TAMPEREEN YLIOPISTO
Pro gradu -tutkielma

Tuukka Hatakka

Multiskalaarikertolasku elliptisillä käyrillä

Informaatiotieteiden yksikkö
Matematiikka
Elokuu 2011

Tampereen yliopisto

Informaatiotieteiden yksikkö

Hatakka, Tuukka: Multiskalaarikertolasku
elliptisillä käyrillä

Pro gradu -tutkielma, 39 s.

Matematiikka

Elokuu 2011

Tiivistelmä

Tässä tutkielmassa käsitellään multiskalaarikertolaskua elliptisillä käyrillä. Multiskalaarikertolaskun laskeminen tehokkaasti on kiinnostavaa, koska se dominoi Elliptisten käyrien allekirjoitusalgoritmin (ECDSA) suoritusaikaa. Työssä keskitytään esittelemään laskemiseen käytettäviä algoritmeja ja niiden matemaattista taustaa. Tavoitteena on tutkia niiden sopivuutta eri avainten koilla. Tutkielman alussa esitetään, miten elliptisistä käyristä voidaan muodostaa ryhmiä. Tämä ominaisuus on perusta elliptisten käyrien kryptografialle. Perusteiden jälkeen tutkielman pääaihetta käsitellään käymällä läpi erilaisia menetelmiä multiskalaarikertolaskun laskemiseen. Ohessa tutustutaan numeroesitysten teoriaan. Tutkielma loppuosassa keskitytään vertailemaan algoritmeja aikaisempiin analyyseihin ja käytännön toteutukseen perustuen.

Sisältö

1 Johdanto

Tämä pro gradu -tutkielma käsittelee multiskalaarikertolaskun laskemista elliptisillä käyrillä. Multiskalaarikertolaskun laskeminen tehokkaasti on tärkeää, koska sitä tarvitaan elliptisten käyrien kryptografian allekirjotusten varmistamisessa. Elliptisten käyrien kryptografia mahdollistaa pienemmät avainten koot, kuin perinteiset menetelmät kuten RSA. Avaintein koilla on erityisesti merkitystä sovelluksissa, joissa muistia on käytössä hyvin rajallisesti. Tutkielman lukijalta oletetaan perustietoja luku- ja ja kuntateorioista. Näiden lisäksi kryptografian perusteet ovat hyödyksi.

Luvussa 2 käydään läpi elliptisten käyrien perusteet, ja näytetään miten niiden pohjalta voidaan muodostaa äärellisiä kuntia. Lähteenä on käytetty tekijöiden Hankerson, Menezes ja Vanstone kirjaa Guide to Elliptic Curve Cryptography.

Luvussa 3 esitetään kahdenna ja lisää -algoritmi skalaarikertolaskulla ja osoitetaan algoritmin tuloksen oikeellisuus. Seuraavan luvun multiskalaarikertolaskun algoritmit toimivat vastaavaan ideaan perustuen. Kappale perustuu edellisen luvun tavoin kirjaan Guide to Elliptic Curve Cryptography.

Luku 4 on tutkielman pääluku. Luvussa käydään läpi multiskalaarikertolaskun laskemiseen käytettäviä algoritmeja. Algoritmit on poimittu Möllerin artikkelista Algorithms for multi-exponentiations. Algoritmien ohessa esitettävät numeroesityksien ja algoritmien analyysit perustuvat artikkeleihin Solinasin Low-Weight Binary Representations for Pairs of Intergers, Avanzin On the Complexity of Certain Multi-exponentiation techniques in cryptography ja J. Muirin Efficient Integer Representations for Cryptographic Operations.

Luvussa 5 vertaillaan luvussa 4 esitettyjä algoritmeja. Vertailu perustuu edellisen luvun analyyseihin ja algoritmien toteutukseen Matlab 7.10 ohjelmistolla.

2 Elliptisten käyrien perusteet

Tässä kappaleessa käydään läpi tutkielmassa tarvittavat perusteet elliptisistä käyristä. Kappale mukailee kirjan Guide to Elliptic Curve Cryptography kappaletta 3.1 (ks.[1, s. 76-85]).

Määritelmä 2.1. Elliptinen käyrä E *yli kunnan* K on muotoa

$$E := y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \tag{2.1}$$

oleva yhtälö, missä $a_1, a_2, a_3, a_4, a_6 \in K$ ja elliptisen käyrän E *diskriminantti*, merkitään Δ, eroaa nollasta (ks. Määritelmä 2.2). Elliptisen käyrän yhtälöä kutsutaan myös *Weierstrassin yhtälöksi.*

Jos L on K:n mikä tahansa laajennus, niin *L-rationaalisten pisteiden* joukko käyrällä E on

$$E(L) = \{(x, y) \in LxL : y^2 + a_1xy + a_3y - x_3 - a_2x_2 - a_4x - a_6 = 0\} \cup \{\infty\},$$

missä ∞ on *piste äärettömyydessä.*

Määritelmä 2.2. Elliptisen käyrän E diskriminantti määritellään seuraavasti:

$$\left.\begin{aligned} \Delta &= -d_2^2d_8 - 8d_d^3 - 27d_6^2 + 9d_2d_4d_6 \\ d_2 &= a_1^2 + 4a_2 \\ d_4 &= 2a_4^2 + a_1a_3 \\ d_6 &= a_3^2 + 4a_6 \\ d_8 &= a_1^2a_6 + 4a_2a_6 - a_1a_3a_4 + a_2a_3^2 - a_4^2. \end{aligned}\right\}$$

Huomautus 1. Käytetään käyrästä E *yli kunnan* K merkintää E/K.

2.1 Yksinkertaistettu Weierstrassin yhtälö

Jokainen elliptinen käyrä voidaan sopivalla muuttujien vaihdolla saattaa yksinkertaisempaan muotoon.

Määritelmä 2.3. Kunnan K *karakteristika* on pienin sellainen positiivinen kokonaisluku n, jolle pätee $n \cdot 1 = 0$, missä 1 on kunnan K ykkösalkio ja 0 kunnan K nolla-alkio. Jos tällaista lukua ei ole olemassa, karakteristika on nolla. Käytetään karakteristikasta merkintää $char(K)$.

Määritelmä 2.4. Olkoon elliptisten käyrien E_1/K ja E_2/K Weierstrassin yhtälöt

$$\begin{aligned} E_1 &: = y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, \\ E_2 &: = y^2 + b_1xy + b_3y = x^3 + b_2x^2 + b_4x + b_6. \end{aligned}$$

Käyrät E_1 ja E_2 ovat keskenään *isomorfisia yli K:n*, jos on olemassa $u, k, s, t \in K$ s.e muuttujien vaihto

$$(x, y) \to (u^2x + r, u^3y + u^2sx + t)$$

muuttaa yhtälön E_1 yhtälöksi E_2. Tätä muutosta kutsutaan *päteväksi muuttujien vaihdoksi.*

Nyt Weierstrassin yhtälöä

$$E := y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

yli K:n, voidaan pätevällä muuttujien vaihdolla yksinkertaistaa huomattavasti. Käsitellään erikseen tapaukset missä $char(K) \neq 2, 3$; $char(K) = 2$ ja $char(K) = 3$

1. Jos $char(K) \neq 2, 3$, niin muuttjien vaihdolla

   $$(x, y) \to \left(\frac{x - 3a_1^2 - 12a_2}{36}, \frac{y - 3a_1x}{216} - \frac{a_1^3 + 4a_1a_2 - 12a_3}{24}\right)$$

   E muuttuu käyräksi

   $$y^2 = x^3 + ax + b \tag{2.2}$$

   missä $a, b \in K$ ja $\Delta = -16(4a_3 + 27b_2)$.

2. Jos $char(K) = 2$, jaetaan käsittely kahteen tapaukseen $a_1 = 0$ ja $a_1 \neq 0$.

   (a) Jos $a_1 \neq 0$, niin muuttujien vaihdolla

   $$(x, y) \to \left(a_1^2x + \frac{a_3}{a_1}, a_1^3y + \frac{a_1^2a_4 + a_3^2}{a_1^3}\right)$$

   E muuttuu käyräksi

   $$y^2 + xy = x^3 + ax^2 + b, \tag{2.3}$$

   missä $a, b \in K$ ja $\Delta = b$. Tällaistä käyrää kutsutaan *ei-supersingulaariseksi.*

   (b) Jos $a_1 = 0$, niin muuttujien vaihdolla

   $$(x, y) \to (x + a_2, y)$$

   E muuttuu käyräksi

   $$y^2 + cy = x^3 + ax^2 + b \tag{2.4}$$

   missä $a, b, c \in K$ ja $\Delta = c^4$. Tällaista käyrää kutsutaan *supersingulaariseksi.*

3. Jos $char(K) = 3$, jaetaan käsittely tapauksiin $a_1^2 = -a_2$ ja $a_1^2 \neq -a_2$.

   (a) Jos $a_1^2 \neq -a_2$, niin muttujien vaihdolla

   $$(x, y) \to \left(x + \frac{d_4}{d_2}, y + a_1 x + a_1 \frac{d_4}{d_2} + a_3 \right),$$

   missä $d_2 = a_1^2 + a_2$ ja $d_4 = a_4 - a_1 a_3$, E muuttuu käyräksi

   $$y^2 = x^3 + ax^2 + b, \tag{2.5}$$

   missä $a, b \in K$ ja $\Delta = -a^3 b$. Tällaista käyrää kutsutaan *ei-supersingulaariseksi.*

   (b) Jos $a_1^2 = -a_2$, niin muuttujien vaihdolla

   $$(x, y) \to (x, y + a_1 x + a_3)$$

   E muuttuu käyräksi

   $$y^2 = x^3 + ax + b, \tag{2.6}$$

   missä $a, b \in K$ ja $\Delta = -a^3$. Tällaista käyrää kutsutaan *supersingulaariseksi.*

2.2 Ryhmälait

Olkoon E elliptinen käyrä yli kunnan K. Nyt voidaan määritellä *jänne-tangentti -sääntö* kahden joukkoon $E(K)$ kuuluvan pisteen yhteenlaskulle siten, että tuloksena on kolmas piste joukossa $E(K)$. Näin määritelty yhteenlasku, joukko $E(K)$ ja piste ∞ muodostavat Abelin ryhmän, missä ∞ toimii nolla-alkiona. Jänne-tangentti-sääntöä voidaan kuvailla seuraavasti. Olkoon P ja Q erilliset pisteet käyrällä E. Pisteiden P ja Q summa R määritellään seuraavasti: Pisteiden P ja Q kautta kulkeva suora leikkaa käyrän E kolmannessa pisteessä, R on tämän leikkauspisteen peilaus x-akselin suhteen. Kun pisteiden P ja Q y-koordinaatit ovat samat sovitaan, että suora leikkaa käyrän pisteessä ∞.

Vastaavasti, jos piste P summataan itsensä kanssa, otetaan pisteen P tangentti. Nyt tangentti leikkaa käyrän E toisessakin pisteessä, ja summan tulos on leikkauspisteen peilaus x-akselin suhteen. Jos tangentti on y-akselin suuntainen, niin summan tulos on ∞.

Tässä pykälässä esitetään tarkat yhtälöt pisteiden yhteenlaskulle.

Ryhmälait käyrälle $\mathbf{E/K : y^2 = x^3 + ax + b, char(K) \neq 2, 3}$

1. *Neutraalialkio:* $P + \infty = P$ kaikilla $P \in E(K)$.

2. *Vasta-alkio:* Jos $P = (x, y) \in E(K)$, niin selvästi $(x, y) + (x, -y) = \infty$ ja $(x, -y) \in E(K)$. Pisteestä $(x, -y)$ käytetään merkintää $-P$.

3. *Yhteenlasku:* Olkoon $P = (x_1, y_1)$ ja $Q = (x_2, y_2)$.
   Jos $P \neq Q$, niin $P + Q = (x_3, y_3)$, missä

$$x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2 \quad \text{ja} \quad y_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)(x_1 - x_3) - y_1.$$

   Olkoon $P = Q$ ja $P \neq -P$. Tällöin $P + Q = 2P = (x_3, y_3)$, missä

$$x_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)^2 - 2x_1 \quad \text{ja} \quad y_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)(x_1 - x_3) - y_1.$$

Huomautus 2. Jatkossa kahden elliptisen käyrän pisteen yhteenlaskua kutsutaan lisäykseksi (merkitään L), jos pisteet ovat erillisiä ja kahdennukseksi (merkitään K), jos pisteet ovat identtisiä.

3 Skalaarikertolasku

Tässä kappaleessa esitetään kahdenna ja lisää -menetelmä pisteen $kP = \underbrace{P + P + \cdots + P}_{k \text{ kpl}}$ laskemiseen, missä k on kokonaisluku ja P elliptisen käyrän E yli kunnan $\mathbb{F}_q$ piste. Pisteen kP määritystä kutsutaan *pistekertolaskuksi* tai *skalaarikertolaskuksi*. Ennen algoritmia käydään läpi perusteet numeroesityksistä, koska niillä on keskeinen rooli menetelmien tehokkuudessa.

Määritelmä 3.1. Luvun k numeroesitys kannan b suhteen on

$$k = \sum_{i=0}^{t-1} k_t b^t \text{ ,missä } 0 \leq k_i < b.$$

Jatkossa käytetään merkintää $k = (k_{t-1} \ldots k_1 k_0)_b$.

Esimerkki 3.1. Luvun 53 numeroesitys (binääriesitys) kannan 2 suhteen on $1 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 2^5 + 2^4 + 2^2 + 1 = (110101)_2$.

Määritelmä 3.2. Esityksen *Hamming-paino* on siinä olevien nollasta poikkeavien lukujen määrä. Vastaavasti esityksen *Hamming-tiheys* on Hamming-paino jaettuna esityksen pituudella.

Esimerkki 3.2. Luvun 53 binääriesityksen $(110101)_2$ Hamming-paino on 4, ja sen Hamming-tiheys on $2/3$.

Esimerkki 3.3. Satunnaisen kokonaisluvun k binääriesityksen Hamming-tiheyden odotusarvo on $1/2$.

3.1 Kahdenna ja lisää -menetelmä

Algoritmi 3.1 (vrt. [1, Algorithm 3.27 s. 97]) laskee skalaarikertolaskun kP tutkien luvun k binääriesitystä vasemmalta oikealle. Algoritmista on olemassa myös versio oikealta vasemmalle (ks. [1, Algorithm 3.26 s. 96]).

Algoritmi 3.1 Vasemmalta oikealle kahdenna ja lisää -menetelmä

Input: $k = (k_{t-1}, ..., k_2, k_1, k_0)_2, P \in E(\mathbb{F}_q)$
Output: kP

```
1: Q ← inf
2: for i = t − 1 downto 0 do
3:     Q ← 2Q
4:     if k_i = 1 then
5:         Q ← Q + P
6:     end if
7: end for
8: return Q
```

Esimerkin 3.3 perusteella luvun k binääriesityksen Hamming-painon odotusarvo on $t/2$, joten algoritmi 3.1 vaatii $t/2$ pisteen lisäystä ja t pisteen kahdennusta, siis

$$\frac{t}{2}L \text{ ja } tK. \tag{3.1}$$

Osoitetaan, että algoritmin 3.1 tulos todellakin on kP.

Lause 3.1. *Algoritmin 3.1 tulos on kP.*

Todistus. Olkoon $(k_{t-1} \dots k_0)_2$ luvun k binääriesitys. Arvo Q muodostuu seuraavasti arvon i pienentyessä

i	Q
$t-1$	$k_{t-1}P$
$t-2$	$2k_{t-1}P + k_{t-2}P$
$t-3$	$2^2k_{t-1}P + 2k_{t-2}P + k_{t-3}P$
$\vdots$	$\vdots$
0	$2^{t-1}k_{t-1}P + 2^{t-2}k_{t-2}P + \cdots + k_0P.$

Nyt

$$\begin{aligned}
&2^{t-1}k_{t-1}P + 2^{t-2}k_{t-2}P + \cdots + k_0P \\
&= (2^{t-1}k_{t-1} + 2^{t-2}k_{t-2} + \cdots + k_0)P \\
&= ((k_{t-1} \dots k_0)_2)P \text{ (määritelmä 3.1)} \\
&= kP.
\end{aligned}$$

□

4 Multiskalaarikertolasku

Multiskalaarikertolaskulla tarkoitetaan lausekkeita $\sum_{i=0}^{n} k_i P_i$, missä k_i on kokonaisluku ja P_i on elliptisen käyrän E yli kunnan $\mathbb{F}_q$ piste, kaikilla $0 \leq i \leq n$. Tässä tutkielmassa keskitytään tapaukseen $k_0P_0 + k_1P_1$, koska tällaisen yhtälön ratkaiseminen on tarpeellista ECDSA allekirjoitusten tarkistamisessa (ks. [2, 3.1]). Kappaleessa oletetaan, että lukujen k_0 ja k_1 binääriesitykset ovat luvun t mittaisia, ja jos näin ei ole, niin täydennetään lyhyempää esitystä nollilla.

4.1 Lomitus menetelmät

Lomitus menetelmissä molemmille skalaarikertolaskuille esilasketaan oma taulukko, joita hyväksikäyttäen lopputulos kootaan vasemmalta oikealle samanaikaisesti molemmille skalaarikertolaskuille.

Muokkaamalla algoritmia 3.1 saadaan yksinkertainenlomitusmenetelmä.

Algoritmi 4.1 Yksinkertainenlomitusmenetelmä

Input: Kokonaisluvut $k_0 = (k_{0,t-1} \dots k_{0,1}k_{0,0})_2$ $k_1 = (k_{1,t-1} \dots k_{1,1}k_{1,0})_2$ ja $P_0, P_1 \in E(\mathbb{F}_q)$.
Output: $k_0P_0 + k_1P_1$

1: $Q \leftarrow \infty$
2: **for** $j = t - 1$ downto 0 **do**
3: $\quad Q \leftarrow 2Q$
4: $\quad$ **if** $k_{0,j} \neq 0$ **then**
5: $\quad\quad Q \leftarrow Q + P_0$
6: $\quad$ **end if**
7: $\quad$ **if** $k_{1,j} \neq 0$ **then**
8: $\quad\quad Q \leftarrow Q + P_1$
9: $\quad$ **end if**
10: **end for**
11: **return** Q

Nähdään että algoritmissa lisäysten määrä on sama kuin lukujen binääriesitysten yhteenlaskettu Hamming-paino. Siis $2(t)\frac{1}{2} = t$, joten algoritmin 4.1 lisäysten L ja kahdennusten K määrien odotusarvot ovat

$$tL \text{ ja } tK. \tag{4.1}$$

4.1.1 Ikkunalomitusmenetelmä

Ikkunalomitusmenetelmä (vrt. [2, 3.1]) vähentää tarvittavia yhteenlaskuja tutkimalla useampaa numeroesityksen bittiä samanaikaisesti. Käytännössä

tämä tapahtuu määrittämällä halutun kokoinen "ikkuna", joka liikkuu esityksen päällä, ja esilaskemalla ikkunan koosta riippuvat pisteet.

Olkoon skalaarikertolaskun k_0P_0 ikkunan koko w_0 ja skalaarikertolaskun k_1P_1 ikkunan koko w_1. Esilaskentavaiheessa lasketaan pisteet e_0P_0 ja e_1P_1 kaikilla parittomilla luvuilla e_i, joilla pätee $1 \leq e \leq 2^{w_i} - 1$,missä $i \in \{0, 1\}$. Tällaisia epätriviaaleja alkioita on $2^{w_0-1}+2^{w_1-1}-2$ kappaletta, joista jokainen vaatii yhden lisäyksen. Lisäksi tarvitaan yksi kahdennus jokaista ikkunaa $w_i > 1$ kohti.

Algoritmi 4.2 Ikkunalomitusmenetelmä

Input: Ikkunoiden leveydet w_0 ja w_1, kokonaisluvut $k_0 = (k_{0,t-1} \dots k_{0,1}k_{0,0})_2$ $k_1 = (k_{1,t-1} \dots k_{1,1}k_{1,0})_2$ ja $P_0, P_1 \in E(\mathbb{F}_q)$.

Output: $k_0P_0 + k_1P_1$

1: *Esilaskennat* Laske $\forall i \in \{0, 1\}$ kaikki skalaarikertolaskut e_iP_i, missä e_i on pariton ja $1 \leq e_i \leq 2^{w_i} - 1$.
2: $Q \leftarrow \infty$
3: $\forall i \in \{0, 1\} : windowhandle_i \leftarrow 0$
4: **for** $j = t - 1$ downto 0 **do**
5: $\quad Q \leftarrow 2Q$
6: $\quad$ **for** $i = 0$ to 1 **do**
7: $\quad\quad$ **if** $windowhandle_i = \phi$ ja $k_{i,j} = 1$ **then**
8: $\quad\quad\quad J \leftarrow j - w_i + 1$
9: $\quad\quad\quad$ **while** $k_{i,J} = 0$ **do**
10: $\quad\quad\quad\quad J \leftarrow J + 1$
11: $\quad\quad\quad$ **end while**
12: $\quad\quad\quad windowhandle_i \leftarrow J$
13: $\quad\quad\quad e_i \leftarrow (k_{i,j} \dots k_{i,J})_2$
14: $\quad\quad$ **end if**
15: $\quad\quad$ **if** $windowhandle_i = j$ **then**
16: $\quad\quad\quad Q \leftarrow Q + e_iP_i$
17: $\quad\quad\quad windowhandle_i \leftarrow 0$
18: $\quad\quad$ **end if**
19: $\quad$ **end for**
20: **end for**
21: **return** Q

Algoritmin lisäysten määrä on nyt sama kuin muodostettujen ikkunoiden määrä, eli maksimissaan $\frac{t}{w_0} + \frac{t}{w_1}$. Lisäksi algoritmi osaa siirtää ikkunaa nollien ohi. Uuden ikkunan muodostamisen aikana sivuutettujen nollien määrän odotusarvo on $\sum_{n\geq 1} \frac{1}{2^n} = 1$, joten algoritmin 4.2 lisäysten ja kahdennusten määrien odotusarvoiksi saadaan

$$\left(\frac{t}{w_0 + 1} + \frac{t}{w_1 + 1}\right) L \text{ ja } aK, \tag{4.2}$$

missä $t - max(w_i) \leq a \leq t$.

Esimerkki 4.1. Esitetään, miten algoritmi 4.2 laskee multiskalaarikertolaskun $53P_0 + 42P_1$, kun $w_0 = 2$ ja $w_1 = 3$.

Esilasketaan pisteet $3P_0, 3P_1, 5P_1$ ja $7P_1$. Nyt

$$53 = (\underbrace{1 \quad 1}\quad 0\ \underbrace{1}\ 0\ \underbrace{1}\ )_2$$
$$42 = (\underbrace{1 \quad 0 \quad 1}\quad 0\ \underbrace{1}\ 0)_2,$$

missä alasulkeet esittävät ikkunoiden paikat.

Seuraavassa taulukossa kuvataan algoritmin pääsilmukan tapahtumia. Tässä j ja Q ovat algoritmin muuttujien arvoja. Vastaavasti P_0 ja P_1 kertovat mitä esilaskettua pisteen monikertaa tarvitaan.

j	P_0	P_1	Q
5			
4	$3P_0$		$3P_0$
3		$5P_1$	$6P_0 + 5P_1$
2	P_0		$13P_0 + 10P_1$
1		P_1	$26P_0 + 21P_1$
0	P_0		$53P_0 + 42P_1$

Tarvitaan siis kuusi lisäystä ja neljä kahdennusta, joiden lisäksi esilaskentoihin tarvitaan kaksi kahdennusta ja kolme lisäystä.

4.1.2 NAF(A non-adjacent form)

Elliptisillä käyrillä vasta-alkion muodostus on erittäin helppoa. Olkoon $P = (x, y) \in E(\mathbb{F}_q)$. Nyt $-P = (x, -y)$, kun $char(\mathbb{F}_q) > 3$. Tästä seuraa, että vähennyslasku on yhtä vaativaa kuin yhteenlasku. Tätä ominaisuutta voidaan hyödyntää käyttämällä *etumerkillistä numeroesitystä.*

Määritelmä 4.1. Luvun k etumerkillinen numeroesitys kannan b suhteen on

$$k = \sum_{i=0}^{l-1} k_i b^i, \text{ missä } |k_i| < b.$$

Merkitään negatiivisia lukuja etumerkillisissä numeroesityksissä selvyyden vuoksi yläviivalla miinusmerkin sijaan. Esimerkiksi $-3 = \overline{3}$. Kutsutaan tapausta $b = 2$ NAF-esitykseksi ja merkitään tällaista esitystä merkinnällä $k = (k_{l-1} \ldots k_1 k_0)_N$.

Lause 4.1. (NAF-esityksen ominaisuudet) *Olkoon k positiivinen kokonaisluku, jolloin seuraavat kohdat pätevät:*

(i) Luvulla k on yksikäsitteinen NAF-esitys, jota merkitään $NAF(k)$.

(ii) Esityksen $NAF(k)$ pituus l on maksimissaan yhden suurempi kuin luvun k binääriesityksen pituus.

(iii) Esityksen $NAF(k)$ Hamming-tiheyden odotusarvo on $1/3$.

Todistus. Vastaavat tulokset on osoitettu yleisemmin seuraavissa lauseissa:

(i) Ks. lause 4.2.

(ii) Ks. lause 4.4.

(iii) Ks. lause 4.3.

□

Algoritmi 4.3 tuottaa NAF-esityksen positiiviselle kokonaisluvulle.

Algoritmi 4.3 NAF-esityksen määritys positiiviselle kokonaisluvulle.

Input: Positiivinen kokonaisluku k.
Output: $NAF(k)$

```
1: i ← 0
2: while k ≥ 1 do
3:    if k on pariton then
4:       k_i ← 2 − (k mod 4)
5:       k ← k − k_i
6:    else
7:       k_i ← 0
8:    end if
9:    k ← k/2
10:   i ← i + 1
11: end while
12: return (k_{l−1}, k_{l−2}, ..., k_1, k_0)_N
```

Esimerkki 4.2. Luvun 53 NAF-esitys on $(10\bar{1}0101)_N$.

Määritelmä 4.2. Olkoon $w > 2$ kokonaisluku. Nyt positiivisen luvun k *w-leveä NAF*-esitys on

$$k = \sum_{i=0}^{l-1} k_i 2^i,$$

missä $|k_i| < 2^{w-1}$, $k_{l-1} \neq 0$ ja enintään yksi luvun w pituisesta peräkkäisestä luvusta on erisuuri kuin nolla. Esityksen w-leveä NAF-pituus on l. Käytetään tällaisesta esityksestä merkintää $NAF_w(k)$. Kirjoitetaan esitys jatkossa muodossa $k = (k_{l-1} \dots k_1 k_0)_w$.

Huomautus 3. Selvästi kaikilla kokonaisluvuilla k pätee $NAF(k) = NAF_2(k)$.

Esimerkki 4.3. Luvun 53 3-leveä NAF-esitys on $NAF_3(53) = (100\bar{1}00\bar{3})_3$. Vastaavasti $NAF_4(53) = (30005)_3$.

Määritellään etumerkillisen numeroesityksen tarpeisiin muokattu modulo.

Määritelmä 4.3. $r = k \pmod{2^w}$ jos $r \equiv k \pmod{2^w}$ ja $-2^{w-1} \leq r < 2^{w-1}$.

Lause 4.2. *Jokaisella kokonaisluvulla k on täsmälleen yksi $NAF_w(k)$-esitys.*

Todistus. (vrt. [5, s. 17-20]) Todistetaan ensin yksikäsitteisyys

Yksikäsitteisyys: Osoitetaan että luvulla k ei voi olla kahta erillistä $NAF_w(k)$-esitystä .

Tehdään vastaoletus, että on olemassa kaksi erillistä $NAF_w(k)$-esitystä kokonaisluvulle k. Siis

$$k = (a_{l-1} \dots a_1 a_0)_w = (b_{l^*-1} \dots b_1 b_0)_w,$$

missä l ja l^* ovat esitysten pituudet. Nyt jollain n pätee $a_k \neq b_k$. Valitaan pienin tällainen luku n. Tällöin

$$k^* = (a_{l-1} \dots a_{n+1} a_n)_w = (b_{l^*-1} \dots b_{n+1} b_n)_w.$$

Luku k^* on parillinen vain jos $a_n = b_n = 0$, joten aikaisemman oletuksen perusteella k^* on pariton. Siis a_n ja b_n poikkeavat nollasta. Nyt määritelmän 4.2 mukaan

$$\begin{aligned}(a_{l-1} \dots a_{n+w} 00 \dots 0 a_n)_w &= (b_{l^*-1} \dots b_{n+w} 00 \dots 0 b_n)_w \\ a_n &\equiv b_n \pmod{2^w}.\end{aligned}$$

Toisaalta $-(2^{w-1} - 1) \leq a_n, b_n \leq 2^{w-1} - 1$, joten

$$-2(2^{w-1} - 1) \leq a_n - b_n \leq 2(2^{w-1} - 1).$$

Nyt ainoa luvun 2^w monikerta tällä välillä on 0, ja lisäksi $2^w \mid (a_n - b_n)$, joten $a_n - b_n = 0$. Tämä on ristitiita oletuksen $a_n \neq b_n$ kanssa. Todistetaan vielä olemassaolo

Olemassaolo: Osoitetaan, että jokaisella kokonaisluvulla k on $NAF_w(k)$-esitys. Tämä onnistuu helpoimmin esittämällä algoritmi esityksen laskemiseksi.

Algoritmi 4.4 laskee esityksen $NAF_w(k)$ luvulle k. Nyt w-leveä NAF-esitys saadaan jakamalla k toistuvasti kahdella, ja lisäämällä nolla esitykseen, jos k on parillinen. Jos k on pariton, lisätään jakojäännös r väliltä $[-2^{w-1}, 2^{w-1} - 1]$. Tällöin $2^{w-1} \mid (k - r)/2$, joten seuraavat $w - 1$ lisäystä ovat nollia.

Algoritmi 4.4 w-leveän NAF-esityksen määritys kokonaisluvulle

Input: ikkunan leveys w, kokonaisluku k
Output: $NAF_w(k)$

```
1: i ← 0
2: while k ≠ 0 do
3:    if k on pariton then
4:       k_i ← k (mods 2^w)
5:       k ← k − k_i
6:    else
7:       k_i ← 0
8:    end if
9:    k ← k/2
10:   i ← i + 1
11:   {
12: end while}
13: return (k_{i−1}, k_{i−2}, ..., k_1, k_0)_w
```

Kuvaillaan algoritmin toimintaa kahdella funktiolla. Funktio f_w kuvaa algoritmin laskennan aikana tapahtuvaa arvon k muuttumista.

$$f_w(k) := \begin{cases} k/2, & \text{jos } k \text{ parillinen,} \\ (k-r)/2^w, & \text{muualla,} \end{cases} \tag{4.3}$$

missä $r \equiv k \pmod{2^w}$.

Funktio g_w tuottaa osia w-leveästä NAF-esityksestä.

$$g_w(k) := \begin{cases} 0, & \text{jos } k \text{ parillinen,} \\ \underbrace{0\ldots0}_{w-1 \text{ kpl}}\, r, & \text{muualla,} \end{cases} \tag{4.4}$$

missä $r \equiv k \pmod{2^w}$. Huomaa, että tulos on osa numeroesitystä eikä yksittäinen luku. Esimerkiksi, jos $w = 2$, niin $f_2(7) = 2$ ja $g_2(7) = 0\bar{1}$. Nyt algoritmi 4.4 voidaan kirjoittaa seuraavasti.

Algoritmi 4.5 w-leveä NAF-funktioilla

Input: Ikkunan leveys w, kokonaisluku k.
Output: $NAF_w(k)$

```
1: a ← φ
2: while k ≠ 0 do
3:    a ← g_w(k) ‖ a {tässä ‖ merkitsee numeroesitysten yhdistämistä}
4:    k ← f_w(k)
5: end while
6: return (a)_w
```

Esimerkiksi esitys $NAF_3(53)$ muodostuu algoritmilla 4.5 seuraavasti.

k	53	7	1
$g_3(k)$	$00\bar{3}$	$00\bar{1}\|\|00\bar{3}$	$1\|\|00\bar{1}\|\|00\bar{3}$

Osoitetaan, että algoritmi 4.5 pysähtyy kaikilla $k \in \mathbb{Z}$. Pysähtyminen tapahtuu kun muuttuja k saa arvon nolla. Tämä tapahtuu, jos $|f_w(k)| < |k|$, koska tällöin jollain kierroksella $k = 0$.

Tutkitaan tapausta $k \neq 0$.

Oletetaan, että k on parillinen. Nyt $f_w(k) = k/2$, joten $|f_w(k)| < |k|$.

Oletetaan, että k on pariton. Jaetaan käsittely kahteen tapaukseen. Tarkastellaan ensin tapausta $|k| < 2^{w-1}$. Tällöin $r = k$, joten $|f_w(k)| = 0 < |k|$.

Jos taas $|k| \geq 2^{w-1}$, niin

$$|f_w(k)| = \left|\frac{k-r}{2^w}\right| \leq \left|\frac{k}{2^w}\right| + \left|\frac{r}{2^w}\right| < \left|\frac{k}{2^w}\right| + \frac{1}{2}.$$

Nyt

$$\frac{1}{2} = \frac{2^{w-1}}{2^w} \leq \left|\frac{k}{2^w}\right|,$$

ja koska $w \geq 2$, niin

$$|f_w(k)| < 2\left|\frac{k}{2^w}\right| = \left|\frac{k}{2^{w-1}}\right| < |k|\,.$$

Täten algoritmi pysähtyy kaikilla $k \in \mathbb{Z}$.

Osoitetaan vielä, että algoritmin tulokselle $(a)_w$ pätee $(a)_w = k$, ja että $(a)_w$ on w-leveässä NAF-muodossa.

Funktion g_w määritelmän perusteella on selvää, että $(a)_w$ on w-leveässä NAF-muodossa. Tarvitsee siis vain osoittaa, että $(a)_w = k$.

Olkoon i ei-negatiivinen kokonaisluku. Käytetään merkintää f_w^i funktion f_w i-kertaisesta yhdistelmästä. Siis

$$f_w^i = \underbrace{f_w \circ f_w \circ \cdots \circ f_w}_{i \text{ kpl}}.$$

Koska algoritmi 4.5 pysähtyy, on olemassa kokonaisluku $i \geq 0$, jolla $f_w^i(k) = 0$.

Muodostetaan induktio tälläisen luvun i suhteen. Jos $i = 0$, niin $f_w^0(k) = 0$, joten $k = 0$. Algoritmin 4.5 tulos syötteellä 0 on $(\phi) = 0$ kuten pitikin.

Jos $i > 0$, olkoon $k^* = f_w(k)$ ja $(a^*)_2$ algoritmin 4.5 tulos syötteellä k^*. Huomaa, että $f_w^i(k) = f_w^{i-1}(k^*)$. Siis induktio-oletuksen perusteella $(a^*)_2 = k^*$. Nyt algoritmin 4.5 määritelmän perusteella

$$\begin{aligned}
& (a)_2 = (a^* \parallel g_w(k))_2 \\
\Rightarrow \quad & (a)_2 = 2^{\|g_w(k)\|}(a^*)_2 + (g_w(k))_2 \\
\Rightarrow \quad & (a)_2 = 2^{\|g_w(k)\|}k^* + (g_w(k))_2 \\
\Rightarrow \quad & (a)_2 = 2^{\|g_w(k)\|}f_w(k) + (g_w(k))_2,
\end{aligned}$$

missä $\|a\|$ merkitsee esityksen pituutta. Nyt funktio f_w voidaan määritellä funktion g_w suhteen seuraavasti.

$$f_w(k) = \frac{n - (g_w(k))_2}{2^{\|g_w(k)\|}},$$

joten

$$(a)_2 = 2^{\|g_w(k)\|}\frac{k - (g_w(k))_2}{2^{\|g_w(k)\|}} + (g_w(k))_2 = k,$$

josta väitee seuraa induktioperiaatteen perusteella. □

Lause 4.3. *w-leveän NAF-esityksen Hamming-tiheys on* $\frac{1}{w+1}$.

Todistus. (vrt. [6, s. 12-13]). Olkoon k satunnainen kokonaisluku. Määritellään algoritmille 4.5 kaksi tilaa. Algoritmi on tilassa (0), kun $g_w(k) = 0$ ja tilassa $(0 \dots 0r)$, kun $g_w(k) = 0 \dots 0r$.

Oletetaan, että algoritmi on tilassa (0). Siis k on parillinen. Nyt algoritmin tila muuttuu tilaan $(0 \dots 0r)$, jos $k/2$ on pariton. Todennäköisyys sille, että satunnainen parillinen luku jaettuna kahdella on pariton on $1/2$. Siis $Tod((0) \to (0 \dots 0r)) = 1/2$, ja vastaavasti $Tod((0) \to (0)) = 1/2$, missä nuoli merkitee tilan muutosta.

Oletetaan, että algoritmi on tilassa $(0 \dots 0r)$. Siis k on pariton. Nyt algoritmin tila $(0 \dots 0r)$ muuttuu tilaan (0), jos $(k - r)/2^w$ on parillinen. Tämän todennäköisyys on $1/2$, joten $Tod((0 \dots 0r) \to (0)) = 1/2$. Vastaavasti $Tod((0 \dots 0r) \to (0 \dots 0r)) = 1/2$.

Siis molemmat tilat esiintyvät todennäköisyydellä $1/2$.

Nyt odotusarvo funktion g_w tuottaman w-leveän NAF-esityksen osan (0 tai $0 \dots 0r$) pituudelle on $\frac{1}{2}(1 + w)$.

Odotusarvo funktion g_w tuottaman w-leveän NAF-esityksen nollasta poikkeavien lukujen määrälle on $\frac{1}{2}(1 + 0) = \frac{1}{2}$.

Siis w-leveän NAF-esityksen Hamming-tiheyden odotusarvo on $\frac{\frac{1}{2}}{\frac{1}{2}(1+w)} = \frac{1}{w+1}$. □

Esityksen nollien määrän lisäksi multiskalaarikertolaskun algoritmin nopeuteen vaikuttaa esityksen pituus. Seuraavassa lauseessa tutkitaan w-leveän NAF-esityksen maksimipituutta. Esitetään ensin tarvittava apulause.

Lemma 4.1. *Olkoon* $(a_{l-i} \dots a_1 a_0)_2$ *w-leveä NAF-esitys, jonka pituus on* l *ja* $l \geq 1$. *Jos* $k = (a_{l-i} \dots a_1 a_0)_2$, *niin* $k > 0$ *jos ja vain jos* $a_{l-1} > 0$.

Todistus. (vrt. [5, s. 21]). Koska esityksen $(a_{l-i} \dots a_1 a_0)_2$ pituus on l, niin $a_{l-1} \neq 0$. Tehdään nyt induktio pituuden l suhteen. Väite pätee selvästi, kun $l = 1$. Oletaan että $l > 1$.

Jos $a_0 = 0$, merkitään $k^* = (a_{l-i} \dots a_1)_2$. Nähdään, että

$$k > 0 \Leftrightarrow 2k^* > 0 \Leftrightarrow n^* > 0 \Leftrightarrow a_{l-1} > 0,$$

missä oikeanpuoleisin ekvivalenssi seuraa induktioperiaatteesta.

Jos $a_0 \neq 0$, niin

$$(a_{l-i} \dots a_1 a_0)_2 = (a_{l-i} \dots a_w 0 \dots 0 a_0)_w.$$

Nyt $(a_{l-i} \dots a_1 a_0)_w$ on w-leveässä NAF-muodossa, joten

$$k = 2^w (a_{l-i} \dots a_w)_2 + a_0, \text{ missä } -2^{w-1} < a_0 \leq 2^{w-1}.$$

Koska $a_{l-1} \neq 0$, niin $(a_{l-i} \dots a_w)_2 \neq 0$, joten

$$k > 0 \Leftrightarrow (a_{l-i} \dots a_w)_2 > 0 \Leftrightarrow a_{l-1} > 0,$$

missä oikeanpuoleisin ekvivalenssi seuraa induktioperiaatteesta. □

Lause 4.4. *Esityksen $NAF_w(k)$ pituus on maksimissaan yhden suurempi kuin luvun $|k|$ binääriesityksen pituus, kun $w \geq 2$.*

Todistus. (vrt. [5, s. 22-23]). Olkoon l esityksen $NAF_w(|k|)$-pituus ja m luvun $|k|$ binääriesityksen pituus. Jos $k = 0$, niin $l = m = 0$, joten väite pätee. Oletaan, että $k \neq 0$. Olkoon $NAF_w(|k|) = (a_{l-1} ... a_1 a_0)_w$. Nyt $|k|$ on positiivinen, joten lemman 4.1 perusteella $a_{l-1} \geq 1$. Nyt

$$\begin{aligned}
& (a_{l-1} a_{l-2} \dots a_1 a_0)_w = |k| \\
\Rightarrow \quad & (1 a_{l-2} \dots a_1 a_0)_w \leq |k| \\
\Rightarrow \quad & (1 \underbrace{00 \dots 0a}_{w} \underbrace{00 \dots 0a}_{w} \dots)_w \leq |k| \\
\Rightarrow \quad & (\underbrace{11 \dots 1}_{w} \underbrace{a a \dots a}_{w} \underbrace{a a \dots a}_{w} \dots)_w \leq (2^{w-1} + \dots + 2^2 + 2^1 + 1) |k| ,
\end{aligned}$$

missä $a = -(2^{w-1} - 1)$. Tutkitaan viimeistä vasemmalla puolella olevaa lauseketta. Vasemmalta oikealle se sisältää sarjan lukuja 1, joita seuraa sarja lukuja a ja lopussa on sarja lukuja 0 (mahdollisesti ei yhtään). Korvataan luvut 0 luvuilla a. Nyt

$$\begin{aligned}
& (\underbrace{11 \dots 1}_{w} \underbrace{a a \dots a}_{l-1})_2 \leq (2^{w-1} + \dots + 2^2 + 2^1 + 1) |k| \\
\Rightarrow \quad & 2^{l-1}(2^w - 1) + a(2^{l-1} - 1) \leq (2^w - 1) |k| \\
\Rightarrow \quad & 2^{l-1}(2^w - 1) - (2^{w-1} - 1)(2^{l-1} - 1) \leq (2^w - 1) |k| \\
\Rightarrow \quad & 2^{l-1} - \frac{2^{w-1} - 1}{2^w - 1}(2^{l-1} - 1) \leq |k| \\
\Rightarrow \quad & 2^{l-1} - \frac{1}{2}(2^{l-1} - 1) < |k| \ \{\text{Arviointi alaspäin}\} \\
\Rightarrow \quad & 2 \cdot 2^{l-2} - 2^{l-2} + \frac{1}{2} < |k| \\
\Rightarrow \quad & 2^{l-2} + \frac{1}{2} < |k| \\
\Rightarrow \quad & 2^{l-2} < |k| .
\end{aligned}$$

Luvun $|k|$ binääriesityksen perusteella tiedetään, että $|k| < 2^m$ joten,

$$\begin{aligned} 2^{l-2} &< 2^m \\ \Rightarrow \quad & l-2 < m \\ \Rightarrow \quad & l-m \leq 1. \end{aligned}$$

□

4.1.3 w-leveä NAF-lomitusmenetelmä

w-leveä NAF-lomitusmenetelmä (vrt. [2, 3.2]) käyttää w-leveää NAF-esitystä.

Esilaskentavaihe on vastaava kuin ikkunalomitusmenetelmässä. Kun $i = 1, 2$, lasketaan e_iP_i kaikille parittomille luvuille e_i, joille pätee $1 \leq e_i \leq 2^{w_i}-1$ (pisteiden vasta-alkioita $-e_iP_i$ ei tarvitse esilaskea, koska vasta-alkion määrittäminen on helppoa). Epätriviaaleja taulukon alkioita on $2^{w_1-1}+2^{w_2-1}-2$ kappaletta, joista jokainen vaatii yhden yhteenlaskun. Lisäksi tarvitaan yksi kahdennus jokaista ikkunaa $w_i > 1$ kohti (pisteen $2P_i$ määritys).

Algoritmi 4.6 w-leveä NAF -lomitusmenetelmä

Input: Ikkunoiden leveydet w_0 ja w_1, $k_0 = (k_{0,t-1}, \ldots, k_{0,1}, k_{0,0})_2$ ja $k_1 = (k_{1,t-1}, \ldots, k_{1,1}, k_{1,0})_2$, $P_0, P_1 \in E(\mathbb{F}_q)$.
Output: $k_0P_0 + k_1P_1$

1: *Esilaskennat* Laske $\forall i \in \{1, 2\}$ kaikki skalaarikertolaskut eP_i missä e_i on pariton ja $1 \leq e \leq 2^{w_i} - 1$. Lisäksi lasketaan algoritmilla 4.4 esitykset $NAF_{w_i+1}(k_i) = (k_{i,t} \ldots k_{i,0})_{w_i+1}$.
2: $Q \leftarrow \infty$
3: **for** $j = t$ downto 0 **do**
4: $\quad Q \leftarrow 2Q$
5: $\quad$ **for** $i = 0$ to 1 **do**
6: $\quad\quad$ **if** $k_{i,j} \neq 0$ **then**
7: $\quad\quad\quad Q \leftarrow Q + k_{i,j}P$
8: $\quad\quad$ **end if**
9: $\quad$ **end for**
10: **end for**
11: **return** Q

Algoritmin lisäysten määrä seuraa suoraan lauseesta 4.3, joten algoritmin 4.6 lisäysten ja kahdennusten määrien odotusarvot ovat

$$\left(\frac{t}{w_1+2} + \frac{t}{w_2+2}\right) L \text{ ja } aK, \tag{4.5}$$

missä $t - max(w_i) \leq a \leq t$.

Esimerkki 4.4. Esitetään, miten algoritmi 4.6 laskee multiskalaarikertolaskun $53P_0 + 42P_1$, kun $w_0 = 2$ ja $w_1 = 3$.

Esilasketaan pisteet $3P_0, 3P_1$, $5P_1$ ja esitykset $53 = (100\bar{1}00\bar{3})_3$ ja $42 = (100050)_4$ algoritmilla 4.4. Kuvataan nyt alla olevassa taulukossa, mitä algoritmin pääsilmukassa tapahtuu. Tässä kohdat j ja Q ovat algoritmin muuttujia ja kohdat P_0 ja P_1 kertovat, mikä pisteen esilaskettu monikerta lisätään tulokseen.

j	P_0	P_1	Q
6	P_0		P_0
5		P_1	$2P_0 + P_1$
4			$4P_0 + 2P_1$
3	$-P_0$		$7P_0 + 4P_1$
2			$14P_0 + 8P_1$
1		$5P_1$	$28P_0 + 21P_1$
0	$-3P_0$		$53P_0 + 42P_1$

Tarvitaan siis viisi lisäystä ja kuusi kahdennusta, joiden lisäksi esilaskentoihin tarvitaan kaksi kahdennusta ja kolme lisäystä.

4.2 Samanaikaiset menetelmät

Samanaikaisissa menetelmissä tulos kootaan avustavan taulukon avulla siten, että tauluun tallennetaan esilaskentavaiheessa myös pisteiden yhteenlaskuja. Skalaarien-esitykset yhdistetään ja käydään läpi rinnakkain. Käydään läpi muutama tarvittava määritelmä.

Määritelmä 4.4. Olkoon *yhdistetty esitys* K esitysten k_0 ja k_1 yhdiste, jossa esitykset yhdistetään $2 \times t - 1$ kokoiseen taulukkoon, missä sarakkeella $i \in \{0, 1\}$ on esitys k_i.

Määritelmä 4.5. Olkoon *yhdistetty Hamming-paino* yhdistetyn esityksen K nollasarakkeiden määrä. Vastaavasti *yhdistetty Hamming-tiheys* on yhdistetty Hamming-paino jaettuna yhdistetyn esityksen pituudella.

Esitetään ensin selkeyden vuoksi yksinkertainen samanaikainen menetelmä (tunnetaan nimellä *Shamir's trick*).

Algoritmi 4.7 Yksinkertainen samanaikainen menetelmä

Input: Kokonaisluvut $k_0 = (k_{0,t-1} \dots k_{0,1}k_{0,0})_2$ $k_1 = (k_{1,t-1} \dots k_{1,1}k_{1,0})_2$ ja $P_0, P_1 \in E(\mathbb{F}_q)$.

Output: $k_0P_0 + k_1P_1$

1: *Esilaskennat* Piste $P_0 + P_1$
2: $Q \leftarrow \infty$
3: **for** $j = t - 1$ downto 0 **do**
4: $\quad Q \leftarrow 2Q$
5: $\quad$ **if** $k_{0,j} \neq 0$ tai $k_{1,j} \neq 0$ **then**
6: $\quad\quad Q \leftarrow Q + (k_{0,j}P_0 + k_{1,j}P_1)$ {Esilaskettu}
7: $\quad$ **end if**
8: **end for**
9: **return** Q

Nyt nähdään, että algoritmissa yhteenlaskujen määrä on sama kuin lukujen yhdistetyn binääriesityksen yhdistetty Hamming-paino. Siis lisäysten ja kahdennusten määrien odotusarvot ovat

$$\left(\frac{3}{4}(t-1)\right) L \text{ ja } (t-1)K. \tag{4.6}$$

4.2.1 Samanaikainen 2^w-äärimenetelmä

Samanaikainen 2^w-äärimenetelmä [2, 2.1] tutkii kerralla kummankin skalaarin binääriesityksestä w kappaletta bitteja jokaisessa laskennan vaiheessa, eli yhteensä käsitellään kerralla $2w$ bittiä.

Esilaskentavaiheessa lasketaan $l_0P_0 + l_1P_1$ kaikilla nollasta poikkeavilla pareilla $(l_0, l_1) \in \{0, \dots, 2^w - 1\}^2$.

Epätriviaaleja yhdistelmiä on $2^{2w} - 1 - 2$ kappaletta. Näistä $2^{2(w-1)} - 1$ kappaletta voidaan laskea kahdentamalla muita pisteitä ja loput $2^{2w} - 2^{2(w-1)} - 2$ kappaletta tarvitsevat yhden lisäyksen.

Algoritmi 4.8 Samanaikainen 2^w-äärimenetelmä

Input: Ikkunan leveys w, $k_0 = (k_{0,t-1} \dots k_{0,1}k_{0,0})_2$ ja $k_1 = (k_{1,t-1} \dots k_{1,1}k_{1,0})_2$, $P_0, P_1 \in E(\mathbb{F}_q)$.
Output: $k_1P_2 + k_2P_2$

1: *Esilaskennat* Laske $l_0P_0 + l_1P_1$ kaikilla nollasta poikkeavilla $(l_0, l_1) \in \{0, \dots, 2^w - 1\}^2$.
2: $Q \leftarrow \infty$
3: **for** $j = \lfloor (t-1)/w \rfloor$ downto 0 **do**
4: **for** $i = 1$ to w **do**
5: $Q \leftarrow 2Q$
6: **end for**
7: **if** $(k_{0,j+w-1} \dots k_{0,j})_2, (k_{1,j+w-1} \dots k_{1,j})_2 \neq (0 \dots 0)$ **then**
8: $Q \leftarrow Q + (k_{0,j+w-1} \dots k_{0,j})_2 P_0 + (k_{1,j+w-1} \dots k_{1,j})_1 P_1$
9: **end if**
10: **end for**
11: **return** Q

Yhteenlaskujen määrä on maksimissaan t/w. Lisäksi algoritmi ohittaa ikkunan (Algoritmin 4.8 rivi 6), jos se sisältää pelkkiä nollia. Tämän todennäköisyys on $(1/4)^w$, joten algoritmin 4.8 lisäysten ja kahdennusten määrien odotusarvot ovat

$$\left(t \cdot \frac{1 - (1/2)^w}{w}\right) L \text{ ja } \left(\left\lfloor \frac{(t-1)}{w} \right\rfloor w\right) K. \tag{4.7}$$

Esimerkki 4.5. Esitetään miten algoritmi 4.8 laskee multiskalaarikertolaskun $53P_0 + 42P_1$, kun $w = 2$.

Esilasketaan pisteet: $2P_0$; $2P_1$; $3P_0$; $3P_1$; $P_0 + P_1$; $2P_0 + 2P_1$; $2P_0 + P_1$; $P_0 + 2P_1$; $3P_0 + P_1$; $P_0 + 3P_1$; $3P_0 + 2P_1$; $2P_0 + 3P_1$ ja $3P_0 + 3P_1$. Nyt

$$53 = (\overbrace{11}\ \overbrace{01}\ \overbrace{01})_2$$
$$42 = (\underbrace{10}_{2}\ \underbrace{10}_{1}\ \underbrace{10}_{0})_2,$$

missä aaltosulkeet esittävät, miten ikkunat asettuvat esityksen päälle. Aaltosulkeitten alla oleva numero kertoo muuttujan j arvon. Kuvataan seuraavalla taulukolla mitä algoritmin pääsilmukassa tapahtuu. Tässä j ja Q ovat algoritmin muuttujia. Huomaa, että arvon j muuttuessa Q kahdennetaan kahdesti.

j	esilaskettu arvo	Q
2	$3P_0 + 2P_1$	$3P_0 + 2P_1$
1	$P_0 + 2P_1$	$13P_0 + 10P_1$
0	$P_0 + 2P_1$	$53P_0 + 42P_1$

Tarvitaan siis kolme lisäystä ja neljä kahdennusta, joiden lisäksi esilaskentoihin tarvitaan kolme kahdennusta ja kymmenen lisäystä.

4.2.2 Samanaikainen liukuva ikkuna -menetelmä

Samanaikainen liukuva ikkuna -menetelmä [2, 2.2] on parannettu versio pykälässä 4.2.1 esitellystä menetelmästä. Liukuvan ikkunan käyttö vähentää esilaskettavat yhdistelmät $(l_0, l_1) \in \{0, \ldots, 2^w - 1\}^2$ niihin, joissa l_0 tai l_1 on pariton.

Esilaskentavaiheessa lasketaan $l_0P_0 + l_1P_1$ kaikilla nollasta poikkeavilla 2-kertaisilla yhdistelmillä $(l_0, l_1) \in \{0, \ldots, 2^w - 1\}^2$, joissa l_0 tai l_1 on pariton.

Epätriviaaleja yhdistelmiä on $2^{2w} - 2^{2(w-1)} - 2$ kappaletta, joista jokainen vaatii yhden lisäyksen. Lisäksi tarvitaan kaksi kappeletta kahdennuksia, kun $w > 1$.

Algoritmi 4.9 Samanaikainen liukuva ikkuna -menetelmä

Input: Ikkunan leveys w, $k_0 = (k_{0,t-1} \dots k_{0,1} k_{0,0})_2$ ja $k_1 = (k_{1,t-1} \dots k_{1,1} k_{1,0})_2$, $P_0, P_1 \in E(\mathbb{F}_q)$.
Output: $\sum_{i=1}^{n} k_i P_i$

1: *Esilaskennat* Laske $l_0 P_0 + l_1 P_1$ kaikilla nollasta poikkeavilla pareilla $(l_0, l_1) \in \{0, \dots, 2^w - 1\}^2$, missä l_0 tai l_1 pariton.
2: $Q \leftarrow \infty$
3: $j \leftarrow t - 1$
4: **while** $j \geq 0$ **do**
5: **if** $k_{0,j} = 0$ ja $k_{1,j} = 0$ **then**
6: $Q \leftarrow Q^2$
7: $j \leftarrow j - 1$
8: **else**
9: $j_{new} \leftarrow max(j - w, -1)$
10: $J \leftarrow j_{new} + 1$
11: **while** $k_{0,J} = 0$ ja $k_{1,J} = 0$ **do**
12: $J \leftarrow J + 1$ {Nyt $j > J \geq j_{new}$}
13: **end while**
14: **for** $i = 0$ to 1 **do**
15: $e_i \leftarrow (k_{i,j} \dots k_{i,J})_2$
16: **end for**
17: **while** $j \geq J$ **do**
18: $Q \leftarrow Q^2$
19: $j \leftarrow j - 1$
20: **end while**
21: $Q \leftarrow Q + (e_0 P_0 + e_1 P_1)$ {Esilaskettu}
22: **while** $j \geq j_{new}$ **do**
23: $Q \leftarrow Q^2$
24: $j \leftarrow j - 1$
25: **end while**
26: **end if**
27: **end while**
28: **return** Q

Algoritmi osaa huomioida ikkunoiden väliin jäävät nollasarakkeet. Niiden lukumäärän odotusarvo on $\sum_{m \geq 1} \frac{1}{4}^m = \frac{\frac{1}{4}}{1 - \frac{1}{4}} = \frac{1}{3}$, joten algoritmin 4.9 lisäysten ja kahdennusten määrien odotusarvot ovat

$$\left(\frac{t}{w + \frac{1}{3}} \right) L \text{ ja } (t-1)\, K. \tag{4.8}$$

Esimerkki 4.6. Esitetään, miten algoritmi 4.9 laskee multiskalaarikertolaskun $53P_0 + 40P_1$, kun $w = 2$.

Esilasketaan pisteet: $3P_0$; $3P_1$; $P_0 + P_1$; $2P_0 + P_1$; $P_0 + 2P_1$; $3P_0 + P_1$; $P_0 + 3P_1$; $3P_0 + 2P_1$; $2P_0 + 3P_1$ ja $3P_0 + 3P_1$.

Nyt

$$53 = (\overbrace{11}\ \overbrace{01}\ 0\ \overbrace{1}\ )_2$$
$$40 = (\underbrace{10}_{2}\ \underbrace{10}_{1}\ 0\ \underbrace{0}_{0}\ )_2,$$

missä sulkeet esittävät, miten ikkunat asettuvat esityksen päälle. Kuvataan vielä taulukolla, mitä algoritmin pääsilmukassa tapahtuu. Tässä j ja Q ovat algoritmin muuttujia.

j	esilaskettu arvo	Q
5		
4	$3P_0 + 2P_1$	$3P_0 + 2P_1$
3		$6P_0 + 4P_1$
2	$P_0 + 2P_1$	$13P_0 + 10P_1$
1		$26P_0 + 20P_1$
0	$P_0 + 2P_1$	$53P_0 + 42P_1$

Tarvitaan siis kolme lisäystä ja neljä kahdennusta, joiden lisäksi esilaskentoihin tarvitaan kaksi kahdennusta ja kymmenen lisäystä.

4.2.3 JSF (The Joint Sparse Form)

JSF on yhdistetty etumerkillinen esitys kahdelle skalaarille. Se on NAF-esityksen laajennus, ja suunniteltu nopeuttamaan algoritmia 4.9. Tämä pykälä noudattelee Solinasin artikkelia (ks. [3]).

Määritelmä 4.6. Olkoon K yhdistetty etumerkillinen esitys skalaareista k_0 ja k_1. Nyt K on JSF-muodossa, jos kohdat (i)-(iii) pätevät.

(i) Mistä tahansa kolmesta peräkkäisestä sarakkeesta ainakin yksi on nollasarake. Toisin sanoen millä tahansa i ja j pätee $k_{i,j+u} = k_{1-i,j+u} = 0$, kun $u = 0$ tai ± 1.

(ii) Peräkkäiset arvot eivät ole vastakkaismerkkisiä. Toisin sanoen tapaus $k_{i,j} * k_{i,j+1} = -1$ on mahdoton.

(iii) Jos $k_{i,j+1} * k_{i,j} \neq 0$, niin $k_{1-i,j+1} = \pm 1$ ja $k_{1-i,j} = 0$.

On syytä todeta, että jos $k_0 = 0$, niin k_1 on NAF-muodossa.

Seuraavassa lauseessa osoitetaan JSF-esityksen yksikäsitteisyys. JSF-esitys on aito, jos ainakin yksi sen vasemmaisista arvoista poikkeaa nollasta.

Lause 4.5. *Positiivisella kokonaislukuparilla k_0 ja k_1 on olemassa vain yksi aito JSF-esitys.*

Todistus. (Vrt. [3, kappale 5]) Tehdään vastaoletus, että on olemassa kaksi toisistaan poikkevaa aitoa JSF-esitystä. Siis

$$\begin{aligned} k_0 =& (u_{0,m} \dots u_{0,1} u_{0,0})_2 = (w_{0,n} \dots w_{0,1} w_{0,0})_2 \\ k_1 =& (u_{1,m} \dots u_{1,1} u_{1,0})_2 = (w_{1,n} \dots w_{1,1} w_{1,0})_2. \end{aligned}$$

Koska esitykset ovat erillisiä, joillain i ja j pätee $u_{i,j} \neq w_{i,j}$. Olkoon g pienin luvun j arvoista, joilla esitykset eroavat. Olkoon nyt

$$\begin{aligned} r_0 =& (u_{0,g-1} \dots u_{0,1} u_{0,0})_2 = (w_{0,g-1} \dots w_{0,1}, w_{0,0})_2 \\ r_1 =& (u_{1,g-1} \dots u_{1,1} u_{1,0})_2 = (w_{1,g-1} \dots w_{1,1}, w_{1,0})_2, \end{aligned}$$

ja

$$\begin{aligned} n_0 =& (k_0 - r_0) \\ n_1 =& (k_1 - r_1). \end{aligned}$$

Täten

$$\begin{aligned} n_0 =& (u_{0,m} \dots u_{0,g+1} u_{0,g})_2 = (w_{0,n} \dots w_{0,g+1} w_{0,g})_2 \\ n_1 =& (u_{1,m} \dots u_{1,g+1} u_{1,g})_2 = (w_{1,n} \dots w_{1,g+1} w_{1,g})_2. \end{aligned}$$

Koska esitykset eroavat kun $j = g$, voidaan olettaa, että

$$u_{0,g} \neq w_{0,g}.$$

Lukujen k_0 ja k_1 rooleja vaihtamalla voidaan olettaa, että n_0 on pariton, muutoin pätisi $u_{0,g} = w_{0,g}$. Parittomuudesta seuraa, että $u_{0,g}$ ja $w_{0,g}$ saavat arvot -1 ja 1. Voidaan olettaa, että $u_{0,g} = 1$ ja $w_{0,g} = -1$.

Oletetaan että $n_0 \equiv 1 \pmod 4$ (tapaus $n_0 \equiv 3 \pmod 4$ on vastaava). Koska $u_{0,g} = 1$, niin $u_{0,g+1} = 0$. Koska $w_{0,g} = -1$, niin $w_{0,g+1}$ on pariton ja kohdan (ii) perusteella $w_{0,g+1} w_{0,g} \neq -1$, joten $w_{0,g+1} = -1$. Kohdasta (iii) seuraa, että $w_{1,g} = 0$ ja $w_{1,g+1} = \pm 1$, joten $n_1 \equiv 2 \pmod 4$. Täten $u_{1,g} = 0$ ja $u_{1,g+1} = \pm 1$.

Kohdasta (i) seuraa, että molemmissa esityksissä on nollasarakkeet kohdassa $g + 2$. Siis

$$u_{0,g+2} = u_{1,g+2} = w_{0,g+2} = w_{1,g+2} = 0.$$

Nyt

$$\begin{aligned} (u_{0,g+2} u_{0,g+1} u_{0,g})_2 =& (001)_2, \text{ eli } n_0 \equiv 1 \pmod 8 \\ (w_{0,g+2} w_{0,g+1} w_{0,g})_2 =& (0\bar{1}\bar{1})_2, \text{ eli } n_0 \equiv 5 \pmod 8. \end{aligned}$$

Tämä on ristiriita, joten väite seuraa. □

Osoitetaan, että kaikille kokonaislukupareille (k_0, k_1) voidaan määrittää JSF-esitys. Helpoin tapa on esittää algoritmi esityksen laskemiseen.

Algoritmi 4.10 JSF-esityksen määritys

Input: Positiiviset kokonaisluvut k_0 ja k_1, joista ainakin toinen poikkeaa nollasta.

Output: Parin k_0 k_1 JSF-esitys.

$n_0 \leftarrow k_0$ ja $n_1 \leftarrow k_1$
$j \leftarrow 0$
$d_0 \leftarrow 0$ ja $d_1 \leftarrow 0$
while $n_0 + d_0 > 0$ tai $n_1 + d_1 > 0$ **do**
  $l_0 \leftarrow d_0 + n_0$
  $l_1 \leftarrow d_1 + n_1$
  for $i = 0$ to 1 **do**
    if l_i parillinen **then**
      $u \leftarrow 0$
    else
      $u \leftarrow l_i \pmod 4$
      if $l_i \equiv \pm 3 \pmod 8$ ja $l_{1-i} \equiv 2 \pmod 4$ **then**
        $u \leftarrow -u$
      end if
    end if
    $u_{i,j} \leftarrow u$
  end for
  for $i = 0$ to 1 **do**
    if $2d_i = 1 + u_{i,j}$ **then**
      $d_i \leftarrow 1 - d_i$
    end if
    $n_i \leftarrow \lfloor n_i/2) \rfloor$
  end for
  $j \leftarrow j + 1$
end while
return u_0 ja u_1

Esimerkki 4.7. Lukujen 53 ja 42 JSF-esitys on

$$53 = (100\bar{1}0\bar{1}\bar{1})_2$$
$$42 = (0101010)_2.$$

Seuraavassa taulukossa näytetään, miten esitys muodostuu algoritmilla 4.10.

$l_0 = n_0 + d_0$	u_0	$l_1 = n_1 + d_1$	u_1
$53 + 0$	$\bar{1}$	$42 + 0$	0
$26 + 1$	$\bar{1}\bar{1}$	$21 + 0$	10
$13 + 1$	$0\bar{1}\bar{1}$	$10 + 0$	010
$6 + 1$	$\bar{1}0\bar{1}\bar{1}$	$5 + 0$	1010
$3 + 1$	$0\bar{1}0\bar{1}\bar{1}$	$2 + 0$	01010
$1 + 1$	$00\bar{1}0\bar{1}\bar{1}$	$1 + 0$	101010
$0 + 1$	$100\bar{1}0\bar{1}\bar{1}$	$0 + 0$	0101010
$0 + 0$		$0 + 0$	

Laajenneteaan seuraavaksi algoritmia 4.10 siten, että se hyväksyy myös etumerkilliset esitykset syötteinä. Tämä on välttämätöntä todistettaessa JSF-esityksen ominaisuuksia.

Määritelmä 4.7. Etumerkillinen numeroesitys $(u_m, \ldots, u_1, u_0)$ on *pelkistetty*, jos kaikilla j pätee $u_j u_{j-i} \neq -1$. Yhdistetty etumerkillinen numeroesitys on *pelkistetty*, jos se koostuu kahdesta pelkistetystä etumerkillisestä numeroesityksestä (JSF on pelkistetty määritelmän 4.6 kohdan (ii) perusteella).

Algoritmi 4.11 laajennettu JSF-esityksen määritys

Input: Pelkistetty yhdistetty etumerkkillinen numeroesitys

$$k_0 = (e_{0,m-1} \dots e_{0,1} e_{0,0})_2$$
$$k_1 = (e_{1,m-1} \dots e_{1,1} e_{1,0})_2$$

Output: Parin k_0 k_1 JSF-esitys.

1: $a_0 \leftarrow e_{0,0}$, $b_0 \leftarrow e_{0,1}$, $c_0 \leftarrow e_{0,2}$, $d_0 \leftarrow 0$
2: $a_1 \leftarrow e_{1,0}$, $b_0 \leftarrow e_{1,1}$, $c_1 \leftarrow e_{1,2}$, $d_1 \leftarrow 0$
3: $d_0 \leftarrow 0$ ja $d_1 \leftarrow 0$
4: **for** $j = 0$ to m **do**
5: **for** $i = 0$ to 1 **do**
6: **if** $d_i \equiv a_i \pmod 2$ parillinen **then**
7: $u \leftarrow 0$
8: **else**
9: $u \leftarrow (d_i + 2b_i + a_i) \mod 4$
10: **if** $d_i + 4c_i + 2b_i + a_i \equiv \pm 3 \pmod 8$ ja $d_{1-i} + 2b_{1-i} + a_{1-i} \equiv 2 \pmod 4$ **then**
11: $u \leftarrow -u$
12: **end if**
13: **end if**
14: $u_{i,j} \leftarrow u$
15: $\{\mathbf{R_{i,j}} \leftarrow (d_i, c_i, b_i, a_i)\}$
16: **end for**
17: $\{\mathbf{S_i} \leftarrow (\mathbf{R_{0,j}}, \mathbf{R_{1,j}})\}$
18: **for** $i = 0$ to 1 **do**
19: **if** $2d_i = 1 + u_{i,j}$ **then**
20: $d_i \leftarrow (d_i + a_i - u_{i,j})/2$
21: $a_i \leftarrow b_i$, $b_i \leftarrow c_i$, $c_i \leftarrow e_{i,j+3}$)
22: **end if**
23: $n_i \leftarrow \lfloor n_{i,j}/2) \rfloor$
24: **end for**
25: **end for**
26: **return** $JSF(k_0, k_1)$

Merkinnät **R** ja **S** on lisätty helpottamaan algoritmin analysointia.

Selvästi algoritmi 4.11 vastaa algoritmia 4.10 tapauksissa, joissa syötteet on esitetty binäärimuodossa.

Analysoidaan algoritmia tarkemmin. Olkoon $\mathbf{S_j}$ algoritmin *tila* ja vektori $(u_{0,j}, u_{1,j})$ tilan $\mathbf{S_j}$ tulos. Kuvaillaan algoritmin toimintaa seuraavasti: Algoritmin pääsilmukan iteraatiossa j tila $\mathbf{S_{j-1}}$ tuottaa tuloksen $(u_{0,j-1}, u_{1,j-1})$, ja algoritmi siirtyy tilaan $\mathbf{S_j}$.

Listataan seuraavaksi kaikki mahdolliset tilojen $\mathbf{R}_{j,i}$ tuottamat vektorit.

Erillaisia mahdollisuuksia on 51 kappaletta (ei $3^4 = 81$, koska esitys on pelkistetty), missä $\mathbf{R_{i,j}} = (d_i, c_i, b_i, a_i)$. Määritellään kaikille $\mathbf{R_{i,j}}$

$$k_{i,j} := 4c_i + 2b_i + a_i \qquad l_{i,j} := d_i + k_i.$$

Merkinnät vastaavat algoritmin 4.10 merkintöjä tapauksissa, joissa algoritmi pätee.

Määritellään tiloille $\mathbf{S_j}$ kahdeksan tilaa riippuen arvosta $l_{i,j}$.

$\mathbf{S_j}$	$\mathbf{R_{0,j}}$	$\mathbf{R_{1,j}}$
C	$l_{0,j}$ parillinen	$l_{1,j}$ parillinen
D	$l_{0,j} \equiv 0 \pmod 4$	$l_{1,j}$ pariton
E	$l_{0,j} \equiv 2 \pmod 4$	$l_{1,j} \equiv \pm 1 \pmod 8$
F	$l_{0,j} \equiv 2 \pmod 4$	$l_{1,j} \equiv \pm 3 \pmod 8$
G	$l_{0,j}$ pariton	$l_{1,j} \equiv 0 \pmod 4$
H	$l_{0,j} \equiv \pm 1 \pmod 8$	$l_{1,j} \equiv 2 \pmod 4$
J	$l_{0,j} \equiv \pm 3 \pmod 8$	$l_{1,j} \equiv 2 \pmod 4$
K	$l_{0,j}$ pariton	$l_{1,j}$ pariton

Tutkitaan nyt, miten tilat seuraavat toisiaan. Tämä on helppoa käymällä läpi kaikki tapaukset.

- Jos $l_{i,j} \equiv 0 \pmod 4$ ja $l_{1-i,j}$ on pariton, niin $l_{i,j+1}$ on parillinen.
- Jos $l_{i,j} \equiv 2 \pmod 4$ ja $l_{1-i,j}$ on pariton, niin $l_{i,j+1}$ on pariton.
- Jos $l_{i,j}$ on pariton ja $l_{1-i,j} \not\equiv 2 \pmod 4$, niin $l_{i,j+1}$ on parillinen.
- Jos $l_{i,j} \equiv 1 \pmod 4$ ja $l_{1-i,j} \equiv 2 \pmod 4$, niin $l_{i,j} \equiv 0 \pmod 4$.
- Jos $l_{i,j} \equiv 3 \pmod 4$ ja $l_{1-i,j} \equiv 2 \pmod 4$, niin $l_{i,j+1}$ on pariton.

Nyt voidaan muodostaa taulukko, josta nähdään miten tilat seuraavat toisiaan.

$\mathbf{S_j}$	$u_{0,j}$	$u_{1,j}$	$\mathbf{S_{j+1}}$
C	0	0	$*$
D	0	± 1	C
E	0	± 1	G
F	0	± 1	K
G	± 1	0	C
H	± 1	0	D
J	± 1	0	K
K	± 1	± 1	C

Tässä $*$ merkitsee, että tilaa C voi seurata mikä tahansa tila. Kaikissa muissa tapauksissa tila $\mathbf{S_j}$ määrittää tilan $\mathbf{S_{j+1}}$

On kinnostavaa tietää, milloin $u_{0,j} = u_{1,j} = 0$, joten ryhmitellään tilat tämän mukaan kolmeen uuteen tilaan.

$\mathbf{S_j}$	vanhat tilat	$u_{0,j} = u_{1,j} = 0$?	$\mathbf{S_{j+1}}$
A	E, F, H, J	Ei	B
B	D, G, K	Ei	C
C	C	Kyllä	A, B, C

Seuraavat kaksi lausetta perustuvat edelliseen analyysiin.

Lause 4.6. *Algoritmi 4.11 tuottaa aina JSF-muodossa olevan yhdistetyn etumerkillisen -esityksen.*

Todistus. (vrt. [3, kappale 7]) On selvää että algoritmin tuottama esitys on yhdistetty etumerkillinen -esitys kokonaisluvuista k_0 ja k_1. Pitää siis osoittaa, että esitys on JSF-muodossa, eli määritelmän 4.6 ehdot (i)-(iii) pätevät.

(i) Ehto pätee, jos kaikilla j ainakin yksi $\mathbf{S_{j-1}}$,$\mathbf{S_j}$ tai $\mathbf{S_{j+1}}$ on tilassa C. Oletetaan, että $\mathbf{S_{j-1}}$ ei ole tilassa C. Tällöin se on joko tilassa B (jolloin $\mathbf{S_j}$ on tilassa C) tai A (jolloin $\mathbf{S_{j+1}}$ on tilassa C).

(ii) Oletetaan, että $u_{0,j} \neq 0$ ja $u_{0,j+1} \neq 0$. Nyt edellisen perusteella $\mathbf{S_j}$ on tilassa J ja $\mathbf{S_{j+1}}$ on tilassa K. On helppoa laskea, että kaikissa tapauksissa $u_{0,j} = u_{0,j+1}$.

(iii) Oletetaan, että $u_{0,j}u_{0,j+1} \neq 0$. $\mathbf{S_j}$ on tilassa J ja $\mathbf{S_{j+1}}$ tilassa K. Nyt Aikaisemmasta taulukosta nähdään, että $u_{1,j} = 0$ ja $u_{1,j+1} = \pm 1$. Tapaus $u_{1,j}u_{1,j+1} \neq 0$ käsitelläänm vastaavasti.

□

Lause 4.7. *JSF-esityksen yhdistetyn Hamming-tiheyden odotusarvo on* $1/2$.

Todistus. (vrt. [3, kappale 9]) Kaikilla $j \geq 0$, merkitköön $Tod_j(A)$ todennäköisyyttä sille, että $\mathbf{S_j}$ on tilassa A satunnaisilla syötteillä k_0 ja k_1 väliltä $[0, 2^m]$, missä $m \geq j + 3$. (Tässä algoritmin 4.11 tuottamat arvot $u_{0,j}$ ja $u_{1,j}$ ovat riippumattomia arvoista $e_{0,n}$ ja $e_{1,n}$, missä $n \geq j + 3$, joten $Tod_j(A)$ on riippumaton arvosta m.) Todennäköisyydet $Tod_j(B)$ ja $Tod_j(C)$ määritellään vastaavasti. Nyt rajoituksesta $m \geq j + 3$ seuraa, että yhdistetyn Hamming -tiheyden määrityksessä ei voida ottaa huomioon kolmea vasemman puoleista arvoa. Kutsutaan tätä *vajaaksi* yhdistetyksi Hamming -tiheydeksi. Asymptoottisesti kolmella viimeisellä arvolla ei ole merkitystä.

Tiedetään, että $u_{0,j} = u_{1,j} = 0$ on tulos vain tilasta C. Täten vajaan Hamming -tiheyden odotusarvo luvuille, jotka ovat pienempiä kuin 2^m, on

$$\pi_m = 1 - \frac{1}{m-2} \sum_{j=0}^{m-3} Tod_j(C). \tag{4.9}$$

Nyt Hamming-tiheyden odotusarvo π on

$$\pi = \lim_{m \to \infty} \pi_m.$$

Arvojen π_m määrittämiseksi muodostetaan rekursio. Aiemmin todettiin, että jos $\mathbf{S_{j-1}}$ on tilassa A, niin $\mathbf{S_j}$ on tilassa B, ja jos $\mathbf{S_{j-1}}$ on tilassa B, niin $\mathbf{S_j}$ on tilassa C.

Jos $\mathbf{S_{j-1}}$ on tilassa C, voi $\mathbf{S_j}$ olla jokaisessa tilassa. Tutkimalla arvoja $l_{0,j}$ ja $l_{1,j}$ on helppo määrittää eri tilojen esiintymisen todennäköisyydet. Esimerkiksi todennäköisyys, että $l_{0,j} \equiv 0 \pmod 4$ on $1/4$ ja todennäköisyys, että $l_{1,j}$ on pariton on $1/2$, joten $Tod_j(D) = (1/4) \cdot (1/2) = 1/8$. Vastaavasti saadaan

$$\begin{aligned} Tod_j(C) &= 1/4 \qquad Tod_j(G) = 1/8 \\ Tod_j(D) &= 1/8 \qquad Tod_j(H) = 1/16 \\ Tod_j(E) &= 1/16 \qquad Tod_j(J) = 1/16 \\ Tod_j(F) &= 1/16 \qquad Tod_j(K) = 1/4. \end{aligned}$$

Siis

$$Tod_j(A) = 1/4 \qquad Tod_j(B) = 1/2.$$

Nyt voidaan muodostaa seuraavat lausekkeet.

$$\begin{aligned} Tod_{j+1}(A) &= \frac{1}{4} Tod_j(C) \\ Tod_{j+1}(B) &= \frac{1}{2} Tod_j(C) + Tod_j(A) \\ Tod_{j+1}(C) &= \frac{1}{4} Tod_j(C) + Tod_j(B). \end{aligned}$$

Tiedetään, että

$$Tod_j(A) + Tod_j(B) + Tod_j(C) = 1.$$

Muodostetaan rekursio

$$Tod_{j+1}(C) = 1 - \frac{3}{4} Tod_j(C) - \frac{1}{4} Tod_{j-1}(C).$$

Rekursion alkuarvot ovat

$$\begin{aligned} Tod_0(A) &= 1/4 \\ Tod_0(B) &= 1/2 \\ Tod_0(C) &= 1/4. \end{aligned}$$

Arvo $Tod_1(C) = 1/2 + 1/16 = 9/16$ saadaan edellisistä kaavoista.

Ratkaistaan rekursion yleinen ratkaisu. Kyseessä on toisen asteen differenssiyhtälö

$$C_j + \frac{3}{4}C_{j-1} + \frac{1}{4}C_{j-2} = 1,$$

missä $Tod_j(C) = C_j$.

Ratkaistaan ensin erikoistapaus C_∞. Kun $j = \infty$ niin $C_\infty = C_j = C_{j-1} = C_{j-2}$, joten

$$C_\infty + \frac{3}{4}C_\infty + \frac{1}{4}C_\infty = 1 \quad \Rightarrow C_\infty = \frac{1}{2}.$$

Ratkaistaan seuraavaksi homogeenisen yhtälön $C_j + \frac{3}{4}C_{j-1} + \frac{1}{4}C_{j-2} = 0$ yleinen ratkaisu.

Yhtälön molemmat juuret ovat kompleksisia, joten yleinen ratkaisu on muotoa

$$C_j = Ar\cos(\theta j - \omega),$$

missä $r = \frac{1}{2^j}$, $\cos\theta = -\frac{3/4}{2\sqrt{1/4}} = -\frac{3}{4}$ ja A, ω vakioita.

Määritetään vakiot A ja ω. Tiedetään, että $C_0 = \frac{1}{4}$ ja $C_1 = \frac{9}{16}$, joten saadaan yhtälöpari

$$\begin{cases} \frac{1}{2} + A\cos(-\omega) = -\frac{1}{4} \\ \frac{1}{2} + \frac{1}{2}A\cos(\theta - \omega) = \frac{9}{16} \end{cases} \Rightarrow \begin{cases} A\cos(\omega) = \frac{1}{4} \\ A(\cos\theta\cos\omega + \sin\theta\sin\omega) = \frac{1}{8}. \end{cases}$$

Ylemmästä yhtälöstä saadaan $\cos\omega = -\frac{1}{4A}$, joten $\sin\omega = \pm\sqrt{1 - \frac{1}{16A^2}}$. Lisäksi tiedetään, että $\cos\theta = -\frac{3}{4}$, joten $\sin\theta = \pm\frac{\sqrt{7}}{4}$. Sijoitetaan saadut arvot alempaan yhtälöön

$$\begin{aligned} &\left(-\frac{3}{4}\left(-\frac{1}{4A}\right) \pm \frac{\sqrt{7}}{4}\left(\pm\sqrt{1 - \frac{1}{16A^2}}\right)\right) = \frac{1}{8} \\ &\Rightarrow -\frac{3}{16} \pm A\frac{\sqrt{7}}{4}\left(\pm\sqrt{1 - \frac{1}{16A^2}}\right) = \frac{1}{8} \\ &\Rightarrow \pm\frac{\sqrt{7}}{4}\left(\pm\sqrt{A^2 - \frac{1}{16}}\right) = -\frac{1}{16}. \end{aligned}$$

Jotta ratkaisu olisi reaalinen, on sinien arvot oltava sellaisia, että toinen on

positiivinen ja toinen negatiivinen. Täten

$$\begin{aligned}
&\Rightarrow \sqrt{A^2 - \frac{1}{16}} = \frac{1}{16\sqrt{7}} \\
&\Rightarrow A = \frac{1}{\sqrt{14}} \\
&\Rightarrow \omega = \sin^{-1}\left(\sqrt{1 - \frac{1}{16A}}\right) = \sin^{-1}\left(\frac{1}{\sqrt{8}}\right).
\end{aligned}$$

Siis rekursion yleinen ratkaisu on

$$Tod_j(C) = \frac{1}{2} + \frac{1}{2^j\sqrt{14}}\cos(j\theta - \omega),$$

missä $\theta = \cos^{-1}\left(-\frac{3}{4}\right)$ ja $\omega = \sin^{-1}\left(\frac{1}{\sqrt{8}}\right)$.

Sijoitetaan rekursion yleinen ratkaisu yhtälöön 4.9. Saadaan

$$\begin{aligned}
\pi_m &= 1 - \frac{1}{m-2}\sum_{j=0}^{m-3}\frac{1}{2} + \frac{1}{2^j\sqrt{14}}\cos(j\theta - \omega) \\
&= 1 - \frac{1}{m-2}\left(\frac{m-2}{2} + \sum_{j=0}^{m-3}\frac{1}{2^j\sqrt{14}}\cos(j\theta - \omega)\right) \\
&= \frac{1}{2} - \frac{1}{\sqrt{14}(m-2)}\sum_{j=0}^{m-3}\frac{1}{2^j}\cos(j\theta - \omega) \\
&= \frac{1}{2} - \frac{1}{\sqrt{14}(m-2)}\left(\sum_{j=0}^{m-3}\frac{1}{2^j}\cos j\theta\cos\omega + \sum_{j=0}^{m-3}\frac{1}{2^j}\sin j\theta\sin\omega\right)
\end{aligned}$$

Muokataan Eulerin lauseen avulla summaa

$$\begin{aligned}
\sum_{j=0}^{m-3}\frac{1}{2^j}\cos j\theta\cos\omega &= \cos\omega\sum_{j=0}^{m-3}\frac{1}{2^j}\frac{e^{ij\theta} + e^{-ij\theta}}{2} \\
&= \cos\omega\frac{1}{2}\sum_{j=0}^{m-3}\frac{e^{ij\theta} + e^{-ij\theta}}{2^j} \\
&= \cos\omega\frac{1}{2}\sum_{j=0}^{m-3}\left(\frac{e^{i\theta} + e^{-i\theta}}{2}\right)^j \\
&= \cos\omega\frac{1}{2}\sum_{j=0}^{m-3}(\cos\theta)^j \\
&= \frac{1}{2}\cos\omega\frac{1 - (\cos\theta)^{m-2}}{1 - \cos\theta} \\
&= \frac{1}{2}\cos\omega\frac{1 - (\cos\theta)^{m-2}}{1 - \cos\theta}
\end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{2} \frac{1}{4\sqrt{14}} \frac{1 - (-\frac{3}{4})^{m-2}}{1 + \frac{3}{4}} \\
&= \frac{1 - (-\frac{3}{4})^{m-2}}{14^{\frac{3}{2}}}
\end{aligned}
$$

Vastaavasti

$$
\sum_{j=0}^{m-3} \frac{1}{2^j} \sin j\theta \sin \omega = \frac{1 - (-\frac{\sqrt{7}}{4})^{m-2}}{4\sqrt{14}}.
$$

Hamming-tiheyden odotusarvo saadaan vajaasta Hamming -tiheydesta

$$
\begin{aligned}
\pi_m &= \frac{1}{2} - \frac{1}{\sqrt{14}(m-2)} \left(\frac{1 - (-\frac{3}{4})^{m-2}}{14^{\frac{3}{2}}} + \frac{1 - (-\frac{\sqrt{7}}{4})^{m-2}}{4\sqrt{14}} \right) \\
\pi &= \lim_{m \to \infty} \pi_m = \frac{1}{2} - 0 \cdot \left(\frac{1-0}{14^{\frac{3}{2}}} + \frac{1-0}{4\sqrt{14}} \right) = \frac{1}{2}.
\end{aligned}
$$

□

Algoritmi 4.9 voidaan helposti muokata käyttämään JSF-esitystä syötteenä. Tutkitaan tapauksia $w = 1$ ja $w = 2$. Tapauksessa $w = 1$ lisäysten ja kahdennusten määrien odotusarvot saadaan suoraan JSF-esityksen Hamming-tiheydestä.

$$
\left(t \cdot \frac{1}{2} \right) L \text{ ja } tK \tag{4.10}
$$

Tapaus $w = 2$ vaatii tarkempaa analyysia, koska nollasarakkeet eivät ole JSF-ominaisuuksista johtuen tasaisesti jakautuneita (vrt. [4, s. 11]). Määritellään nyt kolme tilaa algoritmin 4.9 ikkunalle (huomaa, että algoritmi liikkuu vasemmalta oikealla, kun taas JSF-määritys liikkuu oikealta vasemmalle).

$\mathbf{K_1}$ JSF-määritys on tilassa A. Tilan tuottama sarake on oikeanpuoleinen sarake ikkunassa, ja vasemmanpuoleinen sarake on tilan B tuottama.

$\mathbf{K_2}$ JSF-määritys on tilassa B. Tilan tuottama sarake on vasemmanpuoleinen sarake ikkunassa, ja oikean puoleisessa sarakkeessa on A tai C.

$\mathbf{K_3}$ JSF-määritys on tilassa C. Ikkuna siirtyy eteenpäin.

Tila $\mathbf{K_2}$ vastaa yhden ikkunan muodostamista, mikä taas vastaa yhtä lisäystä algoritmissa.

Lauseesta 4.7 tiedetään, että $Tod(\mathbf{K_3}) = 1/2$. Tämän lisäksi $\mathbf{K_1}$ esiintyy ainoastaan tilan $\mathbf{K_3}$ jälkeen todennäköisyydellä $1/4$, joten $Tod(\mathbf{K_3}) = 1/2 * 1/4 = 1/8$. Siis $Tod(\mathbf{K_2}) = 1 - 1/8 - 1/2 = 3/8$.

Täten tapauksen $w = 2$ lisäysten ja kahdennusten määrien odotusarvot ovat

$$\left(t \cdot \frac{3}{8}\right) L \text{ ja } tK. \tag{4.11}$$

Lisäksi tarvitsee esilaskea pisteet $l_0P_0 + l_1P_1$ ja $l_0P_0 - l_1P_1$ kaikilla nollasta poikkeavilla 2-kertaisilla yhdistelmillä $(l_0, l_1) \in \{1, 2, 3\}^2$, joissa l_0 tai l_1 on pariton, poislukien parit $(1, 3)$, $(3, 1)$ ja $(3, 3)$ määritelmän 4.6 kohdan (iii) perusteella. Esilaskea tarvitsee siis 10 pistettä, joista jokainen vaatii yhden yhteenlaskun.

5 Vertailua

Tässä kappaleessa vertaillaan edellä esitettyjen algoritmien soveltuvuutta ECDSA -allekirjoituksen varmistamisessa. Testeissä käytetään National Institute of Standards and Technology -laitoksen määrittelemiä käyriä P-192, P-224, P-256, P-384 ja P-521 (ks. [7, s. 87-90]). Esimerkiksi käyrä P-192 on määritelty seuraavasti.

Esimerkki 5.1. Merkintä P-192 vastaa käyrää $E : y^2 = x^3 + ax + b$ yli kunnan $\mathbb{Z}_q$, missä

$a = -3$

$b = 2455155546008943817740293915197451784769108058161191238065$

$q = 6277101735386680763835789423207666416083908700390324961279.$

Tutkitaan algoritmeja ensin edellisen kappaleen analyyseihin perustuen.

5.1 Yhteenlaskujen määrien odotusarvot

Kaikkien algoritmien kahdennusten määrän odotusarvot ovat lähellä toisiaan, joten on järkevää keskittyä tutkimaan lisäysten määrien odotusarvoja. Taulukossa 5.1 on määritelty odotetut lisaysten määrät sisältäen esilaskennat. Tässä t on avaimen pituus ja suluissa on tuloksen laskennassa käytetty ikkunan/ikkunoiden koko.

Algoritmi	$t = 192$	$t = 224$	$t = 256$	$t = 384$	$t = 521$
Alg 4.2	95 (4, 4)	108 (4, 4)	119 (5, 5)	162 (5, 5)	208 (5, 5)
Alg 4.6	**82** (4, 4)	**93** (4, 4)	**103** (5, 5)	**144** (5, 5)	**182** (5, 5)
Alg 4.8	104 (2)	119 (2)	134 (2)	188 (3)	233 (3)
Alg 4.9	92 (2)	106 (2)	120 (2)	161 (3)	202 (3)
Alg 4.9+JSF	**82** (2)	**94** (2)	106 (2)	154 (2)	205 (2)

Taulukko 5.1: Algoritmien yhteenlaskujen määrien odotusarvot. Pienimmät arvot lihavoitu.

Taulukosta 5.1 nähdään, että etumerkillistä numeroesitystä käyttämällä saavutetaan pienimmät yhteenlaskujen määrät. Isoimmilla avainten koilla algoritmi 4.9+JSF hidastuu muihin verrattuna, koska sen ikkunan koko on JSF-esityksen ominaisuuksista johtuen kiinnitetty. Tämän taulukon perusteella Algoritmi 4.6 vaikuttaa tehokkaimmalta.

5.2 Matlab-testit

Algoritmit toteutettiin Matlab-ohjelmointiympäristöä käyttäen. Testi alustana toimi Lenovo s205 ja 32-bittinen Matlab 7.10. Taulukossa on ilmoitettu sadan toiston keskiarvo sekunteina.

Algoritmi	$t = 192$	$t = 224$	$t = 256$	$t = 384$	$t = 521$
Algoritmi 4.2	$65,14$	$83,95$	$103,49$	$\mathbf{226,30}$	$\mathbf{406,142}$
Algoritmi 4.6	$\mathbf{64,00}$	$\mathbf{82,57}$	$\mathbf{101,93}$	$\mathbf{226,55}$	$\mathbf{406,35}$
Algoritmi 4.8	$69,21$	$89,25$	$109,97$	$239,81$	$425,90$
Algoritmi 4.9	$66,22$	$85,43$	$105,41$	$227,80$	$\mathbf{406,65}$
Algoritmi 4.9+JSF	$\mathbf{64,95}$	$84,17$	$104,34$	$233,83$	$424,50$

Taulukko 5.2: Matlab testien tulokset. Lyhyimmät ajat lihavoitu.

Taulukko 5.2 noudattelee pääpiirteittäin taulukon 5.1 tuloksia. Jälkimmäisen taulukon pienemmät erot seuraavat siitä, että suurin osa ajasta menee kahdennuksiin, joita ei ole huomitoitu taulukossa 5.1. Suurimpana erona samanaikaiset menetelmät näyttävät tomivan hieman odotettua hitaammin. Ero voi johtua hankalammasta esilaskentojen hallinnasta, sillä algoritmit toteutettiin tässä tapauksessa siten, että ne toimivat kaikilla ikkunoiden koilla. Poikkeuksena JSF-esitystä hyödyntävää Algoritmi. Myös etumerkkillisen

esityksen käyttö näyttäisi auttavan odotettua vähemmän, ja tämä korostuu erityisesti suurilla avainten koilla.

Viitteet

[1] D. Hankerson, A. Menezes & S. Vanstone *Guide to Elliptic Curve Cryptography*. New York: Springer-Verlag, 2004.

[2] B. Möller *Algorithms for multi-exponentiations*. Darmstadt: Technische Universität Darmstadt, 2001.

[3] J. Solinas *Low-Weight Binary Representations for Pairs of Intergers*. USA: National Security Agency, 2001.

[4] R. Avanzi *On the complexity of certain multi-exponentiation techniques in cryptography*. Essen: Institut für Experimentelle Mathematik, 2003.

[5] J. Muir *Efficient Integer Representations for Cryptographic Operations*. Waterloo: University of Waterloo, 2004.

[6] T. Takagi *luentomoniste*
http://www.cdc.informatik.tu-darmstadt.de/TI/Lehre/SS04/Vorlesung/EK/lecture8.pdf.

[7] Information Technology Laboratory *FIPS PUB 186-3* Gaithersburg: National Institute of Standards and Technology, 2009.