Tietojenkäsittelytiede **32** Heinäkuu 2011 sivut 34–47
Toimittaja: Ari Korhonen

Elliptisten käyrien salausmenetelmät ja niiden laskenta ohjelmoitavalla logiikalla

Kimmo Järvinen
Aalto-yliopisto
Tietojenkäsittelytieteen laitos
kimmo.jarvinen@tkk.fi

1 Johdanto

Tässä artikkelissa luodaan katsaus elliptisen käyrien salausmenetelmiin ja niiden laskentaan ohjelmoitavalla logiikalla. Kryptologiaa yleisemmin käsittelevä suomenkielinen artikkeli löytyy tämän lehden aiemmasta numerosta [12].

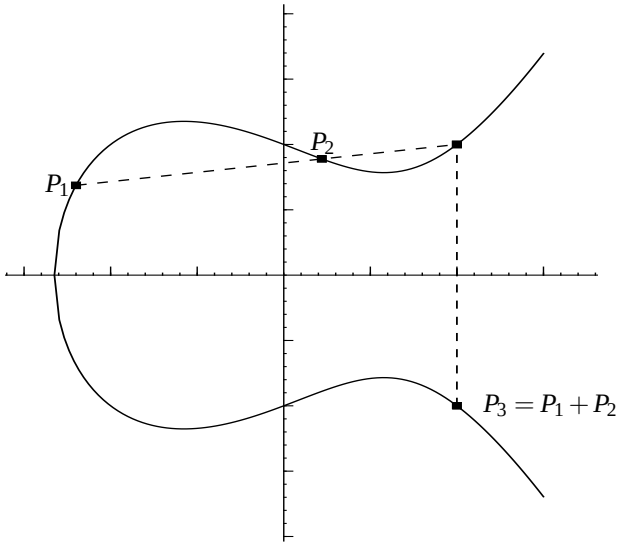
Salausmenetelmien tehokas toteuttaminen on kryptografian osa-alue, joka on herättänyt paljon mielenkiintoa sekä akateemisessa maailmassa että teollisuudessa. Aiheen parissa työskentelee varsin monenkaltaisen taustan omaavia ihmisiä perinteisistä kryptologeista tietotekniikka- ja elektroniikkainsinööreihin. Suuri osa tästä työstä on käsitellyt elliptisten käyrien salausmenetelmiä. Ne ovat julkisen avaimen salausjärjestelmiä, joilla tietty turvallisuustaso voidaan saavuttaa lyhyemmillä avaimen pituuksilla kuin esim. yleisesti tunnetussa RSA-salausjärjestelmässä. Elliptisten käyrien salausjärjestelmien laskentanopeus on myös yleensä RSA:ta suurempi.

Tehokkaan salausjärjestelmätoteutuksen suunnitteleminen on haastavaa, sillä se vaatii monissa sovelluksissa vaikeiden kompromissien tekemistä. Toisaalta vaaditaan suurta nopeutta ja toisaalta toteutuksen käytössä olevat resurssit ovat usein erittäin rajalliset (esim. tehonkulutuksen suhteen). Kolmanneksi toteutus itse ei saisi vaarantaa salausjärjestelmän turvallisuutta eli sen pitäisi pystyä vastustamaan ns. sivukanavahyökkäyksiä.

Artikkelin alussa kappaleissa 2 ja 3 esitellään yleisesti elliptisten käyrien salausmenetelmät ja niiden toteuttamisen kannalta keskeiset algoritmit havainnollisten esimerkkien avulla. Sivukanavahyökkäyksiin ja keinoihin niiltä suojautumiseksi tutustutaan kappaleessa 4. Kappale 5 käsittelee ohjelmoitavia logiikkapiirejä ja niiden käyttöä salausjärjestelmien toteutusalustoina. Lopuksi kappaleessa 6 esitellään pääpiirteittäin kirjoittajan tekemä toteutus ohjelmoitaville logiikkapiireille.

2 Elliptisten käyrien salausmenetelmät

Neal Koblitz ja Victor Miller toisistaan riippumatta ehdottivat elliptisten käyrien käyttöä julkisen avaimen salausjärjestelmissä vuonna 1985 [6, 10]. Elliptiset käyrät ovat tasokäyriä, jotka voidaan määritellä missä tahansa kunnassa K. Elliptisen käyrän, E, pisteet yhdessä niin kutsutun äärettömyydessä olevan pisteen, $\mathcal{O}$, kanssa muodostavat Abelin ryhmän, $E(K)$, yhteenlaskun suhteen. Voidaan siis määritellä yhteenlasku $P_3 = P_1 + P_2$, missä P_i:t ovat käyrän pisteitä eli $P_i \in E(K)$. Ääret-



Kuva 1: Pisteiden P_1 ja P_2 yhteenlasku $P_3 = P_1 + P_2$ elliptisellä käyrällä $y^2 = x^3 - x + 1$, missä $x, y \in \mathbb{R}$. (Kuva on Billy Brumleyn tekemä ja sitä käytetään tekijän luvalla.)

tömyydessä oleva piste on ryhmän nollaalkio eli $P = P + \mathcal{O}$ kaikille ryhmän pisteille P.

Pisteiden yhteenlasku on havainnollisinta esittää, jos K on reaaliluvut $\mathbb{R}$. Tällöin elliptinen käyrä voidaan kirjoittaa muodossa

$$y^2 = x^3 + ax + b, \tag{1}$$

missä $a, b \in \mathbb{R}$, ja $E(\mathbb{R})$ on yhtälön (1) toteuttavien pisteiden joukko sekä $\mathcal{O}$. Kahden pisteen $P_1 \neq \pm P_2$ kautta piirretty suora leikkaa aina kolmannen käyrän pisteen $P_3' = (x_3', y_3')$. Yhteenlaskun tulospiste saadaan peilaamalla tämä piste x-akselin suhteen: $P_3 = -P_3' = (x_3', -y_3')$. Kuvassa 1 on esitetty kahden pisteen yhteenlasku käyrällä $y^2 = x^3 - x + 1$. Jos $P_1 = P_2$, käytetään käyrää kyseisessä pisteessä sivuavaa tangenttia ja puhutaan pisteen kertomisesta kahdella $P_3 = 2P_1$. Jos pisteiden kautta kulkeva suora (tai tangentti) on y-akselin suuntainen, saadaan tulospisteeksi ääretömyydessä oleva piste $\mathcal{O}$.

Elliptisen käyrän pisteitä voidaan kertoa kokonaisluvuilla määrittelemällä kertolasku yhteenlaskun avulla seuraavasti:

$$Q = kP = \underbrace{P + P + \ldots + P}_{k \text{ kertaa}}, \tag{2}$$

missä P ja Q ovat käyrän pisteitä ja k on kokonaisluku. Tämä operaatio on keskeinen kaikissa elliptisten käyrien salausjärjestelmissä ja sen laskemiseksi on kehitetty useita tehokkaita algoritmeja, joihin palataan kappaleessa 3.

Ongelmaa, jossa pyritään selvittämään k, kun tiedetään P ja Q, kutsutaan elliptisen käyrän diskreetin logaritmin ongelmaksi. Elliptisten käyrien käyttö salausjärjestelmissä perustuu uskoon sen laskennallisesta vaativuudesta vastaavasti kuin RSA perustuu uskoon tekijöihin jaon vaativuudesta. Elliptisen käyrän diskreetti logaritmi vaikuttaa kuitenkin oleellisesti vaikeammalta ongelmalta kuin tekijöi-

hin jako, jos elliptinen käyrä on valittu oikein, ja siksi elliptisten käyrien salausmenetelmissä voidaankin käyttää huomattavasti lyhyempiä avaimia. Vaikutelmaa tukevat myös käytännön tulokset: tätä artikkelia kirjoitettaessa (18.10.2010) tekijöihin jako on onnistunut 768-bittisellä RSA moduluksella [5], mutta elliptisen käyrän diskreetti logaritmi on ratkaistu vain 112 bitillä. Yritys murtaa 131-bittinen elliptisen käyrän diskreetti logaritmi on tosin parhaillaan käynnissä. Nykyään suositellaan käytettävän vähintään 160-bittisiä elliptisiä käyriä.

Kryptografisissa sovelluksissa K on äärellinen kunta $\mathbb{F}_q$, jonka alkioiden määrä q on tyypillisesti joko alkuluku p tai kahden kokonaislukupotenssi 2^m. Ensimmäisessä tapauksessa puhutaan alkulukukunnista ($\mathbb{F}_p$) ja jälkimmäisessä binäärikunnista ($\mathbb{F}_{2^m}$).

3 Algoritmit

Elliptisten käyrien salausjärjestelmien vaatima laskenta noudattaa kuvassa 2 esitettyä hierarkiaa. Ylimmällä tasolla on itse salausjärjestelmä, joka käyttää seuraavalla tasolla olevaa kertolaskua elliptisellä käyrällä osana laskentaa. Yksi esimerkki elliptisen käyrän salausjärjestelmästä on elliptisen käyrän digitaalinen allekirjoitusalgoritmi (engl. elliptic curve digital signature algorithm, ECDSA) [11], jossa viestin allekirjoittaminen vaatii yhden ja allekirjoituksen oikeellisuuden varmistaminen kaksi kertolaskua elliptisellä käyrällä. Lisäksi tarvitaan kokonaislukuaritmetiikkaa ja viestistä laskettu yksisuuntainen tiiviste (engl. hash), mutta jatkossa keskitymme ainoastaan elliptisen käyrän operaatioihin, jotka muodostavat laskennallisesti merkittävimmän osan elliptisen käyrän salausjärjestelmistä. Seuraavassa kuvan 2 hierarkia käydään läpi alhaalta ylös kunnan operaatioista kertolaskuun elliptisellä käyrällä.

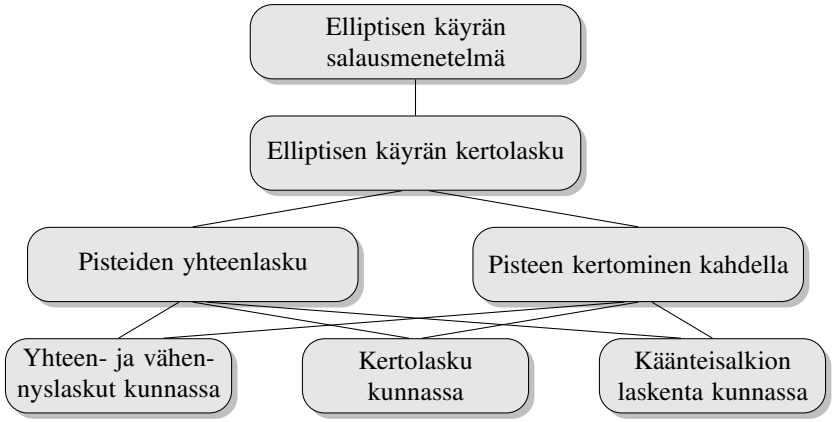
3.1 Kunnan operaatiot

Hierarkian alimmalla tasolla olevat kunnan operaatiot viime kädessä määrittelevät elliptisen käyrän salausjärjestelmätoteutuksen tehokkuuden. Kuten mainittua, käytännössä käytetään yleensä alkulukutai binäärikuntia. Nyrkkisääntönä voidaan sanoa, että alkulukukunnat soveltuvat paremmin ohjelmistotototeutuksille, koska tällöin voidaan hyödyntää prosessorien kokonaislukuaritmetiikkaa, ja binäärikunnat laitteistotototeutuksille, koska tällöin aritmetiikka voidaan toteuttaa ilman muistibittejä (engl. carry). Tilanne saattaa kuitenkin olla muuttumassa, koska prosessorivalmistajat ovat sisällyttäneet uusiin prosessoreihinsa[1] käskyjä binäärikunnan operaatioille, joten myös binäärikunnat ovat toteutettavissa helposti ja tehokkaasti näillä prosessoreilla.

Alkulukukunnassa $\mathbb{F}_p$ operaatiot lasketaan modulo p. Kahden alkion $a,b \in \mathbb{F}_p$ yhteenlasku on siis $a+b \bmod p$, vähennyslasku $a-b \bmod p$ ja kertolasku $ab \bmod p$. Käytännön sovelluksissa pyritään siihen, että modulo p on helposti laskettavissa. Tämä voidaan saavuttaa mm. valitsemalla p sopivasti; esim. valitaan p, joka on Mersennen alkuluku eli muotoa $p = 2^n - 1$, jolloin modulo on laskettavissa vain yhdellä yhteenlaskulla. Elliptisen käyrän salausjärjestelmissä käytettävistä kunnan operaatioista laskennallisesti vaativin on käänteisalkion laskenta eli ongelma, jossa alkiolle a etsitään alkio a^{-1}, jolle pätee $aa^{-1} \equiv 1 \pmod{p}$. Tavallisesti tämä suoritetaan laajennetulla Euclidin algoritmilla tai sen muunnelmalla.

Esimerkki 1 *Kokonaisluvut* $a = 35$ *ja* $b = 98$ *ovat kunnan* $\mathbb{F}_{127}$ *alkioita. Yhteen- ja vähennyslaskut antavat* $(35+98) \bmod$

[1] mm. Intel Core prosessorit vuodesta 2010



Kuva 2: Elliptisten käyrien salausjärjestelmien hierarkia

$127 = 6$ *ja* $(35 - 98) \bmod 127 = 64$. *Alkioiden tulo on* $35 \cdot 98 = 3430 \equiv 1 \pmod{127}$. *Toisin sanoen b on a:n käänteisalkio kertolaskun suhteen:* $b = a^{-1}$. *Koska p on Mersennen alkuluku (*$2^7 - 1$*), modulo voidaan laskea helposti yhdellä yhteenlaskulla. Koska* $3430 = 26 \cdot 2^7 + 102$, *saadaan* $26 + 102 = 128 \equiv 1 \pmod{127}$.

Binäärikunnassa $\mathbb{F}_{2^m}$ alkiot esitetään binääripolynomeina: $a(x) = \sum_{i=0}^{m-1} a_i x^i$, missä $a_i = \{0,1\}$. Operaatiot lasketaan modulo 'm:n asteen jaoton polynomi $p(x)$.' Alkioiden yhteenlasku lasketaan bittikohtaisesti loogisella ehdotontai-operaatiolla (engl. exclusive-or, xor) ja siten se ei vaadi muistibittejä eikä modulo $p(x)$ -laskua. Kertolasku lasketaan polynomien kertolaskulla, jonka tulos redusoidaan modulo $p(x)$. Myös binäärikunnilla $p(x)$ pyritään valitsemaan siten, että modulo $p(x)$ on helppo laskea.

Esimerkki 2 *Polynomit* $a(x) = x^2 + x = \langle 0110 \rangle$ *ja* $b(x) = x^3 + x + 1 = \langle 1011 \rangle$ *ovat kunnan* $\mathbb{F}_{2^4}$ *alkioita. Jaoton polynomi olkoon* $p(x) = x^4 + x + 1$. *Alkioiden yhteenlasku lasketaan seuraavasti:* $a(x) + b(x) = (x^2 + x) + (x^3 + x + 1) = x^3 + x^2 + 1$, *mikä vastaa alkioita kuvaavien bittivektorien ehdoton-tai-operaatiota:* $\langle 0110 \rangle \oplus \langle 1011 \rangle = \langle 1101 \rangle$. *Alkioiden kertolasku lasketaan kahdessa vaiheessa: Polynomien kertolaskulla saadaan* $a(x)b(x) = x^5 + x^4 + x^3 + x$ *ja tulos lasketaan* $(x^5 + x^4 + x^3 + x) \bmod (x^4 + x + 1)$. *Polynomien jakolaskulla saadaan* $x^5 + x^4 + x^3 + x = (x+1)(x^4 + x + 1) + x^3 + x^2 + x + 1$ *eli alkioiden kertolaskun tulos on* $x^3 + x^2 + x + 1 = \langle 1111 \rangle$.

Käytännön järjestelmissä käytetään luonnollisesti huomattavasti suurempia kuntia kuin esimerkeissä 1 ja 2. Kappaleessa 2 mainitusta suosituksesta käyttää vähintään 160-bittisiä käyriä seuraa, että kunnat ovat kokoa $\log_2(p) \approx 160$ ja $m \approx 160$.

3.2 Pisteoperaatiot

Kuten jo kappaleessa 2 kävi ilmi, kertolaskun laskemisessa on kaksi keskeistä operaatiota: pisteen kertominen kahdella $P_3 = 2P_1$ ja pisteiden yhteenlasku $P_3 = P_1 + P_2$. Tässä kappaleessa esitetään yksinkertaisimmat algoritmit näille operaatiolle. Lisäksi käsitellään käytännön järjestelmissä sovellettujen algoritmien toteutusperi-

aatteet kuitenkaan menemättä niiden yksityiskohtiin. Yksityiskohdista kiinnostuneiden lukijoiden tulisi etsiä tietoa esim. aihetta käsittelevistä kirjoista [2].

Olkoon elliptinen käyrä muotoa

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

missä $a, b \in \mathbb{F}_p$. Tällöin $P_3 = (x_3, y_3) = P_1 + P_2 = (x_1, y_1) + (x_2, y_2)$ lasketaan seuraavasti:

$$x_3 = \lambda^2 - x_1 - x_2 \quad \text{ja} \tag{4}$$
$$y_3 = \lambda(x_1 - x_3) - y_1, \tag{5}$$

missä

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1} \quad \text{jos } P_1 \neq \pm P_2 \tag{6}$$

tai

$$\lambda = \frac{3x_1^2 + a}{2y_1} \quad \text{jos } P_1 = P_2. \tag{7}$$

Esimerkki 3 *Pisteet* $(5,4)$ *ja* $(22,19)$ *ovat elliptisellä käyrällä* $y^2 = x^3 + 2x + 19$ *äärellisen kunnan ollessa* $\mathbb{F}_{23}$. *Pisteiden yhteenlasku aloitetaan laskemalla* λ *kaavalla* (6)*:*

$$\lambda = \frac{19 - 4}{22 - 5} = \frac{15}{17} = 15 \cdot 19 = 9,$$

sillä $17^{-1} = 19$ *(eli* $17 \cdot 19 \equiv 1 \pmod{23}$*). Sijoittamalla* λ *kaavoihin* (4) *ja* (5) *saadaan*

$$x_3 = 9^2 - 5 - 22 = 8 \quad ja$$
$$y_3 = 9(5 - 8) - 4 = 15.$$

Eli $(5,4) + (22,19) = (8,15)$. *Sijoittamalla* $(8,15)$ *yhtälöön* (3) *voidaan todeta, että myös yhteenlaskun tulos on käyrällä.*

Kaavoista (6) ja (7) nähdään, että sekä pisteiden yhteenlasku että pisteen kertominen kahdella vaativat käänteisalkion laskennan äärellisessä kunnassa. Kuten kappaleessa 3.1 mainittiin, se on laskennallisesti huomattavasti vaativampi kuin yhteen-, vähennys- tai kertolaskut.

Pisteiden yhteenlasku ja pisteen kertominen kahdella voidaan laskea ilman käänteisalkion laskentaa, jos käytetään perinteisten (x,y)-tasokoordinaattien sijasta projektiivisia koordinaateja, joissa piste esitetään kolmella koordinaatilla: $P = (X,Y,Z)$. Tällöin pisteoperaatioissa tarvitaan enemmän muita kunnan operaatioita, mutta yleensä projektiviisilla koordinaateilla saadaan huomattavia nopeutuksia. Kertolaskun tulospiste on laskennan lopuksi siirrettävä tasokoordinaatteihin, mikä vaatii yhden käänteisalkion laskennan, esim. $(x,y) = (X/Z, Y/Z^2)$ [1].

3.3 Kertolasku

Algoritmit kertolaskun laskemiseksi elliptisellä käyrällä ovat samankaltaisia eksponentointialgoritmien kanssa: kertolaskujen sijasta lasketaan pisteiden yhteenlasku ja neliöintien sijasta piste kerrotaan kahdella.

Yksinkertaisin algoritmi on ns. binäärialgoritmi, jossa kokonaisluvun k binääriesitys $k = \sum_{i=0}^{\ell-1} k_i 2^i = \langle k_{\ell-1} k_{\ell-2} \ldots k_1 k_0 \rangle$, missä $k_i \in \{0,1\}$, käydään järjestyksessä läpi bitti kerrallaan kummasta tahansa päästä aloittaen. Intuitiivisempi "oikealta vasemmalle" (vähiten merkitsevästä bitistä eniten merkitsevään) -versio algoritmista esitetään kuvassa 3(a) ja "vasemmalta oikealle"-versio kuvassa 3(b). Molempien versioiden tapauksessa pitää laskea $\ell - 1$ pisteen kertomista kahdella ja $h(k) - 1$ pisteiden yhteenlaskua, missä $h(k)$ on nollasta poikkeavien k_i:den lukumäärä. Keskimäärin tarvitaan siis n. $\ell/2$ pisteiden yhteenlaskua.

Esimerkki 4 *Lasketaan* $Q = 1145P$ *binäärialgoritmilla oikealta vasemmalle. Koska* $1145 = \langle 10001111001 \rangle$*, lasketaan kertolasku seuraavasti:* $P + 2^3 P + 2^4 P +$

Input: Kokonaisluku $k = \sum_{i=0}^{\ell-1} k_i 2^i$, missä $k_i = \{0,1\}$, ja käyrän piste P
Output: Käyrän piste $Q = kP$
$Q \leftarrow \mathcal{O}$
for $i = 0$ **to** $\ell - 1$ **do**
  if $k_i = 1$ **then**
    $Q \leftarrow Q + P$
  end if
  $P \leftarrow 2P$
end for

(a)

Input: Kokonaisluku $k = \sum_{i=0}^{\ell-1} k_i 2^i$, missä $k_i = \{0,1\}$, ja käyrän piste P
Output: Käyrän piste $Q = kP$
$Q \leftarrow \mathcal{O}$
for $i = \ell - 1$ **to** 0 **do**
  $Q \leftarrow 2Q$
  if $k_i = 1$ **then**
    $Q \leftarrow Q + P$
  end if
end for

(b)

Kuva 3: (a) oikealta vasemmalle ja (b) vasemmalta oikealle binäärialgoritmit kertolaskulle elliptisellä käyrällä

$2^5P + 2^6P + 2^{10}P = 1145P$. *Tarvitaan siis kymmenen pisteen kertomista kahdella ja viisi pisteiden yhteenlaskua.*

Elliptisellä käyrällä pisteiden yhteenlasku ja sen käänteisoperaatio eli pisteiden vähennyslasku ovat käytännössä sama operaatio: $P_1 - P_2 = P_1 + (-P_2)$; pisteen etumerkin vaihtaminen on erittäin edullista (ks. kappale 2). Näin ollen binäärialgoritmissa laskettavien pisteiden yhteenlaskujen määrää voidaan vähentää ottamalla pisteiden vähennyslaskut käyttöön. Tällöin k esitetään etumerkillisellä binääriesityksellä, jossa $k_i = \{-1,0,1\}$. Vast'edes merkitsemme $\overline{1} = -1$.

Etumerkillisellä binääriesityksellä voidaan pienentää pisteiden yhteenlaskujen määrä korvaamalla peräkkäisiä ykkösiä k:n binääriesityksessä, esim. $15 = \langle 1111 \rangle$ korvataan $\langle 1000\overline{1} \rangle = 16 - 1 = 15$. Vaikka on selvää, että yleisesti etumerkillinen binääriesitys ei ole yksikäsitteinen, käytännön järjestelmissä käytetään yksikäsitteistä ns. ei-vierekkäistä muotoa (engl. non-adjacent form, NAF). NAF:ssa kaksi vierekkäistä etumerkillistä bittiä eivät milloinkaan ole molemmat nollasta poikkeavia eli $k_i k_{i+1} = 0$ kaikilla i:n arvoilla. NAF on korkeintaan yhden bitin pidempi kuin tavallinen binääriesitys ja $h(k) \approx \ell/3$ eli kestolaskussa tarvitaan keskimäärin vain $\ell/3$ pisteiden yhteen- tai vähennyslaskua.

On huomionarvoista, että kokonaislukujen eksponentointia käyttävissä julkisen avaimen salausjärjestelmissä (esim. Diffie–Hellman ja RSA) etumerkillisen binääriesityksen käyttäminen ei ole yhtä suoraviivaista, koska jakolasku on huomattavasti kertolaskua monimutkaisempi.

Esimerkki 5 *Lasketaan $Q = 1145P$ binäärialgoritmilla, jossa myös vähennyslaskut ovat sallittuja. Saadaan seuraava etumerkillinen binääriesitys (NAF):* $1145 = \langle 1001000\overline{1}001 \rangle$. *Kertolasku lasketaan näin ollen seuraavasti:* $P - 2^3P + 2^7P + 2^{10}P = 1145P$. *Laskennan kustannus on kymmenen pisteen kertomista kahdella, kaksi pisteiden yhteenlaskua ja yksi vähennyslasku. Säästetään siis kaksi yhteenlaskua, kun pisteiden vähennyslasku otetaan käyttöön.*

Seuraava askel pisteiden yhteenlaskujen vähentämisessä on suorittaa esilaskentaa pisteellä P ja hyödyntää siinä saatuja pisteitä kertolaskussa. Kirjallisuudessa esitetään useita eri esilaskentaa hyödyntäviä algoritmeja, joista seuraavassa esitetään w-levyistä NAF:ia (engl. width-w non-adjacent form, w-NAF) käyttävä al-

goritmi. Tässä esityksessä k_i:lle sallitaan arvot $[-2^{w-1}, 2^{w-1}-1]$. Esityksen pituus on korkeintaan yhden tavallista binääriesitystä pidempi ja sille pätee $h(k) \approx \ell/(w+1)$. Edellä esitetty tavallinen NAF on w-NAF arvolla $w = 2$. Esilaskennassa lasketaan ja tallennetaan $P_i = iP$, missä $i \in \{1, 3, 5, \ldots, 2^{w-1}-1\}$. Tämän jälkeen kertolasku elliptisellä käyrällä suoritetaan binäärialgoritmin tapaan. Esilaskentoja hyödyntävissä algoritmeissa laskenta suoritetaan vasemmalta oikealle (eniten merkitsevästä vähiten merkitsevään bittiin), koska oikealta vasemmalle algoritmeissa tulisi kaikki esilasketut pisteet kertoa kahdella jokaisella i:n arvolla.

Esimerkki 6 *Lasketaan $Q = 1145P$ käyttäen esilaskentaa ja 4-NAF:ia. Aluksi esilasketaan $P_1 = P$, $P_3 = 3P$, $P_5 = 5P$ ja $P_7 = 7P$. Kokonaisluvun k 4-NAF on $\langle 100070000007\rangle$ ja siten vasemmalta oikealle binäärialgoritmia käyttäen saadaan $2^7(2^4P_1 - P_7) - P_7 = 128(16P - 7P) - 7P = 1145P$. Laskenta vaatii siis yksitoista pisteen kertomista kahdella ja vain kaksi pisteiden vähennyslaskua, jos esilaskennan kustannusta ei oteta huomioon. Esilaskenta vaatii yhden pisteen kertomisen kahdella ja kolme pisteiden yhteenlaskua.*

Vaikka yllä olevassa esimerkissä ei esilaskennalla saada vähennettyä operaatioiden kokonaismäärä esilaskennan kustannuksen mitätöidessä saadut hyödyt, oikeissa järjestelmissä käytetyillä k:n arvoilla hyödyt ovat selkeät: Esimerkiksi jos $\ell = 160$, NAF:n $h(k) \approx 160/3 = 53,333\ldots$ ja 4-NAF:n $h(k) \approx 160/5 = 32$. Toisin sanoen keskimäärin säästetään yli 20 pisteiden yhteen- tai vähennyslaskua, joiden rinnalla esilaskennan kustannus menettää merkityksensä.

Yllä esitetyt algoritmit ovat käyttökelpoisia, vaikka sekä k että P vaihtuisivat jokaisella kerralla. Monissa käytännön järjestelmissä toinen näistä arvoista (yleensä P) on vakio tai vaihtuu harvoin. Tällöin saattaa olla mielekästä suorittaa aluksi huomattava määrä esilaskentaa, koska riittää, että esilaskenta suoritetaan vain kerran ennen varsinaisen laskennan aloittamista. Kaikilla yllä esitetyillä algoritmeilla vaaditaan $\ell - 1$ pisteiden kertomista kahdella, mutta P:n ollessa vakio niiden lukumäärä voidaan vähentää jopa nollaan (esilaskemalla jokainen 2^iP).

Pisteiden kertomiset kahdella voi välttää myös käyttämällä erityisiä Koblitzin käyriä, jolloin ne voidaan korvata erittäin yksinkertaisilla Frobeniuksen kuvauksilla [7]. Kappaleen 6 esimerkkitoteutus käyttää Koblitzin käyriä, joten niihin palataan lyhyesti myöhemmin.

4 Sivukanavahyökkäykset ja niiltä suojautuminen

Perinteisesti kryptoanalyysi on käsittellyt salauslaitteita “mustina laatikoina,” joiden sisäisestä toiminnasta ei saada tietoa. Käsitys mustista laatikoista murtui viimeistään 1990-luvun lopulla, kun Paul Kocher ja muut esittelivät ns. sivukanavahyökkäykset. He näyttivät, kuinka useista laajasti käytössä olleista salauslaitteista saattoi halvalla laitteistolla selvittää salaisen avaimen hyödyntämällä laitteesta sen käytön aikana mitattua tietoa, kuten suoritusaikaa [8] tai tehonkulutusta [9]. Muita mahdollisia mitattavia arvoja ovat mm. elektromagneettinen säteily tai jopa ääni. Seuraavassa keskitymme tehonkulutukseen perustuviin hyökkäyksiin, joista on eniten kirjallisuutta.

Eräs havainnollisimmista sivukanavahyökkäyksistä on yksinkertainen tehoanalyysi (engl. simple power analysis, SPA) [9] sovellettuna edellä esitettyyn elliptisen käyrän kertolaskuun binäärialgoritmilla. Algoritmissa piste kerrotaan kah-

della jokaisella kokonaisluvun bitillä k_i, mutta yhteenlasku suoritetaan vain, jos $k_i = 1$. Näissä kahdessa operaatiossa vaadittavat äärellisen kunnan operaatiot poikkeavat toisistaan (ks. kaavat (4)–(7)) ja siten myös operaatioiden tehonkulutuskäyrät ovat erilaiset, mikäli suunnittelija ei ole ottanut sivukanavahyökkäyksiä huomioon. Hyökkääjä voi siis selvittää kertolaskussa käytetyn kokonaisluvun k kytkemällä laitteen oskiloskooppiin ja lukemalla k:n mitatusta virrankulutuskäyrästä.

Esimerkki 7 *Hyökkääjällä on käsissään älykortti, joka suorittaa elliptisen käyrän kertolaskun salaisella k:n arvolla, jonka hyökkääjä haluaa saada selville. Hän tietää, että älykortti laskee kertolaskun binäärialgoritmilla oikealta vasemmalle (kuva 3(a)). Hyökkääjä mittaa laitteen virrankulutuksen kertolaskun laskemisen aikana. Mitatusta käyrästä erottaa selkeästi, että se koostuu kahdenlaisista kuvioista (A ja B), jotka toistuvat seuraavasti: AABAAABABAAABA…ABABAAABA. Selvästi kuvio A vastaa pisteen kertomista kahdella ja kuvio B pisteiden yhteenlaskua. Näin ollen hyökkääjä näkee, että* $k = \langle 100110 \ldots 1001100100 \rangle$ *(BA-pari vastaa bittiä* 1 *ja pelkkä A bittiä* 0*).*

SPA- ja muilta vastaavilta hyökkäyksiltä suojatumiseksi riittävät yleensä yksinkertaiset suojaukset, joilla varmistetaan, että tehonkulutukset ovat päällisinpuolin samanlaiset. Toisin sanoen varmistetaan, että algoritmin laskennassa käytetään aina samanlaista operaatioiden sarjaa riippumatta salaisen avaimen arvosta. Tämä saadaan aikaiseksi mm. laskemalla harhautusoperaatioita (engl. dummy operations). Esimerkiksi binäärialgoritmissa yhteenlasku lasketaan kaikille biteille, mutta sen tulosta käytetään vain, jos $k_i = 1$. Esimerkin 7 tapauksessa mitattu käyrä olisi nyt muotoa *BABABA*…*BABA*, eikä hyökkääjä enää voisi päätellä k:ta sen avulla.

Valitettavasti hyökkääjän keinovalikoima ei rajoitu ainoastaan SPA-hyökkäysten kaltaisiin yksinkertaisiin hyökkäyksiin. Differentiaalitehoanalyysi (engl. differential power analysis, DPA) [9] hyödyntää tilastollisia menetelmiä useaan tehonkulutuskäyrään, joissa on käytetty samaa avainta, ja kykenee hyödyntämään pieniäkin variaatioita operaatioiden tehonkulutuksissa, kunhan tehonkulutuskäyriä on riittävä määrä. DPA pystyy siten kiertämään esimerkiksi harhautusoperaatioita käyttävät suojausmenetelmät.

Sivukanavahyökkäykset eivät välttämättä vaadi edes fyysistä pääsyä laitteeseen, sillä esimerkiksi välimuistihyökkäysten tapauksessa riittää, että hyökkääjä kykenee asentamaan järjestelmään vakoojaohjelman, joka mittaa lukuaikoja prosessorin välimuistista. Näitä aikoja tulkitsemalla hyökkääjä voi arvioida, mitä muuttujia samaa välimuistia käyttävä salausprosessi on käyttänyt ja siten päätellä k:n arvot.

Hyökkääjän mahdollisuudet eivät myöskään rajoitu ainoastaan passiivisiin laitteen toiminnan mittauksiin, vaan hän voi monessa tapauksessa aktiivisesti vaikuttaa laitteen toimintaan. Virhehyökkäykset aiheuttavat tarkoituksellisia lyhytkestoisia virheitä laitteen toimintaan (esim. hetkellisellä käyttöjännitteen pudottamisella) ja laitteen reagointia tulkitsemalla saavat tietoa salaisesta avaimesta. Tällä tavoin voidaan murtaa esimerkiksi harhautusoperaatioita käyttävä suojaus: jos hetkellisellä virheellä ei ole vaikutusta lopputulokseen, on virhe tapahtunut harhautusoperaation aikana.

Kuten edellisestä on käynyt ilmi, sivukanavahyökkäykset ovat erittäin tehokkaita hyökkäyksiä, jotka muodostavat vakavan uhan monille käytännön salausjär-

jestelmille. Suunnittelijan keinovalikoima hyökkäyksiä vastaan suojautumiseksi on kuitenkin kehittynyt samalla, kun uusia hyökkäyksiä on keksitty. Suojakeinot voidaan jakaa karkeasti kahteen kategoriaan: algoritmisiin suojakeinoihin ja laitteistosuojakeinoihin.

Algoritmiset suojakeinot pyrkivät muuttamaan algoritmia siten, että sen laskenta ei vuoda tietoa. SPA-tyyppisiltä hyökkäyksiltä suojautumiseksi riittää, että algoritmia muutetaan siten, että laskennassa käytetyt operaatiot ovat kaikilla k:n arvoilla samat. DPA-hyökkäyksiä voidaan vaikeuttaa elliptisten käyrien salausjärjestelmien tapauksessa esimerkiksi satunnaistamalla P jokaisella laskentakerralla.

Laitteistosuojakeinot pyrkivät toteuttamaan laitteiston siten, että sen mitattavat ominaisuudet eivät riipu laskettavista arvoista. Esimerkiksi logiikkaportit suunnitellaan siten, että niiden tehonkulutus ei riipu sisääntuloarvoista. Täydellisen riippumattomuuden saavuttaminen on kuitenkin mahdotonta, sillä jo tuotantotekniikasta johtuva hienoinen variaatio transistorien ominaisuuksissa saattaa riittää paljastamaan avaimen, jos näytteitä on riittävästi. Käytännössä suunnittelijan tehtävä onkin pyrkiä takaamaan, että toteutus on riittävän hyvä vastustamaan realistisina pidetyt sivukanavahyökkäykset.

Sivukanavahyökkäyksiltä suojautuminen vaatii yleensä ylimääräisiä resursseja, kasvattaa tehonkulutusta ja hidastaa järjestelmän toimintaa. Tämän lisäksi sovelluksen on kyettävä vastustamaan monia erilaisia sivukanavahyökkäyksiä, mikä vaatii monien erilaisten suojakeinojen toteuttamista kasvattaen siten kustannuksia edelleen. Tämän vuoksi sivukanavahyökkäyksiltä suojautuminen vaikeuttaa huomattavasti tehokkaan salausjärjestelmätoteutuksen suunnittelemista, minkä vuoksi edullisten mutta tehokkaiden suojakeinojen löytäminen on tärkeää.

Luonnollisesti kaikki edellä esitetyt hyökkäykset eivät ole realistisia kaikkia salausjärjestelmien sovelluksia vastaan. On esimerkiksi epärealistista olettaa, että hyökkääjä pystyy mittaamaan nettipankkien salausjärjestelmiä laskevien laitteiden tehonkulutuskäyriä jo sen vuoksi, että fyysinen pääsy laitteelle on erittäin vaikeaa. Valitettavasti sovellukset, jotka ovat kaikkein alttiimpia sivukanavahyökkäyksille ovat myös hankalimpia suojattavia rajoitettujen resurssien vuoksi. Esimerkiksi hyökkääjän on helppo saada käsiinsä salausjärjestelmiä käyttävä älykortti ja sen tehonkulutus on helposti mitattavissa. Toisaalta älykortteissa suojakeinoihin käytettävät resurssit ovat rajalliset, mikä vaikeuttaa hyvän salausjärjestelmätoteutuksen suunnittelemista. Jokaisen salausjärjestelmiä käytännössä toteuttavan on huomioitava sivukanavahyökkäyksien muodostama uhka ja arvioitava, mitä suojauksia kyseinen sovellus vaatii.

5 Ohjelmoitava logiikka

Aiemmin kun kryptografiaa käytettiin enimmäkseen sotilas- ja viranomaissovelluksissa, salausjärjestelmät toteutettiin pääosin niitä varten suunnitelluilla integroiduilla piireillä. Nykyään suurin osa salausjärjestelmistä on ohjelmistototeutuksia, jotka suoritetaan yleiskäyttöisillä mikroprosessoreilla, jotka mahdollistavat edulliset ja helposti päivittävät toteutukset. Integroituja piirejä käytetään pääosin sovelluksissa, joissa on tiukkoja reunaehtoja joko käytettävissä olevien resurssien (esim. tehonkulutuksen) tai laskentanopeuden suhteen. Ohjelmoitava logiikka tarjoaa monissa sovelluksissa kolmannen vaihtoehdon, joka yhdistää monia integroitujen piirien ja mikroprosessoripohjaisten toteutusten hyvistä puolista. Ohjelmoitavalla logiikalla voidaan mm. saavut-

taa suuria laskentanopeuksia ilman, että pitää valmistaa erillistä integroitua piiriä.

Ohjelmoitavat logiikkapiirit ovat nimensä mukaisesti piirejä, joiden toiminta voidaan ohjelmoida logiikkatasolla. Tässä artikkelissa ohjelmoitavista logiikkapiireistä käsitellään ainoastaan FPGA-piirejä (field-programmable gate array), jotka ovat nykyään pääasiallisesti käytetty ohjelmoitavien logiikkapiirien luokka. Perinteisesti FPGA-piirejä käytettiin tuotekehitysvaiheessa esimerkiksi toteutuksen testaukseen ennen varsinaisen integroidun piirin valmistamista. FPGA-piirien koko ja tehokkuus ovat kuitenkin kasvaneet merkittävästi, mikä on mahdollistanut niiden käytön monissa lopputuotteissa. Itse asiassa FPGA:t ovat jo pitkään useissa sovelluksissa kokonaan korvanneet erillisten integroitujen piirien valmistamisen.

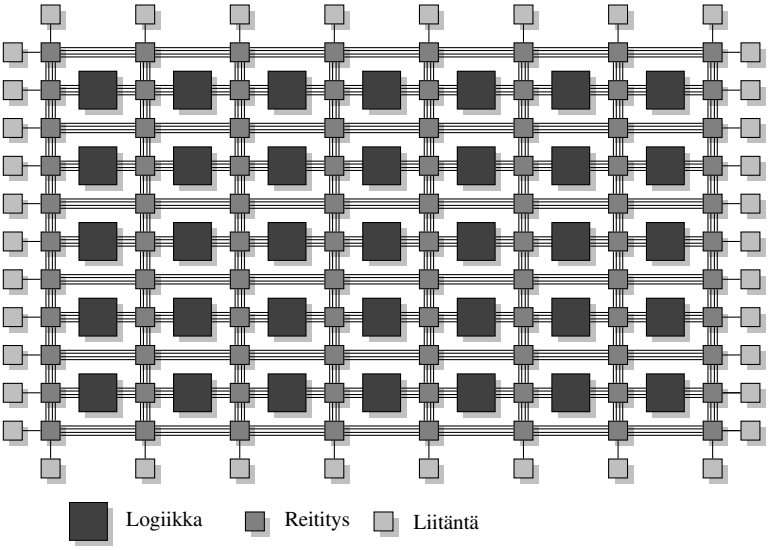
FPGA-piirien perusarkkitehtuuri rakentuu kolmenlaisista ohjelmoitavista komponenteista: logiikka-, reititys- ja liitäntälohkoista. Ohjelmoitavien logiikkalohkojen toiminta perustuu ohjelmoitavien hakutaulujen (look-up table, LUT) käyttöön. Toteutettavan logiikkafunktion totuustaulu ohjelmoidaan LUT:iin, jolloin sisääntuloarvoja vastaava logiikkafunktion arvo voidaan lukea suoraan LUT:sta. Tyypillisesti FPGA-piireissä käytettävät LUT:it ovat kooltaan 16–64 bittisiä (4–6 bittiä sisään ja yksi bitti ulos). FPGA-piireissä ohjelmoitavat logiikkalohkot on sijoitettu matriisimuodostelmaan ja ne on kytketty toisiinsa ohjelmoitavalla reitityksellä. Ohjelmoitavat liitäntälohkot tarjoavat kytkennät piirin ulkopuolelle. Yllä kuvattu FPGA-piirien perusarkkitehtuuri on esitetty kuvassa 4.

Esimerkki 8 *Lasketaan seitsemän bitin* $\langle a_6 a_5 \ldots a_0 \rangle$ *pariteetti eli seitsemän bitin ehdoton-tai (xor) FPGA-piirillä, joka rakentuu 16-bittisistä LUT:eista (neljä bittiä sisään, yksi ulos). Laskenta vaatii kaksi LUT:ia, jotka molemmat ohjelmoidaan laskemaan neljän bitin xor eli niihin ohjelmoidaan seuraava bittivektori (xor-operaation totuustaulu):* $\langle 0110100110010110 \rangle$. *Tämän jälkeen ensimmäisen LUT:n sisäänmenoarvoiksi reititetään esim.* a_6–a_3 *ja sen ulostulona saadaan välitulos* t. *Pariteetti saadaan toisen LUT:n ulostulosta, kun sen sisäänmenoon reititetään* t *ja* a_2–a_0.

Nykyaikaisissa FPGA-piireissä LUT:ien määrä on tyypillisesti tuhansista satoihin tuhansiin. Tämän lisäksi piireille on sijoitettu logiikkalohkojen lisäksi muistilohkoja, kertojia ja muita tiettyjä yleisesti käytettyjä toimintoja varten suunniteltuja komponentteja, jotka merkittävästi parantavat toteutusten tehokkuutta näitä toimintoja käyttävissä sovelluksissa. Joissakin tapauksissa FPGA-piirillä on jopa prosessoriytimiä (esim. PowerPC-ytimet Xilinx Virtex-5 FPGA-piireissä).

FPGA-piirejä käytettäessä suunnittelijan ei välttämättä tarvitse välittää alla olevasta arkkitehtuurista, sillä työkalut syntetisoivat, sijoittelevat ja reitittävät ohjelmakoodin määrittelemät toiminnallisuudet automaattisesti piirin resursseille. Alla olevan rakenteen huomioiminen saattaa toki tuoda merkittäviä tehokkuusparannuksia, joten sen ymmärtäminen ei ole merkityksetöntä. FPGA-piirien toteutukset ohjelmoidaan pääosin matalan tason laitteistokuvauskielillä esim. VHDL:llä, mutta myös korkeamman tason kieliä on olemassa esim. Handel-C (C-kielen laajennus).

FPGA-piirit ovat suosittuja toteutusalustoja salausjärjestelmille, koska niiden bittitason ohjelmoitavuus tarjoaa monia etuja sekä mikroprosessoreihin että sovelluskohtaisiin integroituihin piireihin verrattuna [14]. Ohjelmistototeutuksiin verrattuna FPGA-toteutukset ovat nopeita ja



Kuva 4: FPGA-piirien arkkitehtuuri

kuluttavat vähän tehoa. Sovelluskohtaisiin integroituihin piireihin verrattuna ne ovat edullisia (jos tuotantomäärät ovat pienehköjä) ja ohjelmoitavuuden vuoksi helposti päivitettäviä. FPGA-piireillä toteutus voidaan myös optimoida tarkasti tietyille parametreille, koska parametrit voidaan vaihtaa ohjelmoimalla FPGA uudelleen. Esimerkiksi äärellisen kunnan kertojat voidaan suunnitella vain tietylle alkuluvulle p tai jaottomalle polynomille $p(x)$. Näin saadaan merkittäviä tehokkuusparannuksia verrattuna kertojiin, jotka tukevat kaikkia alkulukuja tai jaottomia polynomeja. Tietysti vastaavia optimointeja voidaan tehdä myös sovulluskohtaisen integroidun piirin tapauksessa, mutta tällöin toteutus on pakotettu käyttämään vain näitä tiettyjä parametreja, mikä rajoittaa toteutuksen käyttöä ja päivitettävyyttä merkittävästi (yleensä liikaa, jotta ne olisivat käytännössä järkeviä). FPGA-piirien tapauksessa optimoinnit voidaan tehdä ilman, että järjestelmän käytettävyys tai päivitettävyys merkittävästi kärsii, koska piiri voidaan aina ohjelmoida uudestaan toisille parametreille.

6 Esimerkki: Erittäin nopea FPGA-toteutus elliptisen käyrän salausmenetelmälle

Seuraavassa käydään pääpiirteittäin läpi kirjoittajan tekemä FPGA-toteutus elliptisen käyrän salausmenetelmästä [4]. Toteutus pyrkii mahdollisimman suureen laskentanopeuteen ja sitä voidaan hyödyntää salausjärjestelmien laskennan nopeuttamiseen esimerkiksi raskaasti kuormitetuissa palvelimissa.

Toteutus hyödyntää monia edellä esitettyjä optimointeja. Kertolasku elliptisellä käyrällä lasketaan esilaskentaa hyödyntävällä algoritmilla ja pisteoperaatiot lasketaan projektiivisia koordinaatteja käyttäen. Myös FPGA-piirien uudelleen ohjelmoitavuuden mahdollistamaa tietyille parametreille optimointia hyödynnetään kaikilla kertolaskun hierarkian tasoilla. Toteutus on suunnitelu käyttämään vain tietyn tyyppisiä elliptisiä käyriä eli ns.

Koblitzin käyriä [7], joiden perusperiaatteet käydään läpi seuraavassa. Toteutuksen arkkitehtuuria voidaan soveltaa kaikille Koblitzin käyrille, mutta seuraavassa esitettävät tulokset ovat ainoastaan standardoidulle käyrälle NIST K-163 [11]. Äärellisen kunnan kertojat on suunniteltu vain tämän käyrän äärelliselle kunnalle, $\mathbb{F}_{2^{163}}$, ja sen jaottomalle polynomille $p(x)$.

6.1 Koblitzin käyrät

Koblitzin käyrät [7] ovat binäärikuntien elliptisiä käyriä, joilla on ominaisuus, että jos piste $P = (x,y)$ on käyrällä, myös sen Frobeniuksen kuvaus eli $\phi(P) = (x^2, y^2)$ on piste samalla käyrällä. Tämä ominaisuus mahdollistaa kertolaskun laskemisen siten, että pisteen kertominen kahdella korvataan Frobeniuksen kuvauksella.

Frobeniuksen kuvausta ei kuitenkaan voida suoraan laskea pisteen kertomisen sijasta, koska $2P = \mu\phi(P) - \phi^2(P)$, missä $\mu = \pm 1$ riippuen kyseisestä käyrästä. Jotta Frobeniuksen kuvauksia voidaan hyödyntää kertolaskussa, tulee kokonaisluku k esittää siten, että siirtymä bitistä toiseen vastaa Frobeniuksen kuvausta eikä pisteen kertomista kahdella kuten normaalissa binääriesityksessä. Kun tämä on tehty, voidaan kertolasku suoritetaan binäärialgoritmeilla, joissa pisteen kahdella kertomisen sijasta lasketaan Frobeniuksen kuvaus.

Edellisestä pisteen kahdella kertomisen ja Frobeniuksen kuvauksen yhteyden kaavasta saadaan, että k:lle pitää etsiä seuraava τ-adinen esitys $k = \sum_{i=0}^{\ell-1} \varepsilon_i \tau^i$, missä $\tau = (\mu + \sqrt{-7})/2$. On löydettävissä kappaleessa 3.3 käsiteltyjä esityksiä vastaavat τ-adiset esitykset; esim. w-levyinen τ-adinen NAF (w-τNAF), jolle pätee $h(k) \approx \ell/(w+1)$ [13]. Toteutus käyttää 4-τNAF:ia.

6.2 Toteutuksen arkkitehtuuri

Kuten edellisistä kappaleista on käynyt ilmi, on tehokkaan Koblitzin käyriä käyttävän toteutuksen kyettävä seuraaviin operaatioihin:

1. kokonaisluvun k muuntaminen binääriesityksestä τ-adiseen esitykseen,
2. esilaskenta pisteellä P,
3. kertolasku elliptisellä käyrällä ja
4. tulospisteen konvertointi projektiivisista koordinaateista tasokoordinaatteihin.

Toteutus pyrkii laskemaan operaatioita rinnakkain niin paljon kuin mahdollista. Yllä olevasta luettelosta kohdat 1 ja 2 eivät riipu toisistaan, joten ne voidaan laskea rinnakkain. Tämän jälkeen rinnakkaisuus toteutetaan laskemalla useampia kertolaskuja samanaikaisesti: yhdelle kertolaskulle lasketaan luettelon kohtia 1 ja 2, toiselle kohtaa 3 ja kolmannelle kohtaa 4. Tämän mukainen toteutuksen arkkitehtuuri on esitetty kuvassa 5.

Luettelon kohta 3 on laskennallisesti vaativin ja suurin osa toteutuksen vaatimista resursseista kuluu sen toteuttamiseen. Laskenta suoritetaan esilaskentaa hyödyntävällä binäärialgoritmilla oikealta vasemmalle eli se koostuu pisteiden yhteen- ja vähennyslaskuista sekä Frobeniuksen kuvauksista. Arkkitehtuuri pyrkii hyödyntämään rinnakkaisuutta myös kertolaskun laskemisessa. Se lasketaan projektiivisia koordinaatteja (ks. kappale 3.2) hyödyntäen eli pisteet esitetään kolmella koordinaatilla (X,Y,Z). Tehokkain algoritmi pisteiden yhteenlaskulle näillä koordinaateilla vaatii kahdeksan kunnan kertolaskua [1] siten, että kriitisellä polulla on neljä kertolaskua, jos käytössä on vähintään kolme kertojaa. Kriittinen polku voidaan kuitenkin lyhentää vain kahteen kunnan kertolaskuun yhtä pisteiden yhteenlaskua kohden käyttämällä neljää kertojaa ja laskemalla peräkkäiset yhteenlaskut limittäin [3]. Tämä on mahdollis-

Kuva 5: Toteutuksen korkean tason arkkitehtuuri

ta, koska yhteenlaskun tulospisteen (P_3) Z-koordinaatin laskentaan ei tarvita toisen lähtöpisteen (P_1) Y-koordinaattia ja siten sen laskenta voidaan aloittaa vapaana olevissa kertojissa ennen edellisen Y-koordinaatin laskennan valmistumista [3].

Kappaleessa 3.3 mainittiin, että esilaskentaa hyödyntävissä kertolaskualgoritmeissa laskenta suoritetaan vasemmalta oikealle, koska muuten jokainen esilaskettu piste pitää kertoa kahdella jokaisella kierroksella. Sama pätee luonnollisesti myös Koblitzin käyrillä eli jokaiselle esilasketulle pisteelle pitää laskea Frobeniuksen kuvaus jokaisella kierroksella. Tämä ei kuitenkaan ole tässä tapauksessa ongelma, koska kuvauksien laskenta on niin nopeaa, että ne ehditään suorittaa rinnakkaisella yksiköllä ennen seuraavan pisteiden yhteenlaskun aloittamista.

Arkkitehtuuri siis laskee kertolaskun Koblitzin käyrällä siten, että laskennan kriittinen polku koostuu ainoastaan pisteiden yhteenlaskuista, joista jokaisen pituus on kaksi kunnan kertolaskua. Koska toteutuksessa käytetään 4-τNAF:ia, koko kertolaskun kriittinen polku koostuu vain noin $2\frac{163}{5} \approx 65$ kunnan kertolaskusta. Huomaa, että ilman rinnakkaisuuden hyödyntämistä polku olisi yli 260 kunnan kertolaskua.

6.3 Tulokset

Kappaleessa 6.2 esitetty arkkitehtuuri kuvattiin VHDL-kielellä ja käännettiin FPGA-valmistaja Alteran työkaluilla Altera Stratix II EP2S180F1020C3 FPGA-piirille.

Toteutus laskee kertolaskun elliptisellä käyrällä keskimäärin 11,71 mikrosekunnissa. Arkkitehtuurissa käytetty rinnakkaisuus mahdollistaa kuitenkin peräti 235550 kertolaskun laskemisen sekunnissa, sillä laskennan pullonkaulana on kappaleen 6.2 luettelon kohta 3, joka vaatii keskimäärin 4,25 mikrosekuntia.

Arkkitehtuuri vaati vain 20 % kyseisen piirin resursseista. Voidaan siis olettaa, että kyseiselle piirille saadaan mahtumaan ainakin neljä rinnakkaista yksikköä ilman, että yhden yksikön nopeus merkittävästi kärsii. Lisäksi Stratix II -piirit ovat jo melko vanhoja (Alteran uusin piirisarja on Stratix V), joten on turvallista sanoa, että yhdellä nykyaikaisella FPGA-piirillä pystytään laskemaan yli miljoona kertolaskua sekunnissa elliptisellä käyrällä, jonka varaan rakennettua salausjärjestelmää ei nykytiedon valossa kyetä murtamaan nyt eikä lähitulevaisuudessa. Samaa arkkitehtuuria voidaan toki käyttää myös korkeamman turvallisuustason Koblitzin käyrille.

Kiitokset

Kirjoittaja haluaa kiittää Billy Brumleya kuvan 1 tekemisestä ja luvasta käyttää sitä tässä artikkelissa.

Viitteet

1. Essame Al-Daoud, Ramlan Mahmod, Mohammad Rushdan ja Adem Kilicman: A new addition formula for elliptic curves over $GF(2^n)$, *IEEE Transactions on Computers*, 51 (2002), IEEE, s. 972–975.
2. Darrel Hankerson, Alfred Menezes ja Scott Vanstone: *Guide to Elliptic Curve Cryptography*. Springer-Verlag, 2004.
3. Kimmo Järvinen ja Jorma Skyttä: Fast point multiplication on Koblitz curves: Parallelization method and implementations, *Microprocessors and Microsystems*, 33 (2009), Elsevier, s. 106–116.
4. Kimmo Järvinen: Optimized FPGA-based elliptic curve cryptography processor for high-speed applications, *Integration—the VLSI Journal*, Elsevier, painossa. DOI: *10.1016/j.vlsi.2010.08.001*.
5. Thorsten Kleinjung, Kazumaro Aoki, Jens Franke, Arjen K. Lenstra, Emmanuel Thomé, Joppe W. Bos, Pierrick Gaudry, Alexander Kruppa, Peter L. Montgomery, Dag Arne Osvik, Herman te Riele, Andrey Timofeev ja Paul Zimmermann: Factorization of a 768-bit RSA Modulus, *Advances in Cryptology—CRYPTO'10, Lecture Notes in Computer Science, vol. 6223*. Springer-Verlag, 2010, s. 333–350.
6. Neal Koblitz: Elliptic curve cryptosystems, *Mathematics of Computation*, 48 (1987), AMS, s. 203–209.
7. Neal Koblitz: CM-curves with good cryptographic properties, *Advances in Cryptology—CRYPTO'91, Lecture Notes in Computer Science, vol. 576*. Springer-Verlag, 1992, s. 279–287.
8. Paul C. Kocher: Timing attacks on implementations of Diffie–Hellman, RSA, DSS, and other systems, *Advances in Cryptology—CRYPTO'96, Lecture Notes in Computer Science, vol. 1109*. Springer-Verlag, 1996, s. 104–113.
9. Paul Kocher, Joshua Jaffe ja Benjamin Jun: Differential power analysis, *Advances in Cryptology—CRYPTO'99, Lecture Notes in Computer Science, vol. 1666*. Springer-Verlag, 1999, s. 388–397.
10. Victor Miller: Use of elliptic curves in cryptography, *Advances in Cryptology—CRYPTO'85, Lecture Notes in Computer Science, vol. 218*. Springer-Verlag, 1986, s. 417–426.
11. National Institute of Standards and Technology (NIST): Digital signature standard (DSS), *Federal Information Processing Standard, FIPS PUB 186-3*, kesäkuu 2009.
12. Kaisa Nyberg: Kryptologia — tiedon turvaamisen tiede, *Tietojenkäsittelytiede*, 26 (2007), s. 32–53.
13. Jerome A. Solinas: Efficient arithmetic on Koblitz curves, *Designs, Codes and Cryptography*, 19 (2000), Springer, s. 195–249.
14. Thomas Wollinger, Jorge Guajardo ja Christof Paar: Security on FPGAs: State-of-the-art implementations and attacks, *ACM Transactions on Embedded Computing Systems*, 3 (2004), ACM Press, s. 534–574.