu i z osmimi zahtevki.

Na sliki 3.2 vidimo prikaz binarnega drevesa cikla i, v kateri je bilo 8 zahtevkov za žigosanje. Če slučajno ni dovolj listov za gradnjo popolnega drevesa, vstavimo potrebno število listov, ki imajo naključno ali vnaprej določeno vrednost.

Časovni žig posameznega dokumenta vsebuje vse potrebne vrednosti za izgradnjo pravilne veje drevesa. V našem primeru bi časovni žig za četrti dokument izgledal tako:

$$C = \mathrm{sig}_{\mathrm{TSA}}(y_4, t_i, [(y_3, L), (P_{1,2}, L), (P_{5,8}, R), (RP_{i-1}, L)])$$

t_i predstavlja datum in čas cikla i, L (levo) in R (desno) pa povedo kako moramo združevati vrednosti, da pravilno pridemo do super povzetka cikla RP_i.

Stranka, ki sedaj želi preveriti časovni žig dokumenta, najprej preveri, ali je certifikat

$$C = \mathrm{sig}_{\mathrm{TSA}}(y, t, [(h_1, b_1), (h_2, b_2), \ldots, (h_\ell, b_\ell),]) \tag{3.3}$$

v pravilni obliki in izračuna $y_1 = H(x)$. $\ell = \lceil \log_2 m \rceil + 1$ in predstavlja višino binarnega drevesa, $h_1, \ldots, h_\ell$ izračunane povzetke na vsaki stopnji drevesa in $b_1, \ldots, b_\ell \in \{L, R\}$ način kako združujemo vrednosti med seboj.

Certifikat odklenemo z javnim ključem TSA in preverimo ali je $y = y_1$. Nato se lotimo izračuna super povzetka cikla.

Algoritem 3.5.1 *:*

for $i \leftarrow 1 \ldots \ell$
    if $b_i = L$*:*
        $y_{i+1} \leftarrow H(h_i|y_i)$
    else if $b_i = R$*:*
        $y_{i+1} \leftarrow H(y_i|h_i)$

Zadnja vrednost y_{l+1} je sedaj super povzetek našega cikla. TSA povprašamo kakšen je bil časovni žig ob času t in preverimo ali se ujema z izračunanim. Tako smo preverili, da je bil dokument x res časovno žigosan ob času t. Za ta izračun smo porabili $\mathcal{O}(\log_2 m)$ časa. Vidimo, da smo do zaupanja vrednega časovnega žiga prišli v $\mathcal{O}(\log_2 m)$ korakih, za razliko od linearnega povezovanja, kjer rabimo kar $\mathcal{O}(m)$ korakov.

Trditev 3.5.1 *Pri drevesnem povezovalnem modelu TSA ne more spremeniti datuma časovnega žiga.*

Dokaz Če bi hoteli datum zamenjati s poznejšim, se pojavi isti problem kot pri trditvi 3.3.1. Tudi tukaj še nimamo dokumentov potrebnih za sestavo drevesa v poznejšem ciklu.

Žigosanje s predhodnim datumom lahko realiziramo na dva načina:

1. Našemu preverjevalcu lahko pošljemo časovni žig nekega dokumenta x, ki je bil žigosan v ciklu, v katerega želimo uvrstiti svoj dokument. V žigu (enačba 3.3) se nahaja povzetek y dokumenta x. Potrebujemo torej tak $x' \neq x$, da velja $H(x) = H(x')$. Ker uporabljamo funkcijo brez trčenj, x' ne moremo izračunati.

2. Pri tem načinu ponaredimo celotni časovni žig. To lahko naredimo, saj preverjevalec na koncu preveri samo izračunani super povzetek tistega cikla. Imamo torej časovni žig

$$C = \mathrm{sig}_{\mathrm{TSA}}(y, t, [(h_1, b_1), (h_2, b_2), \ldots, (h_\ell, b_\ell),])$$

   kjer je $y = H(x)$ in t čas cikla k v katerega želimo uvrstiti dokument x. Preostane nam samo še računanje super povzetka. Za to uporabimo

algoritem 3.5.1, vendar izračunamo samo do povzetka y_i. Preostane nam samo še izračun take vrednosti h_ℓ, da bo veljajo:

$$RP_k = \begin{cases} H(h_\ell|y_i), b_\ell = L \\ H(y_i|h_\ell), b_\ell = R \end{cases}$$

Ob pogoju, da je zgoščevalna funkcija H brez trčenj takšnega h_ℓ ne moremo izračunati. Podoben sklep lahko naredimo za vsak nivo drevesa. Trditev 3.5.1 velja.

■

Pri tem načinu časovnega žigosanja pa se pojavi en problem. Vsi časovni žigi v istem ciklu imajo enak čas. To pomanjkljivost lahko odpravimo na dva načina. Prvi način od nas zahteva popolno zaupanje v TSA, da bo le-ta zahtevke v istem ciklu razporedila po istem vrstnem redu kot jih je prejela in tako ohranila relativni vrstni red med dokumenti v istem ciklu. Drugi način odpravi zaupanje v TSA, saj gre za mešanico linearnega povezovanja in akumuliranega drevesnega modela, tako da lahko določimo relativne položaje dokumentov v istem ciklu tudi brez zaupanja v TSA. Ta način presega obseg tega dokumenta. O njem si lahko bralec več prebere v [6].

Poglavje 4

Distribuirano časovno žigosanje

V prejšnjem poglavju smo se morali zanašati na TSA in na njene sposobnosti za pravilno časovno žigosanje dokumenta. Distribuirano časovno žigosanje pa je popolnoma drugačna rešitev problema, saj TSA ni pomemben. Imamo množico klientov $P = \{p_1, p_2, \ldots, p_m\}$, ki med seboj sinhrono komunicirajo in uporabljajo trojico protokolov $(\mathcal{S}, \mathcal{C}, \mathcal{V})$. Protokol za žigosanje $\mathcal{S}$ omogoča vsakemu udeležencu, da žigosa eno ali več sporočil. Protokol $\mathcal{C}$ za ugotavljanje avtentičnosti žiga izvaja klient, ki poskuša prepričati preverjevalca, ki uporablja potrditveni protokol $\mathcal{V}$, da je bilo sporočilo pravilno žigosano.

Takšna definicija je zelo splošna, saj omogoča kakršnekoli modele komunikacije in ne zahteva simetrije med protokoli, saj nekateri protokoli ne potrebujemo nobene interakcije med seboj. Dokumente časovno žigosamo samo s protokolom $\mathcal{S}$, šele ko se pojavi potreba po preverjanju časovnega žiga se uporabljata tudi protokola $\mathcal{C}$ in $\mathcal{V}$.

Udeleženci torej eden drugemu omogočajo časovno žigosanje. Tako kot pri časovnem žigosanju s pomočjo TSA, poznamo tudi pri distribuiranem žigosanju več metod [7], ki si jih bomo ogledali v nadaljevanju.

4.1 Zapomni si vse

Ta metoda zahteva od vsakega uporabnika, da pošlje svoj dokument (lahko tudi v šifrirani obliki ali povzetku) vsem ostalim udeležencem. Ostali si morajo dokument shraniti in zraven dodati še čas in datum, ko so ga prejeli. Za potrditev časovnega žiga mora sedaj katerikoli pošten udeleženec samo najti pravilni dokument in potrditi, da je bil res časovno žigosan ob določenem času. Če bi torej hoteli dokument napačno časovno žigosati, bi potrebovali sodelovanje vseh udeležencev, da bi nam pomagali pri tem.

Protokol za žigosanje $\mathcal{S}$ v tem primeru zahteva samo sprejem vse poslanih dokumentov ter hranjenje le-teh. Protokola $\mathcal{C}$ za ugotavljanje avtentičnosti žiga tukaj ni, potrditveni protokol $\mathcal{V}$ pa zahteva samo vpogled v vse shranjene dokumente. Čeprav je ta metoda zelo enostavna, lahko vsak udeleženec ob kateremkoli času pošlje poljuben dokument. To pa pomeni nesprejemljivo zahtevo po prostoru, saj bi moral vsak udeleženec shraniti čisto vse dokumente.

4.2 Naključne priče

Pri tej metodi predpostavljamo, da obstaja varni psevdo-naključni generator G, ki je na voljo vsem klientom. Psevdo-naključni generator je algoritem, ki iz poljubne začetne vrednosti vrne neko zaporedje, katerega je nemogoče izračunati v doglednem času s kakšnim drugim algoritmom.

Lahko rečemo, da je takšen generator nepredvidljiv. Takšni generatorji so opisani v [8] in [9]. Dokazali so, da takšni generatorji obstajajo, če obstajajo zgoščevalne funkcije brez trčenj [10].

Klient p ima sedaj povzetek $H(x) = y$, ki bi ga rad časovno žigosal. y uporabi kot začetno vrednost psevdo-naključnega generatorja G, ki nam vrne množico k števil. Te sedaj predstavljajo nekatere uporabnike v množici ($k < n$).

$$G(y) = (p_1, p_2, \ldots, p_k)$$

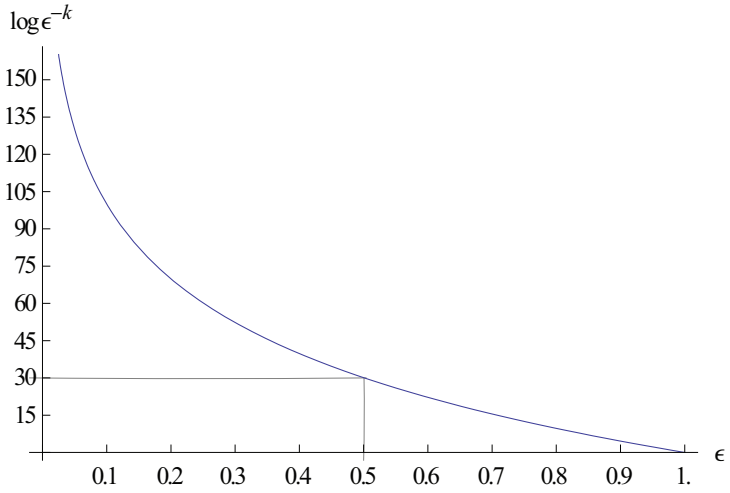
Klient sedaj pošlje (y, p) vsem k izbranim udeležencem. Od njih dobi nazaj podpisan dokument $s_j = \mathrm{sig}_j(t, p, y)$, kateremu so vsi dodali še čas t. Časovni žig dokumenta x bi imel sedaj obliko $[(y, p), (s_1, s_2, \ldots, s_k)]$, ki ga mora shraniti samo tisti, ki je zahteval časovno žigosanje. Protokol za žigosanje $\mathcal{S}$ zahteva od udeležencev sprejem dokumenta, kateremu doda čas t, ga podpiše s svojim privatnim ključem in pošlje nazaj kot s_j. Protokola $\mathcal{C}$ nimamo, potrditveni protokol $\mathcal{V}$ pa od udeležencev zahteva samo javne ključe, da lahko z dešifriranjem s_j potrdimo čas žigosanja dokumenta.

Trditev 4.2.1 *Za ponareditev časovnega žiga potrebujemo sodelovanje vseh $G(y) = (p_1, p_2, \ldots, p_k)$ izbranih udeležencev.*

Dokaz Dokaz trditve 4.2.1 je trivialen. Če nam ne bi pomagali vsi udeleženci, bi obstajal vsaj eden pošten udeleženec, ki bi preverjevalcu posredoval pravilni časovni žig in tako ustvaril dvom v verodostojnost žiga.

■

Kako pa lahko najdemo takšno začetno vrednost generatorja G, da nam bo vseh k udeležencev pomagalo pri napačnem žigosanju? Predpostavimo, da je delež takšnih klientov enak ε. Pričakovano število začetnih vrednosti, ki bi jih morali poizkusiti bi sedaj znašalo ε^{-k}. Ker smo predpostavili, da je G psevdo-naključni generator, ni hitrejšega načina kot naključno izbiranje med vsemi možnimi začetnimi vrednostmi.



Slika 4.1: Število različnih možnosti začetnih vrednosti generatorja G pri različnih vrednostih ε in $k = 100$.

Na sliki 4.1 vidimo, da četudi bi nam pomagalo ponarediti časovni žig polovica klientov in bi izbirali med $k = 100$ klienti bi še vedno morali najti eno izmed 10^{30} možnosti za pravilno začetno vrednost generatorja G in seveda najti tak dokument x katerega povzetek bi bil enak y. Ob uporabi zgoščevalne funkcije brez trčenj to seveda ni mogoče.

4.3 Povezovanje z uporabo ciklov

Povezovanje je postopek, kjer vsak uporabnik, ki bi rad časovno žigosal dokument, izračuna povzetek svojega dokumenta $y = H(x)$ in nato še trenutni super povzetek vseh uporabnikov $Y_i = H(y|Y_{i-1})$, kjer je Y_{i-1} prejšnji super povzetek vseh uporabnikov. Preverjanje veljavnosti časovnega žiga je lahko dolgotrajen postopek, saj moramo v nekaterih primerih preveriti in izračunati dolgo verigo, da pridemo do željenega super povzetka. Potrebujemo torej veliko stopnjo sodelovanja med uporabniki, saj lahko že pri nesodelovanju enega

uporabnika izgubimo možnost za testiranje žiga. Tukaj se pojavi potreba po uvedbi ciklov.

Med vsakim ciklom razpošljejo uporabniki povzetke svojih dokumentov vsem ostalim uporabnikom. Na koncu cikla vse te vrednosti združimo in izračunamo super povzetek cikla iz združenih vrednosti in super povzetka prejšnjega cikla. Protokol $\mathcal{S}$ zahteva od vsakega uporabnika, ki je v trenutnem ciklu oddal svoj dokument, da si zapomni tudi povzetke dokumentov ostalih uporabnikov. Vsi uporabniki pa si morajo zapomniti super povzetek cikla.

Da bi ugotovili ali je dokument pravilno časovno žigosan, uporabnik med protokolom $\mathcal{C}$ posreduje vse povzetke dokumentov v istem ciklu. Potrditveni protokol $\mathcal{V}$ zahteva od uporabnika, da izračuna super povzetek vseh dokumentov in ga primerja s super povzetkom tistega cikla.

4.4 Hierarhični sistem z uporabo ciklov

Hierarhični sistem je v primerjavi z ostalimi edini, ki je tako časovno kot prostorsko učinkovit. Če bi ga primerjali s centraliziranimi modeli bi bil najbolj podoben akumuliranemu drevesnemu povezovalnemu sistemu. Tako kot pri povezovanju tudi tukaj uporabljamo cikle, vendar sedaj vrednosti razvrstimo hierarhično, tako da se znebimo dodatne potrebe po prostoru.

V enem ciklu si moramo zapomniti samo $\mathcal{O}(\log m)$ vrednosti, kjer je m število sodelujočih. Tako kot prej imamo tudi zdaj super povzetek cikla, ki je sestavljen iz vseh trenutnih vrednosti in super povzetka prejšnjega cikla. Razlika je v tem, da m udeležencev sedaj razvrstimo v liste n-drevesa [11]. Oglejmo si sedaj kako bi izgledali naši $\mathcal{S}, \mathcal{C}, \mathcal{V}$ protokoli.

4.4.1 Protokol za žigosanje $\mathcal{S}$

1. V ciklu k zbere klient p_i vse svoje dokumente in skupaj s super povzetkom prejšnjega cikla R_{k-1} izračuna povzetek $h_{k,i}$. To nato pošlje vsem ostalim uporabnikom. Če klient v ciklu ne žigosa nobenega dokumenta uporabi vnaprej določeno vrednost R_0.

2. Vrednosti $h_{k,i}$ so nato razporejene v liste n-drevesa. Če kakšni listi ostanejo prazni se vanje zapišejo vnaprej določene vrednosti. Sedaj lahko izračunamo R_k, ki predstavlja super povzetek cikla k. To naredimo tako, da postopoma računamo povzetke vsakega starša (podobno kot v poglavju 3.5).

3. Vsak udeleženec p_i posebej izračuna celotno drevo in s tem tudi R_k, vendar si shrani samo svoje potomce in njihove brate.

4.4.2 Protokol za avtentičnost žiga $\mathcal{C}$

Da bi p_i dokazal, da je dokument žigosal v ciklu k mora:

1. dostaviti vse dokumente, ki jih je žigosal tisti cikel (seveda v šifrirani obliki ali povzetku).

2. dostaviti vse povzetke njegovih prednikov in njihovih bratov.

4.4.3 Potrditveni protokol $\mathcal{V}$

Da bi ugotovili, ali je bil dokument res žigosan v ciklu k moramo:

1. izračunati povzetek vseh dokumentov, ki nam jih je priskrbel p_i skupaj s super povzetkom cikla R_{k-1}. Preverimo če se ta vrednost ujema s $h_{k,i}$.

2. z danimi podatki izračunamo super povzetek cikla R_k in pri ostalih udeležencih preverimo ali se ujema z njihovo shranjeno vrednostjo.

Trditev 4.4.1 *Če sta izpolnjeni obe zahtevi protokola $\mathcal{V}$ (4.4.3), vemo da je bilo žigosanje res opravljeno v ciklu k in da je dokument pravilno žigosan.*

Dokaz Preverimo najprej prvo zahtevo. Izračunati moramo povzetek $h'_{k,i}$ z vsemi dokumenti, ki nam jih priskrbi p_i pri izvajanju protokola $\mathcal{C}$ (4.4.2) in super povzetkom R_{k-1}, ki ga lahko zahtevamo od poljubnega uporabnika. Ker nam p_i priskrbi tudi $h_{k,i}$, ga lahko tudi sam prej izračuna skupaj z dokumentom, ki ga je dodal že po žigosanju. Tako nam namesto $h_{k,i}$ pošlje nov $h'_{k,i}$ in preverjanje seveda uspe. Prva zahteva nam torej sama po sebi še ne zagotavlja potrditve, da je dokument pravilno žigosan

Če je p_i dodal kak dokument, potem $h'_{k,i} \neq h_{k,i}$ saj uporabljamo zgoščevalne funkcije brez trčenj. Tako moramo pri drugi zahtevi iz danih podatkov in $h'_{k,i}$ namesto $h_{k,i}$ izračunati super povzetek cikla R_k. To se izkaže za nemogoče in je že pokazano v dokazu trditve 3.5.1, točka 2. Trditev 4.4.1 velja.

■

Izkaže se, da je pri takšni definiciji optimalni delilni faktor našega drevesa $n = 3$. Če pa bi spremenili protokole, tako da bi si zapomnili samo brate naših prednikov, bi bil optimalni delilni faktor $n = 2$. Sedaj bi morali vrednosti naših prednikov izračunati kar iz danih podatkov.

Poglavje 5

Napadi

Obstajata dve glavni vrsti napadov na protokole časovnega žigosanja [12]. V prvem primeru poskuša napadalec dokument žigosati z datumom, ki se je zgodil pred originalnim žigosanjem. To je usodno za dokumente, kjer je veljaven najstarejši dokument[1]. Takšnemu tipa napada se reče "*back-dating attack*"[13]. Napadalec lahko sodeluje s TSA in poskuša ustvariti ponarejen, vendar veljaven časovni žig.

Pri drugi vrsti napadov pa napadalec poskuša datum časovnega žiga zamenjati s poznejši, tako da bo časovni žig še vedno veljaven. To je usodno za dokumente, kjer je veljaven najnoveši dokument[2]. Takšnemu tipu napada pravimo "*forward-dating attack*".

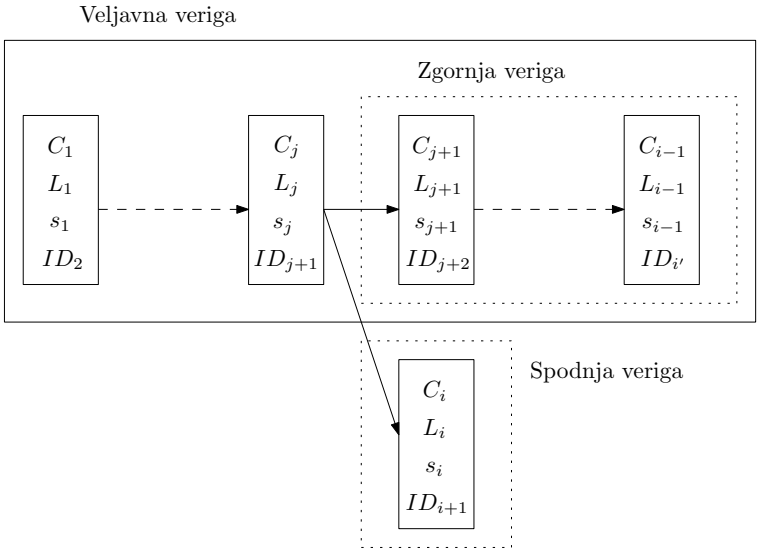
5.1 *back-dating attack*

Predstavljen je napad, kjer napadalec lahko sodeluje s TSA, ki mu pomaga delno vstaviti nov časovni žig v verigo že obstoječih (poglavje 3.3.1). V tem primeru je potrebna samo kratka nadomestna veriga, ki jo moramo vstaviti v obstoječo (samo začetek te nadomestne verige je povezan s pravilno). Na sliki 5.1 je spodnja veriga nadomestna. Ta napad kaže, da metoda 3.3.1 ne predstavlja potrebne mere varnosti, če ne moremo zaupati TSA. S pomočjo TSA lahko napadalec vstavi svoj dokument v že obstoječo verigo časovnih žigov.

Uporabnik p_i mora torej sodelovati s TSA, da bi svoj dokument y_i (katerega časovni žig je s_i) napačno žigosal s preteklim datumom. Če pogledamo sliko

[1]Recimo, ko gre za varovanje intelektualne lastnine.

[2]Recimo, ko imamo opravka z digitalnimi oporokami.



Slika 5.1: Razcep v verigi dokumentov. Vsak od manjših pravokotnikov predstavlja časovni žig uporabnika. Veljavna veriga predstavlja sliko ob normalnem izvajanju linearnega povezovalnega modela.

5.1 vidimo, da bi moral žig s_i po pravilih biti na koncu verige, takoj za s_{i-1}. Vendar je na sliki takoj za s_j. Kako nam to pomaga? Predvidevajmo, da žigi s_j, s_{j+1} in s_{i-1} vsebujejo čase t_j, t_{j+1} in t_{i-1}, kjer velja $t_j < t_{j+1} < t_{i-1}$. Če bi hoteli sedaj žig s_i pravilno umestiti v verigo (takoj za s_{i-1}) bi mu morali določiti čas $t_i > t_{i-1}$. V resnici pa ga umestimo takoj za s_j in mu lahko določimo katerikoli čas, tako da velja $t_i > t_j$. Ker je sedaj $t_i < t_{i-1}$ smo dokument žigosali s preteklim datumom.

Vse naslednje časovne žige lahko zdej uvrstimo v verigo na dva načina. Prvi način je, da vse naslednje dokumente dodamo spodnji verigi (takrat bi $\mathrm{ID}_{i'}$ pri žigu s_{i-1} nadomestili z ID_i). Pri drugem načinu pa bi žige izmenično dodajali spodnji in zgornji verigi (sedaj bi $\mathrm{ID}_{i'}$ pri žigu s_{i-1} nadomestili z ID_{i+2}. Ta način je boljši, saj omogoča preverjevalcu, da preveri tudi žige, ki so na desni strani v verigi.

Sedaj ko smo umestili žig s_i v verigo, predvidevajmo da ga želi nekdo preveriti. Če bi preverjali žige, ki so desno v verigi, ne bi odkrili nič napačnega, če pa bi se odločili preverjati žige na levi strani verige, bi videli da ima prejšnji časovni žig s_j zaporedno številko ID_{j+1}, preverjevalec pa pričakuje ID_i. Napadalcu še vedno preostane nekaj možnosti:

1. TSA in p_i lahko sodelujeta tudi s p_j in mu določita zaporedno številko

ID_i. Za to moramo seveda izračunati kolizijo zgoščevalne funkcije.

2. Rečemo, da je $\mathrm{ID}_{j+1} = \mathrm{ID}_i$. To lahko dosežemo s tem, da p_i periodično podpisuje neke (lahko čisto nepomembne) dokumente. Tako lahko p_i s pomočjo TSA napačno žigosa svoj dokument povsod kjer ima svoje podtaknjene časovne žige.

Druga možnost je seveda dosti boljša, saj ni potrebno računati kolizij. Tako smo dokument napačno časovno žigosali s predhodni datumom.

5.2 *forward-dating attack*

Naslednji napad je možen, če TSA uporablja metodi, ki sta opisani v 3.2 in 3.3.1. Pri napadu ne potrebujemo sodelovanja TSA in ni potrebno poznati vsebino datoteke. Uporabljamo ga lahko za recimo elektronske oporoke, ko je veljavna zadnja verzija oporoke. Recimo, da odvetnik časovno žigosa prvo verzijo oporoke. Čez nekaj časa pa se pojavi sprememba in ponovno žigosa novo oporoko, ki pa je za napadalca slabša kot prva. Napadalec deluje kot vmesni poslušalec in posname vse seje med TSA in odvetnikom. Ko se pojavi druga verzija oporoke, lahko uporabi posneto sejo prvega žigosanja in jo še enkrat pošlje TSA. Tako v resnici zgleda, kot da je prva verzija tudi zadnja, ki je bila časovno žigosana.

Kako se lahko zavarujemo pred takim napadom? Imamo več možnosti:

- TSA lahko pred žigosanjem pošlje klientu naključno številko r. Klient nato nazaj pošlje napadalcu povzetek y in $\mathrm{sig}_K(y|r)$. Napadalec sedaj ne more izračunati pravilnega odgovora.

- TSA in klient se dogovorita za naključni enkratni sejni ključ (npr. SSL), s katerim je kriptirana celotna seja.

- TSA in klient opravita avtentikacijo. TSA nato določi enkratni ID klienta, ki ga vključita v časovni žig.

Vidimo, da se v resnici ni težko zavarovati pred takšnim napadom, saj se vse te rešitve že uporabljajo tudi pri vsakdanjih varnih povezavah.

5.3 Kompromitirana zgoščevalna funkcija ali digitalna shema

Tema ne spada ravno med napade na digitalno časovno žigosanje, vendar je vseeno vredna omembe. Ob primeru kompromitiranja ene ali druge funkcije postane naš časovni žig neveljaven, saj ga lahko v tem primeru nekdo enostavno ponaredi.

Kako smo torej lahko prepričani, da bosta čez recimo 10 let naša osnovna zgoščevalna funkcija in digitalna shema še vedno dovolj varni za uporabo? Ena od rešitev je predčasna zamenjava funkcije, če vemo, da bo kmalu postala nezanesljiva. Takrat lahko TSA vse časovne žiga ponovno žigosamo z novo, bolj varno shemo. Vendar pa se lahko zgodi, da nezanesljivosti določene sheme ne moremo napovedati.

Da naši časovni žigi ne bi bili ogroženi tudi v takšnih situacijah, lahko TSA uporabi sočasno žigosanje z uporabo več različnih funkcij. TSA torej izbira med množico zgoščevalnih funkcij in digitalnih shem. Za vsak podpis naključno izbere dve različni zgoščevalni funkciji in dva različna digitalna podpisa. Tako za vsak dokument izda dva časovna žiga. Če je sedaj ena od shem kompromitirana in s tem tisti časovni žig neveljaven, ima TSA na razpolago še en časovni žig, s pomočjo katerega lahko obnovi neveljavnega z uporabo nove sheme.

Poglavje 6

Uporaba časovnih žigov

Potrebnost časovnega žigosanja se zazna vsakič, ko je v realnem svetu za določen dokument nujno potreben časovni zaznamek. To pa je zelo pogosto. Eden še posebaj pomembnih primerov, ko je zapis časa ključen element, je intelektualna lastnina in industrijska lastnina, torej nek dokument v elektronski obliki, ki predstavlja obstoj ideje ali izuma v določenem trenutku. Pogosta je tudi situacija, ko za določene dokumente v danem trenutku ni pokazana potreba po časovnem zaznamku ampak je za njih bistven podatek o verodostojnosti. Primer je lahko podjetje, ki vsak dan kreira večje število dokumentov, ki v danem trenutku nimajo neke kritične vrednosti, se pa kasneje taka situacija lahko pojavi. Brez ustrezne zaščite se lahko take dokumente popravi ali zbriše. V primeru ustreznega časovnega žigosanja takih problemov ni.

Danes obstajajo na svetu različna podjetja, ki ponujajo storitev časovnega žigosanja. Nekatere med njimi so na kratko predstavljena v nadaljevanju.

6.1 Aplikacije in storitve na internetu

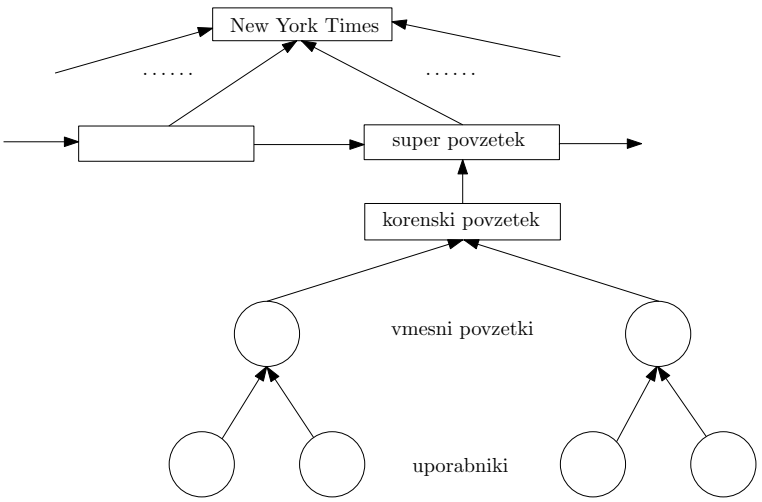
Pri implementaciji storitve časovnega žigosanja si podjetja postavljajo zahteve kot so:

- dostopnost preko interneta,
- zanesljivost, varnost, skalabilnost,
- usklajenost s standardi,
- cenovna sprejemljivost,
- možnost integracije.

Mnoga podjetja svojo storitev časovnega žigosanja ponujajo kot notarsko storitev in tudi v raznih drugih publikacijah se pojem notariata pogosto pojavlja. Izraz se lahko razume kot overjanje določenega podpisa v določenem trenutku in za določeno vsebino s strani tretje osebe. Podobno kot notar s svojim žigom in podpisom overi podpis uporabnika v določenem trenutku, tako podjetje, ki ponuja storitev časovnega žigosanja, s svojim podpisom in dodanim časovnim žigom v določenem trenutku lahko overi veljavnost podpisa povzetka določene vsebine, ki jo uporabnik pošlje na overjanje.

Surety [http://www.surety.com] je podjetje, ki ponuja storitev digitalnega časovnega žigosanja že od leta 1992. Ponujajo aplikacijo katero si uporabnik namesti na lokalni računalnik in z njo enostavno časovno žigosa različne elektronske podatke, tekstovne podatke, audio in video podatke, s pomočjo dodatnega razvojnega orodja pa lahko storitev časovnega žigosanja uporabljamo tudi v svojih aplikacijah.

Sama storitev temelji na akumuliranem drevesnem povezovalnem modelu. Dolžina cikla je 1 sekunda. Torej vsako sekundo dobijo korenski povzetek in s pomočjo super povzetka prejšnjega cikla naredijo super povzetek trenutnega cikla. Te so shranjeni v univerzalnem registru, do katerih lahko dostopajo tudi preverjevalci. Za dodatno varnost enkrat tedensko izračunajo povzetek vseh 604800 super povzetkov posameznih sekundnih ciklov in ga objavijo v reviji New York Times.



Slika 6.1: Sistem digitalnega časovnega žigosanja podjetja Surety

Stamper [http://www.itconsult.co.uk/stamper.htm] je brezplačna storitev digitalnega žigosanja, ki temelji na elektronski pošti. Če hoče uporabnik časovno žigosati dokument ga pošlje na text@stamper.itconsult.co.uk in dobi povratno elektronsko pošto, ki vsebuje časovni žig. Časovni žig lahko nato preverimo s poslano elektronsko pošto na naslov post@stamper.itconsult.co.uk.

Dana rešitev z določenega vidika ni dovolj varna, saj jim razkrijemo vsebino sporočila, vendar ga lahko sami že predhodno kriptiramo ali naredimo njegov povzetek. Po drugi strani pa je storitev primerna za ljudi, ki računalništva in modernih tehnologij ne poznajo dobro in lahko uporabijo storitev časovnega žigosanja samo s svojim znanjem o elektronski pošti.

e-TimeStamp [http://www.digistamp.com] je, tako kot Surety, podjetje, ki se profesionalno ukvarja z digitalnim časovnim žigosanjem. Tudi oni imajo svojo aplikacijo s pomočjo katere lahko uporabniki žigosajo svoje dokumente. Aplikacija pošlje povzetek (uporabljajo SHA-256 ali SHA-512) dokumenta, ki ga želimo žigosati njihovemu strežniku. Ta žigosa prejeti povzetek, ga podpiše z njihovim privatnim ključem (dolžine 2048 bitov) in žig pošlje nazaj uporabniku. Vidimo, da tukaj potrebujemo brezpogojno zaupanje v TSA, kot je opisano v 3.2.

timeMaker [http://www.timemarker.org/en] je še ena izmed oblik brezplačnega časovnega žigosanja, ki se uporablja v današnjih dneh. Dokumente lahko žigosamo kar preko obrazca na njihovi strani. Po končanem si lahko tudi sami shranimo časovni žig našega dokumenta. Tudi preverjanje dokumentov poteka kar preko obrazca na spletni strani. Po končanem nalaganju datoteke, jo poiščejo v svoji bazi in nam dostavijo sporočilo o obstoju časovnega žiga za njo.

Vidimo, da tudi ta storitev mogoče ni najbolj varna, saj podatke pošiljamo preko interneta, kjer s prisluškovanjem lahko do njih dostopajo tudi ostali. Dostop do podatkov imajo tudi oni sami, tako da moramo zaupati tudi njim.

Omenjena sta dva brezplačna in dva plačljiva sistema časovnega žigosanja. Vidimo, da so razlike med njimi kar velike in da je vsak sistem implementiran na svoj način. Opazimo pa tudi, da moramo za kvalitetno storitev plačati, saj so brezplačne rešitve vprašljive glede varstva podatkov.

Poglavje 7

Zaključek

Digitalno časovno žigosanje je zbirka mehanizmov, ki zagotavljajo dolgoročno verodostojnost digitalnih dokumentov skupaj s časom, ki pove, kdaj je tak zapis nastal. Predstavljena sta bila dva različna modela časovnega žigosanja, centralizirano in distibuirano. Pri obeh smo skozi različne primere postopoma prišli do optimalne rešitve, ki je podobna za oba problema.

Marsikdo bi se sedaj vprašal, kateri model je boljši, centralizirani ali distribuirani. Odgovor na to vprašanje ni trivialen in ga pogojuje kar veliko faktorjev. Če gledamo iz stališča varnosti je distribuiran model boljši, saj imamo veliko množico uporabnikov, kateri nam zagotavljajo verodostojnost časovnega žiga, medtem ko moramo pri centraliziranem zaupati TSA tudi pri bolj naprednih rešitvah (napad opisan v 5.1).

Vendar pa je v splošnem rešitev s TSA bolj popularna in razširjena, na kar kažejo tudi primeri v poglavju 6.1. V resnici je težko dobiti dovolj veliko množico uporabnikov za distribuirano časovno žigosanje, ki nam bi omogočala visoko stopnjo zaupanja. Tudi v primeru sporov na sodišču so agencije za digitalno časovno žigosanje boljša rešitev, saj je zaupanje na plečih ene same organizacije. Podjetja, ki se ukvarjajo profesionalno s časovnim žigosanjem, torej dobro skrbijo za varnost svojih časovnih žigov (lep primer je Surety).

Ker lahko z digitalnim žigosanjem določimo tako avtorja, kot čas nastanka dokumenta,se zdi, da bo časovno žigosanje v prihodnosti uporabljeno za najrazličnejše namene. Od uporabe pri dolgoročnih pogodbah do uporabe pri osebnih dnevnikih in pismih. Papirnati svet se počasi spreminja v elektronskega. Če danes zgodovinar odkrije knjigo iz obdobja starih Egipčanov, je verodostojnost ugotovljena iz fizičnih lastnosti najdbe. Čez 100 let pa bo pri podobni najdbi, s podatki v računalniški obliki, lahko čas nastanka ugotovljen in dokazan samo na podlagi ustreznega verodostojnega časovnega žiga.

Literatura

[1] Stuart A. Haber and Wakefield Scott Stornetta. *How to time-stamp a digital document*. Journal of Cryptology, 3(2):99-111, 1991.

[2] Alfred P. Menezes, Paul C. Van Oorschot, and Scott A. Vanstone. *Handbook od Applied Cryptography*. CRC Press, 1996.

[3] D. Bayer, S. Haber, and W.S. Stornetta. *Improving the efficiency and reliability of digital time-stamping*. In R.M. Capocelli, A. De Santis, and U. Vaccaro, editors, Sequences II: Methods in Communication, Security and Computer Science, pages 329-334. Springer Verlag, New York, 1993.

[4] S. Haber and W.S. Stornetta. *Secure names of bit-strings*. In Procedings of the 4th ACM Conference on Computer and Communication Security, pages 28-35. ACM Pres, April 1997.

[5] H. Massias and J.-J. Quiquater. *Time and cryptography*. Tehnical report, TIMESEC Project (Federal Government Project, Belgium), 1997.

[6] A. Buldas, P. LAud, H. Lipmaa, J. Villemson, *Time-stamping with binary linking schemes*. Advances in Cryptology - Crypto '98, LNCS 1462, H. Krawczyk, Ed., Springer-Verlag, pp. 486-501, 1998.

[7] J. Benaloh, M. de Mare. *Efficient roadcast time-stamping*. Clarkson University, Department of Math and Computer Science. TR 91-1, August 1991.

[8] M. Blum and S. Micali. *How to generate cryptographically strong sequences of pseudo-random bits*. SIAM Journal on Computing, 13(4):850-864, Nov. 1984.

[9] A.C.Yao. *Theory and applications of trapdoor functions*. In Proc. 23rd FOCS, pp. 80-91. IEEE, 1982.

[10] R. Impagliazzo, L. Levin, and M. Luby. *Pseudorandom generation from one-way functions*. In Proc. 21st STOC, pp. 12-24. ACM, 1989.

[11] N-ary Trees, http://www.brpreiss.com/books/opus5/html/page257.html (12.6.2010)

[12] T. Matsuoa and S. Matsuo. *On universal composable security of time-stamping problem*. Applied Public Key Infrastructure: 4th International Workshop: IWAP 2005, pp. 169-181, 2005.

[13] M. Just. *Some timestamping protocol failures*. In Proc. Symposium on Network and Distributed Security (NDSS98), 1998.