Hyväksymispäivä Arvosana

Arvostelija

Filtering for Private Collaborative Benchmarking

Seppo Koljonen

Helsinki 12.4.2007

Monen osapuolen tietoturvaprotokollat -seminaari, kevät 2007

HELSINGIN YLIOPISTO

Tietojenkäsittelytieteen laitos

HELSINGIN YLIOPISTO – HELSINGFORS UNIVERSITET

Tiedekunta/Osasto	Laitos – Institution
Matemaattis-luonnontieteellinen	Tietojenkäsittelytieteen laitos

Tekijä – Författare
Seppo Koljonen

Työn nimi – Arbetets titel
Filtering for Private Collaborative Benchmarking

Oppiaine – Läroämne
Tietojenkäsittelytiede

Työn laji – Arbetets art	Aika – Datum	Sivumäärä – Sidoantal
Seminaari	12.4.2007	17

Tiivistelmä – Referat

Monen osapuolen suorituskyvyn vertailu on tärkeä osa nykymuotoisten yritysten toimintaa. Suorituskyvyn vertailu tarkoittaa sitä, että yritykset vertailevat omia suorituskykymittareitaan (esimerkiksi tuotteen toimitusaika, kassavirta tai sijoitetun pääoman tuottoprosentti) suhteessa toisten yritysten arvoihin. Ongelmana on ollut se, että nämä luvut ovat hyvin luottamuksellisia, eivätkä saa joutua toisten yritysten tietoon. Turvallinen monen osapuolen laskenta (SMC) tarjoaa joukon protokollia ongelman ratkaisemiseen.

Tässä esitelmässä esitetään protokolla, jota voidaan käyttää suodattimena ennen varsinaisia suorituskykyvertailun suorittavia protokollia. Pohjana lopulliselle suodatusprotokollalle esitetään vertailuportit järjestävässä verkossa, permutaatio-, ja sekoituspermutaatioprotokollat ja suodatuksen jälkeen tapahtuvaa suorituskykyvertailua varten on esitetty tehokkain tunnettu Yao'n miljonäärit protokollan toteutus.

Avainsanat – Nyckelord

Säilytyspaikka – Förvaringställe

Muita tietoja – Övriga uppgifter

SISÄLTÖ

1 Johdanto

Suorituskyvyn mittaaminen yrityksissä on johdon prosessi, jossa yritykset vertaavat oman yrityksensä suoriutumista markkinoilla suhteessa toisiinsa erilaisin mitattavissa olevin tilastollisin määrein. Kun useat yritykset ottavat osaa tähän prosessiin yhtä aikaa, voidaan katsoa, että kyseessä on monen osapuolen suorituskyvyn mittaaminen. Suorituskyvyn mittareita yrityksissä voivat olla esimerkiksi tuotteen toimitusaika, kassavirta tai sijoitetun pääoman tuottoprosentti (ROI).

Ongelmana monen osapuolen suorituskykyvertailuissa on se, että nämä määreet ovat yleensä hyvin luottamuksellisia, eikä niitä haluta saattaa vertailun takia julkisiksi. Tähän asti ratkaisut ovat perustuneet luotettavan kolmannen osapuolen käyttämiseen ja vertailutietojen anonymisoimiseen. Toisin sanoen poistetaan tieto, mitkä luvut ovat lähtöisin mistäkin yrityksestä.

Turvallinen monen osapuolen laskenta SMC (*Secure Multi-Party Computation*) mahdollistaa näiden suorituskykymääreiden laskemisen ja osapuolten välisen vertailun ilman kolmatta osapuolta, niin että yritykset eivät silti saa tietoa, mikä arvo on peräisin miltäkin yritykseltä tai mitä muut vertailtavat arvot ovat. Suorituskykyvertailussa on myös huomattava, että joidenkin yritysten suorituskykymääreet voivat poiketa niin paljon keskiarvosta, että niiden ottaminen mukaan vertailuun voisi vääristää tuloksia. Siksi on tärkeää rajata vertailu vain *k*-parhaisiin arvoihin.

Kerschbaum ja Terzidis esittävät artikkelissaan [KT06] yleisen monen osapuolen suodatusprotokollan, jota voidaan käyttää ennen varsinaisia suorituskyvyn laskevia ja tulokset k-parhaisiin arvoihin rajoittavia protokollia.

Jokaisella vertailun osapuolella on suorituskykyvertailua varten yksi lähtöarvo ja kun vertailu on suoritettu, jokainen osapuoli on saanut itselleen järjestysluvun, joka kertoo sen aseman verrattuna muihin. Suodatusprotokollaa tarvitaan järjestämään osapuolten lähtöarvot tietoturvallisesti niin, että varsinainen suorituskykyvertailu voi alkaa.

Vaatimuksia suodatusprotokollalle:

1) Monen osapuolen protokolla, jossa jokaisella osapuolella on vain yksi vertailun lähtöarvo
2) Mikään osapuoli ei saa tietää tai oppia mitään muiden osapuolten luvuista
3) Mikään osapuoli ei saa tietää, kuuluuko sen vertailuluku k-parhaiden lukujen joukkoon

Luvussa 2 esitellään aiheeseen liittyvää edeltävää tutkimusta. Luvuissa 3 ja 4 esitellään miten suorituskykyarvoja voidaan vertailla järjestävän verkon ja Yao'n miljonäärit protokollan avulla. Luvut 5 ja 6 esittävät permutaatio ja sekoituspermutaatioprotokollat, joita käytetään hyväksi lopullisessa suodatusprotokollassa luvussa 7. Luku 8 on varattu yhteenvedolle.

2 Edeltävä tutkimus

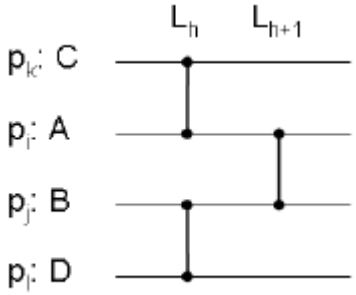
Turvallisesta monen osapuolen laskennasta on jo olemassa useita yritysmaailmaan liittyviä sovelluksia. Esimerkiksi toimitusketjun hallintaan on kehitetty protokolla, joka estää tavarantoimittajia saamasta selville jälleenmyyjien liikevoittoa. Varsinaista suorituskykyvertailua varten on tosin kehitetty vain yksi protokolla, jossa käytetään hyväksi jakolaskun laskemista salaisella jakajalla. Tämän protokollan laajennuksia voidaan käyttää suorituskyvyn laskemiseen tässä esitelmässä esitetyn suodatusprotokollan jälkeen.

Lukujen yksityisen järjestämisen ongelmaan on olemassa erilaisia protokollia, kuten protokolla joka laskee kahden yhteenlaskuksi hajotetun vektorin huippuarvon ja protokolla, joka suorittaa binaarijärjestämis-algoritmin (binary sort) yksityisesti.

Muita tärkeitä protokollia ja tekniikoita suorituskykyvertailulle ja tässä esitetylle suodatusprotokollalle ovat sekoittavat verkot (*mix networks*), julkisen avaimen homomorfiset salaukset ja järjestävät verkot (*sorting network*).

3 Vertailuportit järjestävässä verkossa

Suodatusprotokolla emuloi järjestäviä verkkoja, joissa jokainen osapuoli on yhdistetty yhteen sisään tulevaan langoitukseen (*input wire*). Järjestäminen etenee kerros kerrokselta ja saman kerroksen vertailut voidaan suorittaa yhtä aikaa. Osapuolet vertailevat keskenään omia lukujaan ja vaihtavat ne keskenään jos tarvetta. Tehokkaimmat järjestävät verkot ovat $O(\log^2 n)$ syvyisiä. Viestinvaihdon monimutkaisuus on $O(n \log^2 n)$ ja suoritus päättyy $O(\log^2 n)$ kierroksella. Vertailuportteja edeltää salainen, osapuolten järjestyksen permutoiva protokolla, joka esitellään tarkemmin luvussa 7. Vertailuporttien tapauksessa lukujen vertailuun voidaan käyttää Yao'n miljonäärit protokollaa, joka piilottaa osapuolten vertailemat luvut toisiltaan.



Kuva 1: Osa järjestävää verkkoa

Kuvassa 1 on esitetty osa järjestävää verkkoa.

1) Olkoon p_1 ,…, p_n protokollan osapuolia ja $x = (x_1,\ldots,x_n)$ näiden osapuolien vertailuarvoja. Jokaisella osapuolella p_i on yksi vertailuarvo x_i, kaikilla $i = 1, \ldots, n$.
2) Lisäksi oletetaan, että $i < j$, $k \neq j$, $i \neq l$, $k < i$ ja $j < l$

Käytetään julkisen avaimen, semanttisesti turvallista, homomorfista salausta, jolloin $E_A(\cdot)$ on vertailuluvun salaus Alicen julkisella avaimella ja $D_A(\cdot)$ tämän salauksen purku Alicen yksityisellä avaimella. Oletetaan myös, että julkinen avain on kaikkien osapuolten tiedossa.

Salauksen homomorfisuudesta seuraa, että on olemassa operaatio X siten, että $E_A(x)$ X $E_A(y) = E_A(x+y)$ ja operaatio *, siten että $E_A(X)^*$ $y = E_A(x*y)$. Toteutus perustuu julkista avainta käyttävään Paillier'in salaukseen, jolloin X on modulaarinen kertolasku ja * modulaarinen potenssiinkorotus.

Alaindeksi [CA] tarkoittaa Charlien (C) ja Alicen (A) välistä vertailua ja $r_{[ca]}$ on tästä vertailusta saatu tulos. Samoin esim. [BD] tarkoittaa Bobin ja Donnan välistä vertailua ja $r_{[bd]}$ tämän vertailun tulosta.

3.1 Protokolla

Kuvassa 2 on vertailuportteja ja järjestävää verkkoa käyttävän protokollan täsmällinen määritelmä. Alicella on aluksi arvo **a** ja Bobilla arvo **b.** Vertailun tarkoituksena on laskea a < b ja lopuksi vaihtaa arvoja, jos tarpeellista. Yksityisyysvaatimuksena protokollan käytölle on se, että Alice tai Bob eivät saa tietää tai oppia mitään toistensa luvuista, joten vertailun lähtöarvot a ja b on jaettu kahteen osaan niin, että kumpikaan ei saa tietää, kuin osan omasta ja osan toisen osapuolen luvusta. Alicella on osa omasta vertailuluvustaan (a_A) ja osa Bobin vertailuluvusta (b_A). Samoin Bobilla on osa omasta luvustaan (b_B) ja osa Alicen luvusta (a_B).

Alicella on luvut a_A ja b_A eli $a = a_A + a_B$
Bobilla on luvut a_B ja b_B eli $b = b_A + b_B$

Vertailu voidaan jaosta huolimatta suorittaa seuraavasti
$a_A + a_B < b_A + b_B \Leftrightarrow a_A - b_A < b_B - a_B$.

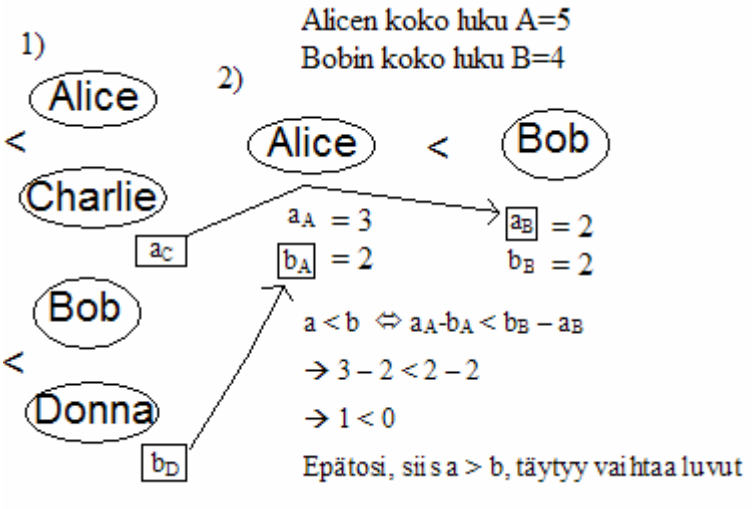
Yhden vertailuaskeleen suorittamiseksi pitää mukaan ottaa edellisellä kerroksella (kuvassa 1 kerros l_h) tehdyt vertailut. Aluksi on siis jo tehty vertailu c < a ja b < d, josta Charliella on jäljellä osa Alicen vertailuluvusta (a_C) ja Donnalla on jäljellä osa Bobin vertailuluvusta (b_D). Seuraavaksi Charlie lähettää osansa Alicen luvusta Bobille ja Donna osansa Bobin luvusta Alicelle.

Participants	Message / Operation
C ⟶ B	$a_B = a_{C[CA]}$
D ⟶ A	$b_A = b_{D[BD]}$
A ⟷ B	$\rho = \mathrm{Yao}(a_A - b_A, b_B - a_B)$
A	if $\neg\rho$ then $\mathrm{swap}(a_A, b_A)$
B	if $\neg\rho$ then $\mathrm{swap}(a_B, b_B)$

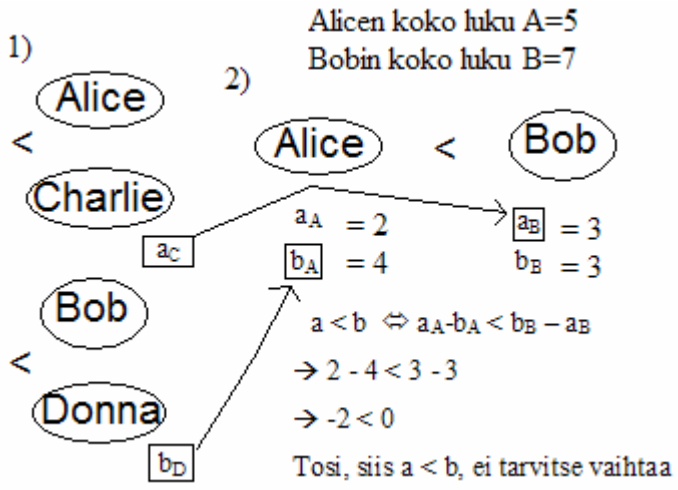
Kuva 2: Protokollan täsmällinen määritelmä

3.2 Vertailuporttien esimerkki

Kuvissa 3 ja 4 on esitetty kaksi sovellettua käytännön esimerkkiä protokollan toiminnasta, joista selviää lukujen jakaminen ja osapuolten välinen viestinvaihto. Esimerkissä Alicen ja Bobin vertailu ei vielä käytä YAO'n miljonäärit protokollaa, joten Alice lähettää vain omien lukujensa erotuksen Bobille ja Bob omansa Alicelle. Näin ne voivat saada vielä selville toistensa luvut.



Kuva 3: Esimerkki, kun A > B



Kuva 4: Esimerkki, kun A < B

4 YAO'n miljonäärit protokolla

Edellisessä luvussa 3 nähtiin, että tarvitaan jokin salausmekanismi, jotta Alicen ja Bobin välisessä vertailussa ne eivät voi saada selville toistensa lukuja. Yao'n miljonäärit

protokollan tarkoituksena on piilottaa luvut niin, että lukujen selvittäminen on mahdotonta.

Tässä käytetään erityistä satunnaista piilotustekijää (hiding factor) **r**, jolla vertailuluku kerrotaan. Jos vertailtava luku on O(m) kokoinen, täytyy satunnaisen piilotustekijän olla $O(m^2)$ suuruinen. Jatkossa halutaan säilyttää suurempi kuin vertailu ja estää syklinen/kiertävä ”wrap-around” modulo n. Negatiivisia lukuja ei voi esittää moduloaritmetiikassa, joten sen vuoksi määritellään arvoalueen [0, n-1] ylempi puoli negatiivisiksi luvuiksi.

$$[\lceil \frac{n}{2} \rceil, n-1] \equiv [-\lfloor \frac{n}{2} \rfloor, -1]$$

Lukualueen yläosa vastaa siis negatiivisia lukuja. Esimerkiksi n=10 [5,9] -> (kuvautuu) [-5, -1] eli nyt [0..5, -5..-1].

Koska piilottaminen tapahtuu kertolaskun avulla, törmätään ongelmaan jos luvut a ja b ovat yhtä suuria, koska tällöin myös piilotuksen tulos on 0. Tämän takia tarvitaan toinenkin piilotustekijä **r’**, joka on pienempi, kuin **r.** Piilotustekijöiksi valitaan kaksi satunnaislukua, niin että $0 \leq r' < r$.

1. Alice sends $E_A(a)$ to Bob.
2. Bob chooses random numbers r and r' with $0 \leq r' < r$.
3. Bob computes $E_A(c) = E_A(a)^r \cdot E_A(-r \cdot b + r') = E_A(r \cdot a - r \cdot b + r')$.
4. Bob sends $E_A(c)$ to Alice.
5. Alice decrypts $c = D_A(E_A(c))$ and decides $a < b$ if and only if $c \geq \frac{n}{2}$. The following derivation shows this equivalence:

$$c \bmod n \geq \frac{n}{2}$$

$$c < 0$$

$$r \cdot (a - b) + r' < 0$$

$$a - b \leq -1 < -\frac{r'}{r} < 0$$

6. Alice sends the bit $a < b$ to Bob.

Kuva 5: Protokollan täsmällinen määritelmä

Kuvassa 5 on esitetty Yao’n miljonäärit protokollan täsmällinen määritelmä. Alice lähettää ensin oman vertailulukunsa ($a = a_A - b_A$) salattuna julkisella avaimella Bobille. Tämän jälkeen Bob valitsee kaksi satunnaista piilotustekijää **r** ja **r’**. Bob suojaa oman

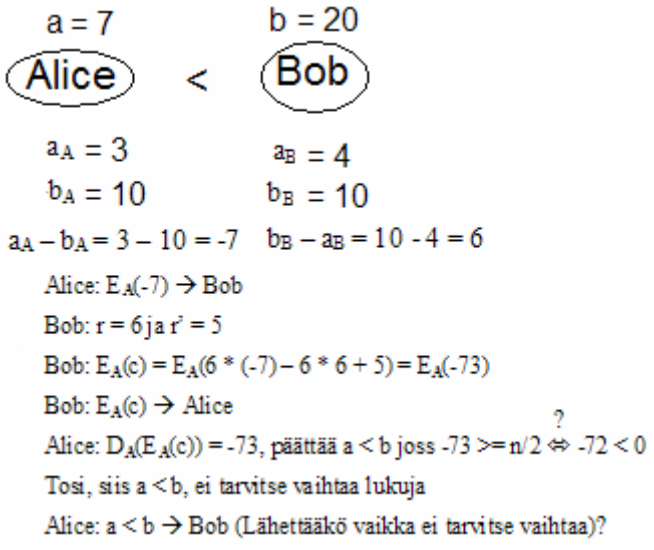
vertailulukunsa b ($b=b_B - a_B$), piilotustekijöiden avulla kuvan 5 esittämällä kaavalla ja lähettää tuloksen takaisin Alicelle. Alice purkaa Bobilta saadun viestin $E_a(c)$ omalla yksityisellä avaimellaan ja päättää tuloksena saadun luvun perusteella a < b jos ja vain jos c >= n/2 (negatiivinen arvo). Lopuksi Alice lähettää tiedon vertailun tuloksesta Bobille. Osapuolten ei tarvitse lähettää omia lukuja toisilleen, vaan riittää, että molemmat vaihtavat oman puolensa osajakoja.

Bob ei saa tietää mitään Alicen luvusta, koska se on salattu julkisen avaimen salauksella ja Bob ei tiedä purkamiseen vaadittavaa Alicen yksityistä avainta. Toisaalta Bobin Alicelle palauttama luku b on suojattu piilotustekijöillä. Vaikka molemmat lähettävät vertailuluvut toisilleen, kumpikaan ei saa tietää mitään toistensa oikeista vertailuluvuista toisin kuin kuvien 3 ja 4 esimerkeissä. Tämä ei vielä riitä, sillä sekä Alice, että Bob oppivat vielä toistensa keskinäisen järjestyksen. Tämä ongelma poistetaan luvuissa 5 ja 6 esitettyjen permutaatio- ja suodatusprotokollien avulla.

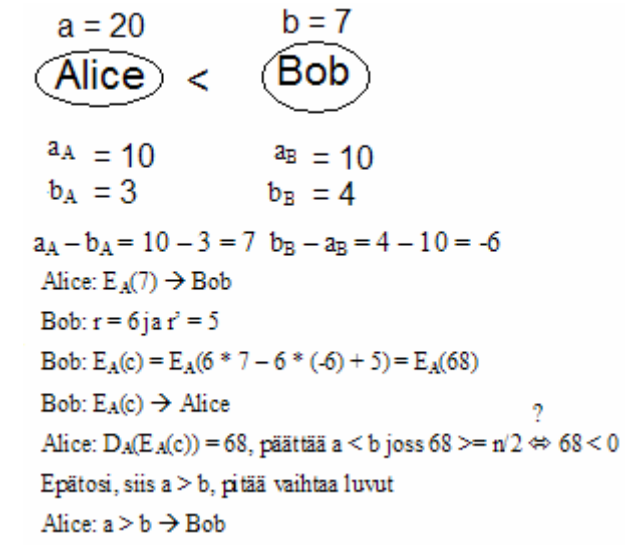
Protokollassa on vähäinen aukko, jonka avulla Alice voi saada tietoonsa, että molemmat vertailuluvut ovat samoja (tarkempi todistus sivuutettu). Esimerkiksi vertailtaessa 160-bittisiä lukuja todennäköisyys tälle tapahtumalle on $p < 2^{-314}$.

4.1 Yao’n miljonäärit protokollan esimerkki

Seuraavalla sivulla olevissa kuvissa 6 ja 7 on esitetty kaksi soveltavaa esimerkkiä Yao’n miljonäärit protokollan käytöstä kahdessa erilaisessa vertailutilanteessa. Esimerkeissä ei selvyyden vuoksi käytetä varsinaista julkisen avaimen salausta, joten Alicen lähettämä vertailuarvo on salaamaton ja Bob saa sen selville. Esimerkki näyttää piilotustekijöiden r ja r’ hyödyn, sillä kun Bob lähettää oman lukunsa Alicelle, Alice ei voi enää helposti johtaa B:n lukua tietämättä piilotustekijöitä. Jos käytettäisiin julkisen avaimen salausta, Alice salaisi aluksi omien osajakojensa erotuksen omalla julkisella avaimellaan ennen lähettämistä Bobille. Näin ollen Bob ei saisi tietää Alicen vertailulukua, mutta kuten esimerkit osoittavat vertaileminen onnistuu myös salattujen vertailulukujen kesken, joten yksityisyysvaatimus on saavutettu.



Kuva 6: Esimerkki, kun a < b



Kuva 7: Esimerkki, kun a > b

5 Permutaatioprotokolla

Edellisessä luvussa 4 on kuvattu, miten vertailun osapuolet voivat vertailla toistensa lukuja saamatta tietää toisen osapuolen lukua. Toisaalta vertailun osapuolet saavat silti tietää vertailun tuloksen, toisin sanoen kumman luku oli isompi. Tässä luvussa esitellään permutaatioprotokolla, joka suoritetaan ennen varsinaista vertailua. Protokolla luo osapuolista satunnaisen permutaation, ja aiheuttaa sen, että vertailun osapuolet eivät tiedä kenen kanssa lukuja verrataan ja eivät näin ollen voi oppia tai jäljittää keskinäistä arvojärjestystään. Samanlaista permutaatiomallia on käytetty muissakin SMC-

sovelluksissa, mutta ne vaativat O(n) kierrosta verrattuna tässä luvussa esitetyn protokollan O(log n) kierrokseen.

Permutaatioprotokolla käyttää apuna sekoittavia verkkoja, joiden avulla saavutetaan seuraavat ominaisuudet.

1) Bob ei tiedä, että viesti tuli Alicelta
2) Bobilla on anonyymi vastauskanava $c_{[BA]}$, jonka kautta hän voi vastata Alicelle

Määritelmiä:

A $\rightarrow_{mix}$ B tarkoittaa Alicen Bobille lähettämää viestiä

A $\leftarrow_{mix}$ B tarkoittaa Bobin anonyymiä vastausta

Oletetaan, että on käytettävissä synkroninen, autentikoitu yleislähetyskanava $C_{broadcast}$ ja että ensimmäisellä verkkokerroksella vertailu tapahtuu osapuolten p_{2i-1} ja p_{2i} välillä kaikille i = 1,..., n/2. Merkitään myös, että P_{odd} tarkoittaa parittomien osapuolien ja P_{even} parillisten osapuolten joukkoja.

Lasketaan protokollan $P_{permutation}$ avulla satunnainen permutaatio II, jossa jokainen osapuoli p_i tietää vain oman paikkansa permutoidussa vektorissa.

1. Jokainen osapuoli p_i, jolla ei ole lähettävää vertailukumppania (*incoming partner*) ts. joka ei vielä tiedä kenen vertailuluvun vastaanottaa, ilmoittaa itsestään muille yleislähetyksellä $C_{broadcast}$ kanavan kautta. Olkoon S_{free} näiden osapuolten joukko, joka alussa sisältää kaikki osapuolet. Jos joukko on tyhjä, protokollan suoritus päättyy.

$$p_i \rightarrow c_{broadcast} : p_i \qquad \text{if } \nexists \Pi(x) = i$$

2. Jokainen osapuoli p_{in}, jolla ei ole vastaanottavaa vertailukumppania (*outgoing partner*) ts. joka ei vielä tiedä kenelle lähettää oman vertailulukunsa, valitsee satunnaisesti kumppanin p_{out} joukosta S_{free} ja lähettää tälle viestin m.

$$p_{in} \rightarrow_{mix} p_{out} : m \qquad \text{if } \nexists \Pi\ (in)$$

3. Olkoon s_i saapuneiden viestien joukko osapuolella p_i. Jokainen osapuoli p_i valitsee nyt satunnaisesti yhden osapuolen p_j joukosta s_i (jos se ei ole tyhjä) . p_i lähettää p_j:lle *accept*-viestin anonyymin vastauskanavan kautta. Kaikille muille joukon s_i osapuolille p_k lähetetään viesti *reject*.

$p_j \leftarrow_{mix} p_i$: accept $\quad p_j \in s_i$
$p_k \leftarrow_{mix} p_i$: reject $\quad \forall p_k \in s_i \wedge k \neq j$

4. Saadessaan *accept*-viestin, p_{in} lisää permutaatioon II(in) = out.

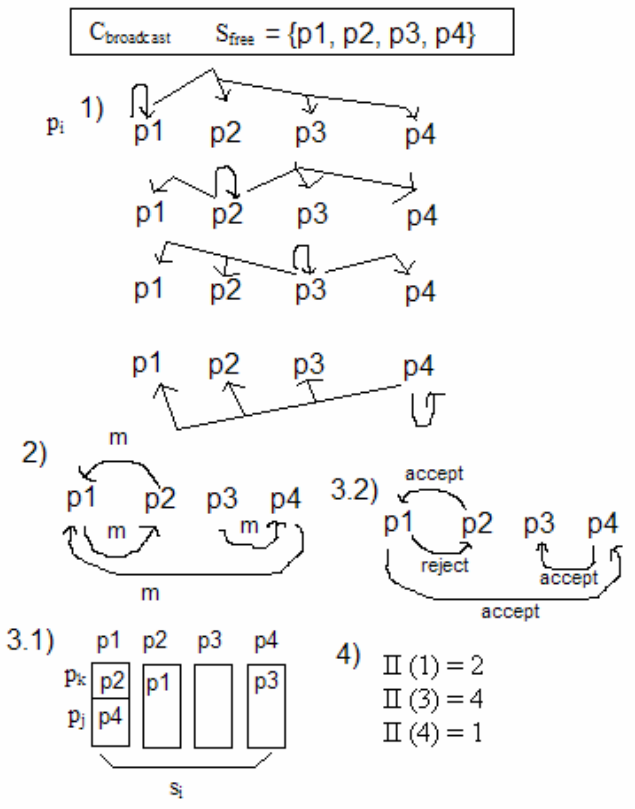
accept ➔ II(in) = out

5. Jatketaan protokollaa askeleesta 1, kunnes jokaisella osapuolella on kumppani, jonka kanssa vertailla lukuja.

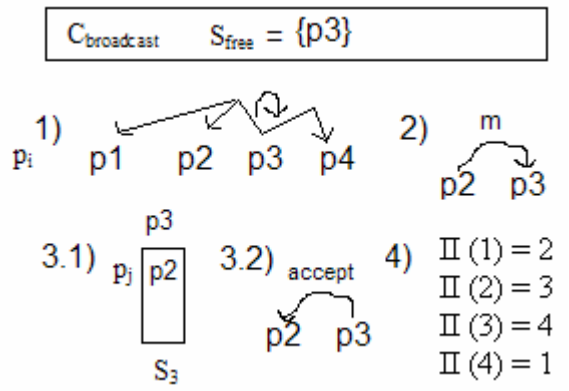
Protokollan suoritus vaatii pahimmassa tapauksessa O(n) kierrosta, jos jokainen osapuoli valitsee joka kerta saman toisen osapuolen p_{out}. Jos jokainen osapuoli valitsee eri kumppanin protokollan suoritus päättyy tällä kierroksella. Keskimäärin kierroksia joudutaan käymään läpi O(log n).

5.1 Permutaatioprotokollan esimerkki

Seuraavan sivun kuvissa 8 ja 9 on esitetty sovellettu esimerkki permutaatioprotokollan suorittamisesta, kun osapuolia on neljä. Aluksi mikään osapuoli p_i ei vielä tiedä kenen vertailuluvun vastaanottaa. Kuvan 8 kohdassa 1 jokainen p_i lähettää itsestään tiedon muille yleislähetyksenä $C_{broadcast}$ kanavan kautta. Tämän jälkeen kohdassa 2 jokainen osapuoli, joka ei vielä tiedä kenelle lähettää lukunsa valitsee satunnaisesti toisen osapuolen joukosta S_{free} ja lähettää tälle viestin m. On huomattava, että jokainen osapuoli voi valita myös itsensä kumppanikseen, vaikka se ei esimerkkikuvasta käy ilmi. Kohdassa 3.1 jokainen valitsee saapuneiden viestien joukostaan s_i jonkin osapuolen p_j (paitsi p_3, jonka joukko on tyhjä) ja lähettää tälle kohdassa 3.2 viestin *accept.* Kaikille muille osapuolille p_k lähetetään viesti *reject*. Kun jokin osapuoli on saanut accept-viestin, se lisää p_{out} kumppanikseen sen osapuolen jolta accept-viestin sai. Koska p_1 hylkäsi p_2:n tarjouksen, p_2 ei vielä löytänyt osapuolta, jolle lähettää lukunsa ja toisaalta kukaan ei pyytänyt lupaa saada lähettää p_3:lle. Tarvitaan vielä protokollan 2. kierros, jotta kaikki löytävät itselleen parin.



Kuva 8: Permutaatioprotokollan 1. kierros



Kuva 9: Permutaatioprotokollan 2. kierros

Protokollan 2. kierros on esitetty kuvassa 9. Ensin jokainen osapuoli, joka ei vielä saa lukua keneltäkään (p_3) ilmoittaa itsestään yleislähetyksellä. Kohdassa 2 jokainen osapuoli, jolla ei ole vielä ketään jolle lähettää lukunsa (p_2) valitsee toisen osapuolen joukosta S_{free} ja lähettää tälle viestin m. Seuraavaksi p_3 lähettää p_2:lle takaisin *accept*-viestin. Protokollan suoritus päättyy seuraavan kierroksen alussa, kun kaikilla on joku osapuoli p_{in}, jolta saa vertailuluvun.

Viestien lähetyksessä osapuolelta toiselle käytetään anonyymiä yleislähetyskanavaa ja sekoittavia verkkoja, joissa välityspalvelimet piilottavat viestin lähteen ja kohteen toisiltaan. Toisaalta jokaisella osapuolella on anonyymi vastauskanava, jolla ne voivat vastata saamiinsa viesteihin. Päädytään siihen, että osapuolet eivät tiedä missään vaiheensa, kuin oman paikkansa permutaatiossa.

6 Sekoituspermutaatioprotokolla

Sekoituspermutaatio tai sekoitus (*derangement*) tarkoittaa permutaatiota, jossa mikään elementti ei saa säilyä omalla paikallaan. Esitellään $P_{derangement}$, joka muuttaa luvussa 5 esitetyn $P_{permutation}$ protokollan askelta 2 hieman.

2. Jokainen osapuoli p_{in}, jolla ei ole vastaanottavaa vertailukumppania (*outgoing partner*) ts. joka ei vielä tiedä kenelle lähettää oman vertailulukunsa, valitsee satunnaisesti kumppanin p_{out} joukosta S_{free}, joka ei kuitenkaan ole se itse ja lähettää tälle viestin m. Jos osapuoli on ainoa, jolla ei vielä ole lähettävää vertailukumppania (*incoming partner*), se lähettää fail- viestin muille yleislähetyksenä ja protokolla alkaa alusta tyhjällä permutaatiolla II.

$p_{in} \rightarrow_{mix} p_{out} : m$ if $\nexists \Pi(in) \wedge out \neq in$

$p_{in} \rightarrow c_{broadcast} : fail$ if $s_{free} = \{in\}$

Teoreettisesti on mahdollista, että protokollan suoritus ei lopu koskaan, sillä ei voida olla varmoja, että sekoitus löytyy. Todennäköisyys *p* sille, että satunnainen permutaatio on myös sekoitus, on 1/e. Toisaalta edellä esitetty muutos protokollaan aiheuttaa sen, että keskimääräinen vaativuus uudelleensekoitukselle on luokkaa O(1) ja sekoituksen löytämiselle O(log n) kierroksella.

7 Lopullinen suodatusprotokolla

Tässä luvussa esitetään lopullinen suodatusprotokolla $P_{final,}$ jota voidaan käyttää alustamaan vertailun osapuolten luvut ja järjestys ennen varsinaisen suorituskykyvertailun aloittamista. Aluksi osapuolet valitsevat satunnaisen sekoituspermutaation ja lähettävät vertailulukunsa tämän järjestyksen läpi kaksi kertaa.

Ensimmäisellä kierroksella lukujen lähettäjä on jo piilotettu ja toinen kierros piilottaa kohteen lähettäjältä.

Lopullisessa suodatusprotokollassa vertailuluvut pitää jakaa kahtia vertailun osapuolten kesken, kuten luvun 2 vertailuporttien tapauksessa. Yksi vertailuaskel tapahtuu osapuolten p_{2i-1} ja p_{2i} välillä kaikille i = 1,..., n/2. Permutaatiota käytetään vain näiden parien välillä. Luvut salataan julkisen avaimen salauksella sekä piilotekijöillä r_a ja r_b. Alla suodatusprotokollan täsmällinen määritelmä.

1. Jokainen ”pariton” osapuoli p_{2i-1} valitsee satunnaisesti toiselta p_{odd}:iin kuuluvalta osapuolelta p_j julkisen avaimen ja lähettää:

   $p_{2i-1} \rightarrow p_{2i} : j, r_{2i-1}, E_{pj}(x_{2-1} - r_{2i-1}) \qquad j \neq 2i - 1$

2. Jokainen ”parillinen” osapuoli p_{2i} suorittaa $P_{derangement}$ protokollan. Ne saavat parillisten osapuolten sekoituksen II.

3. Jokainen ”parillinen” osapuoli p2i lähettää sekoittavan verkon yli kumppanilleen $P_{intermediate} = PII(2_i)$ viestin:

   $p_{2i} \rightarrow_{mix} p_{intermediate} : j, r_{2i-1}, E_{pj}(x_{2i-1} - r_{2i-1}), r_{2i}, E_{pj}(x_{2i} - r_{2i})$

4. Jokainen välittäjäosapuoli $p_{intermediate}$ (eli jokainen ”parillinen” osapuoli) lähettää pj:lle:

   $p_{intermediate} \rightarrow p_j : E_{pj}(x_{2i-1} - r_{2i-1}), E_{pj}(x_{2i} - r_{2i})$

5. Jokainen osapuoli p_j, johon on otettu yhteyttä, valitsee satunnaisesti julkisen avaimen p_{odd}:in osapuolelta p_k jokaiselle viestille ja lähettää takaisin välittäjälle ($p_{intermediate}$):

   $p_j \rightarrow p_{intermediate} : k, E_{pk}(x_{2i-1} - r_{2i-1}), E_{pk}(x_{2i} - r_{2i})$

6. Jokainen välittäjäosapuoli $p_{intermediate}$ valitsee satunnaiset arvot r_a ja r_b ja laskee:

   $E_{pk}(a_A) = E_{pk}(x_{2i-1} - r_{2i-1}) \times r_a = E_{pk}(x_{2i-1} - r_{2i-1} + r_a)$

   $a_B = r_{2i-1} - r_a$

   $E_{pk}(b_A) = E_{pk}(x_{2i} - r_{2i}) \times r_b = E_{pk}(x_{2i} - r_{2i} + r_b)$

   $b_B = r_{2i} - r_b$

7. Jokainen välittäjäosapuoli $p_{intermediate}$ lähettää sekoittavan verkon yli kumppanilleen $p_{last} = p_{II}(p_{intermediate})$:

   $p_{intermediate} \rightarrow_{mix} p_{last}$: k, a_B, $E_{pk}(a_A)$, b_B, $E_{pk}(b_A)$

   last = II(intermediate)

8. Jokainen viimeinen osapuoli p_{last} (jälleen jokainen ”parillinen” osapuoli) lähettää järjestävän verkon yli kumppanilleen p_{last-1}:

   $p_{last} \rightarrow p_{last-1}$: k, $E_{pk}(a_A)$, $E_{pk}(b_A)$

9. Jokainen osapuoli p_{last-1} valitsee satunnaiset luvut r'_a ja r'_b, joilla piilottaa omat osajakonsa ja lähettää tulokset osapuolelle p_k (jos tarpeellista):

   $p_{last-1} \rightarrow p_k$: $E_{pk}(a_A)$ x $E_{pk}(r'_a)$, $E_{pk}(b_A)$ x $E_{pk}(r'_b)$

10. Jokainen osapuoli p_k, johon otettiin yhteyttä palauttaa puretun sisällön jokaiselle viestille ja p_{last-1} piilotus poistuu:

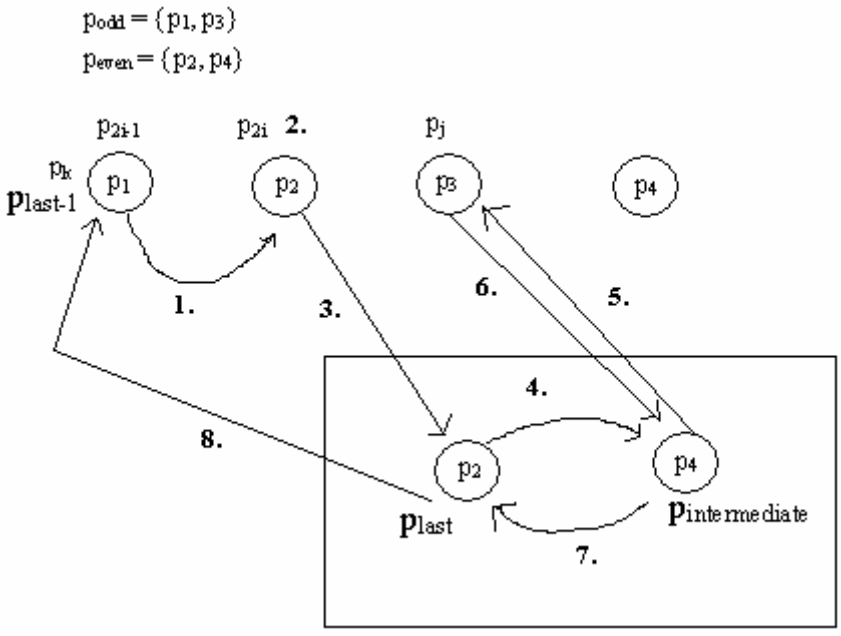
    $p_k \rightarrow p_{last-1}$: $a_A + r'_a$, $b_A + r'_b$

Nyt kaikki osapuolet (parit) on järjestetty satunnaisen permutaation mukaan ja vertailuluvut on jaettu niin, että luvussa 3 esitetyn järjestävän verkon suoritus voi alkaa. Kaikkien viestien sisältö pysyy jäljittämättömissä, sillä välittäjäosapuoli vaihtaa julkisia avaimia, niin että mikään salausavain ei anna vinkkiä siitä mistä viesti tuli ja se myös uudelleensatunnaistaa viestien sisällön. Solmut, jotka purkavat salauksen eivät pysty päättelemään mitään vertailuluvuista, sillä satunnaiset piilotusluvut sokaisevat ne.

7.1 Lopullisen suodatusprotokollan esimerkki

Seuraavan sivun kuvassa 10 on yksinkertaistettu esimerkki lopullisen suodatusprotokollan toiminnasta. Kuvassa on neljä osapuolta ja se esittää yhden vertailuaskelen esimerkin viestienvälityksestä osapuolten välillä, kun tarkoituksena on alustaa osapuolet p_1 ja p_2 vertailua varten. Aluksi p_1 valitsee julkisen avaimen joltain toiselta P_{odd}:iin kuuluvalta jäseneltä p_j. (kuvan tapauksessa siis p_3:lta), salaa oman vertailulukunsa sillä ja lähettää salatun luvun vertailuparilleen p_2:lle. P_2 käynnistää sekoituspermutaatioprotokollan parillisten osapuolten kesken ja lähettää tämän jälkeen p_1:ltä saamansa luvun ja oman lukunsa salattuna välittäjänä toimivalle osapuolelle p_4.

Välittäjänä toimiva osapuoli p_4 lähettää p_2:lta saamansa luvut p_3:lle (joka on se p_j, jonka julkisella avaimella p_1 alun perin lukunsa salasi). P_3 pystyy nyt purkamaan saadut luvut omalla salaisella avaimellaan ja vaihtaa julkisen avaimen uuteen. Se valitsee julkisen avaimen joltain toiselta p_{odd}:iin kuuluvalta jäseneltä p_k, (tässä tapauksessa p_1) salaa alkuperäiset luvut uudelleen tällä julkisella avaimella ja lähettää tiedot takaisin välittäjäosapuolelle p_4. P_4 puolittaa luvut osapuolten kesken osiin niin, että alkuperäisen vertailun osapuolilla on molemmilla vain osa omasta luvustaan ja osa toisen luvusta (vrt. vertailuporttiprotokolla) ja lähettää kumppanilleen p_{last}:lle (p_2) tämän osapuolen osat sekä p_{last}:n vertailukumppanin p_{last-1}:n (p_1) osat salattuna. P_2 saa näin tarvitsemansa osat lukujen lopullista vertailua varten. Se lähettää edelleen välittäjältä saamansa p_1:n osat p_1:lle ja lopputuloksena on se, että p_1:llä on luvut a_A ja b_A ja p_2:lla on luvut a_B ja b_B, kuten luvun 3 vertailuporttiesimerkin alkutilanteessa.



Kuva 10: Esimerkki lopullisesta suodatusprotokollasta

8 Yhteenveto

Käyttämällä luvussa 7 mainittua protokollaa, saadaan järjestettyä *n* osapuolen vertailuluvut niin, että varsinainen suorituskykyvertailu voi alkaa. Ensin osapuolten luvut permutoidaan niin, että mikään toinen osapuoli ei tiedä mistä sen saama vertailuluku tuli. Sitten luvut jaetaan salaisesti osapuolten kesken suodatusprotokollan avulla niin, että luvussa 3 esitetyn vertailuportteja käyttävän järjestävän verkon suoritus

voi käynnistyä. Tämän jälkeen voidaan suorittaa mikä tahansa suorituskykyalgoritmi, joka rajoittaa tulokset k- parhaisiin arvoihin.

Parhaiden k-arvojen keskiarvo voidaan laskea seuraavasti. Ensin suoritetaan edellisessä kappaleessa esitetyt toiminnot, niin että luvut ovat järjestyksessä. Sitten ensimmäinen osapuoli valitsee satunnaisen piilotustekijän r ja lähettää summan sum = r + a_A + b_A toiselle osapuolelle. Jokainen seuraava osapuoli lisää summaan omat vertailuarvonsa sum = sum + a_B + b_B. Jos ollaan edelleen k-parhaiden arvojen sisällä, toinen osapuoli jatkaa lähetystä eteenpäin. Viimeinen osapuoli lähettää tuloksena saadun summan ensimmäiselle osapuolelle, joka lähettää puolestaan sum – r yleislähetyksellä kaikille muille osapuolille. Keskiarvoksi tulee sum/k. Jokainen osapuoli voi myös laskea keskihajonnan vähentämällä keskiarvon saamastaan summasta ja korottamalla sen toiseen potenssiin.

9 Lähde

[KT06] Kerschbaum F., Terzidis O., *Filtering for Private Collaborative Benchmarking*, http://www.fkerschbaum.org/etrics06.pdf, ETRICS 2006