PROTOKOL *ZK-SNARK* SEBAGAI ALTERNATIF DALAM *CRYPTOCURRENCY*

TUGAS AKHIR

Karya tulis sebagai salah satu syarat
untuk memperoleh gelar Sarjana dari
Institut Teknologi Bandung

Oleh
ARYASUTA NAYOTTAMA
NIM: 10118040
Program Studi Sarjana Matematika

INSTITUT TEKNOLOGI BANDUNG
September 2022

PROTOKOL *ZK-SNARK* SEBAGAI ALTERNATIF DALAM *CRYPTOCURRENCY*

Oleh
ARYASUTA NAYOTTAMA
NIM: 10118040
Program Studi Sarjana Matematika

Institut Teknologi Bandung

Menyetujui
Pembimbing
Tanggal 2 September 2022

(Dr. Intan Muchtadi)

Prakata

Puji syukur dipanjatkan kepada Tuhan Yang Maha Esa, sebab atas pertolongan nya, penulis dapat menyelesaikan tugas akhir dan mendokumentasikannya dalam buku ini.

Pada kesempatan ini pula, penulis tidak lupa untuk mengucapkan terima kasih atas segala bantuan, bimbingan, semangat, dan dorongan dari berbagai pihak yang turut berperan selama proses penyusunan tugas akhir ini.

1. Ibu dan ayah tercinta, Budi Setiawan Handoyo dan Maria Susanti Hadi. Terima kasih atas segala bentuk cinta dan dukungan kepada penulis.

2. Dosen wali penulis, Dr. Novry Erwina. Terima kasih atas segala nasihat dan saran yang diberikan kepada penulis.

3. Dosen pembimbing, Dr. Intan Muchtadi. Terima kasih atas segala bimbingan dan dukungan yang diberikan, dan kesabarannya dalam membimbing di waktu pandemi.

4. Dosen penguji: Ibu Pritta Eriana Putri dan Dr. Fajar Yuliawan. Terima kasih atas dorongan, saran, dan kritik yang membangun dan mendorong penulis untuk terus belajar.

5. Dr. Fajar Yuliawan, selaku dosen aljabar. Terima kasih atas saran dan masukannya dalam membangun sebuah program.

6. Sahabat dekat penulis di Matematika ITB: Frietz Noor Abraham, Rahaditya Muhammad Zuhd, dan Mikhael Belmiro. Terima kasih atas seluruh pengalaman selama berkuliah di ITB.

7. Sahabat satu kos Cimbel71. Terima kasih atas semangat yang diberikan kepada penulis selama berkuliah di ITB.

8. Sahabat *discord*, terutama: Amadeus Ryo Luchen dan Fernaldy Sudianto. Terima kasih atas saran, bantuan, dan semangat yang diberikan, serta menjadi tempat curhat pagi penulis.

Tentunya masih banyak pihak yang berperan dalam membantu penulis, tapi karena keterbatasan waktu, kertas, dan tinta, penulis tidak dapat menyebutkan satu- per-satu. Akhir kata, penulis dengan kerendahan hati menerima segala saran dan masukan yang membangun. Semoga tugas akhir ini dapat bermanfaat bagi para pembaca dan semua pihak yang bersangkutan.

Jakarta, 11 Juli 2022

Penulis,
Aryasuta Nayottama

Sari

Sekarang ini ramai diperbincangkan mengenai *cryptocurrency* terutama *bitcoin*. *Cryptocurrency* sendiri adalah mata uang digital yang diamankan dengan menggunakan kriptografi. *Bitcoin* dan *zcash* adalah salah satu dari sekian banyak mata uang digital tersebut. Tetapi transaksi *bitcoin* tidak menyembunyikan data mengenai transaksi yang dilakukan. Berbeda dengan *zcash*, dimana data transaksi dapat disembunyikan. *Zcash* melakukan ini dengan menggunakan *zero knowledge proof*. *Zero knowledge proof* adalah metode yang digunakan untuk memverifikasi apakah seseorang mengetahui suatu informasi tanpa memberitahu informasi tersebut.

Bahasan pertama dalam tugas akhir ini adalah teori dasar yang digunakan dalam *zk-SNARKs*. *Zk-SNARKs* adalah salah satu protokol peningkatan dari *zero knowledge proof*, dimana proses verifikasi berlangsung singkat dan hanya terjadi satu interaksi. Dalam prosesnya, *zk-SNARKs* menggunakanlapangan hingga, pemetaan bilinear, dan interpolasi Lagrange.

Bahasan kedua adalah kurva eliptik dan *zk-SNARKs* sendiri. Selain itu, skema dan juga algoritma dari *zk-SNARKs* akan dijelaskan, beserta contohnya. Di tugas akhir ini juga akan dijelaskan mengenai *zero knowledge proof* beserta contoh sederhananya.

Kata kunci: kurva eliptik, lapangan hingga, pemetaan bilinear, interpolasi Lagrange.

Abstract

Nowadays, many people are talking about *cryptocurrency* especially *bitcoin*. *Cryptocurrency* is a digital currency that is secured using cryptography. *Bitcoin* and *zcash* are two of the many digital currencies. But the *bitcoin* transactions are not concealing the data about those transactions. However, *zcash* is different, it can conceal the transaction's data. *Zcash* uses *zero knowledge proof*. *Zero knowledge proof* is a method used to verify whether someone knows an information without revealing the information.

This final project first discusses about ground theories that are used in *zk-SNARKs*. *Zk-SNARKs* is one of *zero knowledge proof* protocols improved, which the verification process happens quickly and there is only one interaction. In the process, *zk-SNARKs* uses finite field, bilinear mapping, and Lagrange interpolation.

The second subject this final project discusses is elliptic curve and *zk-SNARKs* itself. Also, it discusses about the scheme and algorithm of *zk-SNARKs* alongside few examples. This final project also explains about *zero knowledge proof* and simple examples.

Keywords: elliptic curve, finite field, bilinear mapping, Lagrange interpolation.

Daftar Isi

Bab I Pendahuluan

Sekarang ini *cryptocurrency* menjadi topik pembicaraaan dimana-mana. Apa itu *cryptocurrency*? *Cryptocurrency* adalah mata uang digital yang didesain sebagai alat pertukaran seperti uang yang dilakukan melalui koneksi komputer. Berbeda dengan uang fiat yang bersifat *centralized* (terpusat), *cryptocurrency* bersifat *decentralized* (terdesentralisasi). Uang *centralized* bersifat terpusat karena dicetak, diedarkan, dan dikelola oleh satu pihak misalnya pemerintah. Sedangkan, *cryptocurrency* bersifat *decentralized* karena tidak bergantung pada satu pihak terpusat seperti pemerintah atau bank untuk dikelola.

Cryptocurrency tidak memiliki bentuk fisik seperti uang kertas, kepemilikan *cryptocurrency* biasa disebut dengan istilah *coin*. Setiap transaksi dan kepemilikan *cryptocurrency* disimpan dalam buku besar digital yang disimpan dalam *database* di berbagai komputer. Buku besar juga diamankan menggunakan kriptografi untuk mengontrol penambahan *coin* dan memverifikasi transfer kepemilikan *coin*.

Sama seperti uang fiat, *cryptocurrency* juga memiliki banyak kurensi yang beredar. Salah satunya adalah *Bitcoin*. *Bitcoin* adalah *crpytocurrency* pertama yang beredar pada tahun 2009. *Bitcoin* diciptakan pertama kali oleh sekelompok orang bernama Satoshi Nakamoto [9]. Selain *Bitcoin*, terdapat banyak kurensi lainnya seperti *Dogecoin*, *Ethereum*, *Zcash*, dll.

Berbeda dengan transaksi uang fiat yang dilakukan melalui pihak terpusat yaitu bank, transaksi *cryptocurrency* dilakukan secara *peer-to-peer* yaitu transaksi dilakukan secara langsung antara pengirim dan penerima tanpa ada pihak terpusat seperti bank. Data mengenai transaksi tersebut juga ditampilkan dan disimpan dalam *blockchain* dimana setiap transaksi yang disimpan saling berhubungan satu sama lain, hal ini dilakukan untuk mencegah kecurangan yang terjadi pada suatu transaksi. *Bitcoin* dan beberapa *cryptocurrency* lainnya menampilkan data transaksi seperti jumlah transaksi, kurensi pada transaksi, pengirim, dan penerima secara publik. Artinya orang-orang dapat melihat siapa yang mengirim dan siapa yang menerima transaksi. *Zcash* mengatasi hal ini dengan menggunakan *zero knowledge proof* sehingga transaksi yang dilakukan dapat menyembunyikan data transaksi.

Zero knowledge proof adalah suatu metode yang digunakan dimana satu pihak (*prover*) ingin membuktikan kepada pihak lain (*verifier*) bahwa suatu pernyataan benar tanpa ada informasi lain yang keluar selain fakta bahwa pernyataan tersebut benar.

Zero knowledge proof digunakan oleh *Zcash* dalam proses transaksi untuk membuktikan bahwa kondisi-kondisi untuk suatu transaksi sudah terpenuhi tanpa memberikan informasi lain seperti alamat transaksi atau jumlah transaksi. Lebih lanjut, *Zcash* menggunakan *zk-SNARK* yang merupakan protokol peningkatan dari *zero knowledge proof* dimana proses verifikasi terjadi dengan singkat dan hanya terdapat satu interaksi.

Dalam tugas akhir ini, akan dijelaskan bagaimana *zero knowledge proof* dapat menyembunyikan data transaksi.

Penulisan buku tugas akhir ini terbagi menjadi lima bab: pendahuluan, komponen dasar *zk-SNARK*, skema *zk-SNARK*, program *zk-SNARK*, dan simpulan dan saran.

Pada bab pendahuluan, akan dipaparkan latar belakang dan rumusan masalah yang mendasari penelitian yang dilakukan pada tugas akhir ini. Akan dipaparkan juga sistematika pembahasan buku tugas akhir ini.

Pada bab komponen dasar *zk-SNARK*, akan dipaparkan dan dijelaskan teori-teori apa saja yang dibutuhkan dalam membangun skema *zk-SNARK*. Pertama, akan dijelaskan mengenai lapangan hingga. Kedua, akan dijelaskan mengenai kurva eliptik dan juga pemetaan bilinear. Ketiga, akan dijelaskan mengenai interpolasi Lagrange dan juga *QAP* (*Quadratic Arithmetic Program*).

Pada bab skema *zk-SNARK*, akan dijelaskan bagaimana cara *zk-SNARK* bekerja. Pertama, akan dijelaskan mengenai *zero knowledge proof*, berbagai contoh, dan juga *zk-SNARK*. Kedua, akan dijelaskan bagaimana komponen-komponen dasar di bab sebelumnya digunakan dalam *zk-SNARK* dan contoh sederhana.

Pada bab program *zk-SNARK*, akan dipaparkan algoritma *zk-SNARK* dalam *pseudo code* dalam bahasa *Python*.

Pada bab terakhir, simpulan dan saran, berisi simpulan dari seluruh proses dan hasil yang didapatkan pada tugas akhir ini. Juga memuat saran agar penelitian mengenai *zero knowledge proof* dapat berjalan lebih baik.

Bab II Komponen dasar *zk-SNARK*

II.1 Lapangan Hingga

Dalam prosesnya, *zk-SNARK* dan kurva eliptik menggunakan lapangan hingga . Oleh karena itu, sebelum memasuki penjelasan mengenai *zk-SNARK*, perlu dibahas terlebih dahulu mengenai lapangan hingga.

II.1.1 Grup

Definisi 2.1. Suatu *grup* $(G,*)$ adalah himpunan G dengan operasi biner $*$ pada G yang memenuhi sifat-sifat berikut:

1. *Tertutup*: $\forall x,y \in G$ berlaku $x*y \in G$.
2. *Asosiatif*: $(x*y)*z = x*(y*z)$ untuk setiap $x,y,z \in G$.
3. *Identitas*: terdapat elemen $e \in G$ yang memenuhi $x*e = e*x = x$ untuk setiap $x \in G$.
4. *Invers*: untuk setiap elemen $x \in G$ terdapat elemen $x^{-1} \in G$ yang memenuhi $x*x^{-1} = x^{-1}*x = e$.

Definisi 2.2. Suatu grup dikatakan berhingga jika memiliki berhingga banyaknya elemen. Kardinalitas dari suatu grup berhingga G disebut orde, ditulis $|G|$.

Definisi 2.3. Suatu grup G dikatakan *abelian* atau *komutatif* jika untuk setiap $x,y \in G$ berlaku $x*y = y*x$.

Definisi 2.4. Suatu subhimpunan H dari grup G adalah *subgrup* dari G jika H:

1. H tidak kosong,
2. H tertutup terhadap operasi $*$ di G,
3. Himpunan H dengan operasi $*$ membentuk grup.

Definisi 2.5. Suatu grup perkalian $(G,*)$ dikatakan *siklik* jika terdapat suatu elemen $a \in G$ dimana untuk setiap $b \in G$, $b = a^j$ untuk suatu bilangan bulat j. Elemen tersebut disebut generator dari grup siklik, dan dapat ditulis $G = \langle a \rangle$.

II.1.2 Gelanggang dan Lapangan

Selain grup, juga terdapat gelanggang dan lapangan. Perbedaan mendasar antara grup dan gelanggang adalah operasi yang didefinisikan pada himpunan. Grup didefinisikan dengan satu operasi, sedangkan gelanggang didefinisikan dengan dua operasi.

Definisi 2.6. Suatu *gelanggang* $(R,+,*)$ adalah suatu himpunan R, dengan dua operasi biner $+$ dan $*$ dimana

1. $(R,+)$ membentuk grup komutatif,
2. R tertutup terhadap operasi $*$,
3. asosiatif terhadap operasi $*$, yaitu $(a*b)*c = a*(b*c)$ untuk setiap $a,b,c \in R$,
4. berlaku sifat *distributif*, yaitu untuk setiap $a,b,c \in R$ berlaku $a*(b+c) = (a*b)+(a*c)$ dan $(b+c)*a = (b*a)+(b*c)$.

Definisi 2.7.

- Suatu gelanggang disebut *gelanggang dengan identitas* jika gelanggang tersebut mempunyai identitas perkalian (biasa dinotasikan dengan e atau 1).
- Suatu gelanggang disebut *komutatif* jika operasi $*$ komutatif.
- Suatu gelanggang disebut *daerah integral* jika gelanggang tersebut adalah gelanggang komutatif dengan identitas $e \neq 0$ dimana $ab = 0$ mengakibatkan $a = 0$ atau $b = 0$.
- Suatu gelanggang disebut *gelanggang pembagian* jika elemen- elemen tak nol membentuk grup terhadap operasi $*$.
- Gelanggang pembagian yang komutatif disebut *lapangan*.

Lapangan dapat juga didefinisikan sebagai berikut.

Definisi 2.8. Suatu *lapangan* $(F,+,*)$ adalah suatu himpunan dengan dua operasi biner $+$ dan $*$ dimana

1. $(F,+)$ membentuk grup komutatif,
2. $(F-0,*)$ membentuk grup komutatif,

3. memenuhi sifat *distributif* dimana untuk setiap $a,b,c \in R$ berlaku $a*(b+c) = (a*b)+(a*c)$ dan $(b+c)*a = (b*a)+(b*c)$.

Operasi $+$ biasa disebut operasi aditif atau penjumlahan, sedangkan operasi $*$ biasa disebut multiplikatif atau perkalian.

Biasanya, 0 ditulis untuk menotasikan elemen identitas pada grup komutatif F terhadap operasi penjumlahan, dan $-a$ untuk menotasikan invers penjumlahan dari $a \in F$.

Sama seperti grup, himpunan F dapat memiliki kardinalitas berhingga atau tak hingga. Ketika kardinalitas F berhingga, maka F disebut *lapangan hingga* atau *lapangan Galois*.

Teorema 2.1. Setiap daerah integral hingga adalah lapangan.

Bukti. Misalkan R adalah daerah integral hingga dengan elemen $a_1, a_2, \ldots, a_n$. Untuk suatu elemen tetap tak nol a, $aa_1, aa_2, \ldots, aa_n$. $aa_1, aa_2, \ldots, aa_n$ saling berbeda, karena jika $aa_i = aa_j$, maka $a(a_i - a_j) = 0$ dan karena $a \neq 0$ haruslah $a_i - a_j = 0$ atau $a_i = a_j$. Sehingga setiap elemen R dapat ditulis dalam bentuk aa_i, secara khusus $e = aa_i$ untuk suatu i dengan $1 \leq i \leq n$, dimana e adalah elemen identitas R. Karena R komutatif, $a_i a = e$ dan a_i adalah invers perkalian a. Sehingga elemen-elemen tak nol R membentuk grup komutatif dan R adalah lapangan.

Definisi 2.9. Suatu lapangan $(F,+,*)$ dapat memiliki karakteristik n apabila untuk setiap $a \in F$ bilangan n adalah bilangan bulat positif terkecil sehingga $na = 0$. Apabila tidak ada n yang memenuhi, maka karakteristik dari F adalah 0.

II.1.3 Lapangan Hingga

Diberikan suatu bilangan bulat n. Definisikan $[x]_n$ sebagai himpunan semua bilangan bulat yang mempunyai residu (sisa) yang sama dengan pembagian x dengan n.

Contoh 2.1. Misal $n = 5$. Perhatikan bahwa $12 = 5 \cdot 2 + 2$ dan $27 = 5 \cdot 5 + 2$. Jadi sisa pembagian 12 dan 27 adalah 2. Dengan demikian $[12]_5 = [27]_5$ dan diperoleh juga bahwa $12, 27 \in [12]_5$.

Sekarang jika kita tulis $[12]_5 = \{\ldots, -8, -3, 2, 7, 12, 17, \ldots\}$ untuk setiap $a, b \in [12]_5$ kita peroleh $[a] = [b]$. Misalnya $[-8]_5 = [12]_5 = [17]_5$.

Diberikan suatu bilangan bulat n, definisikan $\mathbb{Z}_n := \{[x]_n : x \in \mathbb{Z}\}$. Dengan demikian, jelas bahwa $\mathbb{Z}_n := \{[0], [1], \ldots, [n-1]\}$. Masing-masing dari $[x]$ disebut sebagai *kelas residu* yang mengandung x. Himpunan $\mathbb{Z}_n$ disebut sebagai *gelanggang kelas residu.*

Teorema 2.2. Gelanggang kelas residu $\mathbb{Z}_p$ dengan p prima adalah lapangan.
Bukti. Berdasarkan Teorema 2.1., cukup diperlihatkan bahwa $\mathbb{Z}_p$ adalah daerah integral. Elemen $[1]$ merupakan identitas dari $\mathbb{Z}_p$ dan $[a][b] = [ab] = [0]$ jika dan hanya jika $ab = kp$ untuk suatu bilangan bulat k. Karena p prima, p membagi ab jika dan hanya jika p membagi a atau b. Sehingga $[a] = [0]$ atau $[b] = [0]$, maka $\mathbb{Z}_p$ tidak memiliki pembagi nol.

Misalkan p bilangan prima. Definisikan lapangan *Galois* dengan orde p, $\mathbb{F}_p$ sebagai $\mathbb{Z}_p$.

Untuk selanjutnya, $\mathbb{F}[x]$ digunakan untuk menyatakan *gelanggang polinomial atas lapangan* $\mathbb{F}$, kecuali apabila dikatakan lain.

Definisi 2.10. Polinomial $p(x) \in \mathbb{F}[x]$ disebut tak tereduksi (*irreducible*) jika $p(x)$ berderajat positif dan $p(x)$ tidak dapat dinyatakan sebagai perkalian antara dua polinomial berderajat positif dan lebih kecil daripada derajat $p(x)$. Dengan kata lain, jika $p(x) = a(x)b(x)$ maka $a(x)$ konstan atau $b(x)$ konstan.

Contoh 2.2. $x^2 + 1 \in \mathbb{R}[x]$ tidak tereduksi di $\mathbb{R}$ tetapi sebagai elemen $\mathbb{C}[x]$ merupakan polinomial tereduksi.

Definisi 2.11. Misal F lapangan dan $f(x) \in F[x]$ polinomial tak tereduksi, $F[x]/\langle f(x)\rangle$ bersifat seperti gelanggang kelas residu $\mathbb{Z}_n$, dimana $\langle f(x)\rangle$ merupakan ideal yang dibangun oleh $f(x)$.

$$\begin{aligned} F[x]/\langle f(x)\rangle &:= \{p(x) + \langle f(x)\rangle : p(x) \in F[x]\} \\ &= \{\overline{q(x)} : q(x) \text{ adalah sisa pembagian dengan } f(x)\}. \end{aligned}$$

Contoh 2.3. Misal $F = \mathbb{F}_2$ dengan $f(x)$ polinomial tak tereduksinya adalah $x^3 + x + 1$. Maka,

$$\begin{aligned} \mathbb{F}_2[x]/\langle f(x)\rangle &= \{\text{sisa pembagian dengan } x^3 + x + 1\} \\ &= \{\overline{a_0 + a_1x + a_2x^2} : a_0, a_1, a_2 \in \mathbb{F}_2\} \end{aligned}$$

Ambil $x^4 + x + 1 \in \mathbb{F}_2[x]$, perhatikan bahwa $x^4 + x + 1 = f(x)\cdot x + x^2 + 1$. Maka $\overline{x^4 + x + 1} = \overline{x^2 + 1} \in \mathbb{F}_2[x]/\langle f(x)\rangle$.

Sehingga didapatkan

$$\mathbb{F}_2[x]/\langle f(x)\rangle = \left\{\overline{0}, \overline{1}, \overline{x}, \overline{x+1}, \overline{x^2}, \overline{x^2+1}, \overline{x^2+x}, \overline{x^2+x+1}\right\}$$

dan dapat ditulis sebagai

$$\mathbb{F}_2[x]/\langle f(x)\rangle = \{(000),(001),(010),(011),(100),(101),(110),(111)\}$$

II.2 Enkripsi Homomorfik

II.2.1 Enkripsi Homomorfik

Enkripsi homomorfik dilakukan dengan mengambil suatu nilai basis g (misal 5), kemudian mengenkripsi nilai yang kita inginkan dengan memangkatkan g dengan nilai tersebut. Contoh, kita memangkatkan 5 dengan 3, maka hasil enkripsinya adalah

$$5^3 = 125.$$

Selain itu, kita juga bisa mengalikan nilai enkripsi dengan nilai lain.

$$125^2 = 15625 = (5^3)^2 = 5^{2\cdot 3} = 5^6$$

Jika kita ingin menambahkan nilai enkripsi, misal $3+2$, diperoleh

$$5^3 \cdot 5^2 = 5^{3+2} = 5^5 = 3125.$$

Sama halnya dengan mengurangi nilai enskripsi $3-2$.

$$\frac{5^3}{5^2} = 5^3 \cdot 5^{-2} = 5^{3-2} = 5^1 = 5.$$

Tetapi, karena nilai basis 5 diketahui publik, maka cukup mudah untuk mencari nilai rahasia, yaitu dengan membagi nilai enkripsi dengan 5 sampai diperoleh 1. Banyaknya langkah pembagian dengan 5 adalah nilai rahasianya. Maka, diperlukan enkripsi homomorfik kuat dengan lapangan hingga.

II.2.2 Enkripsi Homomorfik Kuat

Misal nilai basis adalah 5 dan lapangan hingga yang digunakan adalah $\mathbb{F}_7$, maka

$$5^1 = 5 \pmod 7$$
$$5^2 = 4 \pmod 7$$
$$5^3 = 6 \pmod 7$$

Dengan nilai eksponen berbeda, dapat dihasilkan nilai enkripsi yang sama

$$5^5 = 3 \pmod 7$$

$$5^{11} = 3 \text{ (mod 7)}$$
$$5^{17} = 3 \text{ (mod 7)}$$

Sehingga, untuk mencari nilai eksponen atau nilai rahasia sangat sulit. Seperti enkripsi homomorfik pada sebelumnya, operasi yang dilakukan juga sama.

$$\begin{aligned}
\text{enkripsi :} & \quad 5^3 = 6 \text{ (mod 7)} \\
\text{perkalian :} & \quad 6^2 = (5^3)^2 = 1 \text{ (mod 7)} \\
\text{penjumlahan :} & \quad 5^3 \cdot 5^2 = 5^5 = 6 \text{ (mod 7)} \\
\text{pengurangan :} & 5^3 - 5^2 = \frac{6}{4} = 6 \cdot 4^{-1} = 5 = 6 \text{ (mod 7)}
\end{aligned}$$

Fungsi enkripsi homomorfik ini dinotasikan dengan

$$E(x) = g^x \text{ (mod } n)$$

dimana v adalah nilai yang ingin dienkripsi.

Tetapi, ada beberapa operasi yang tidak bisa dilakukan oleh enkripsi homomorfik ini:

1. perkalian antara dua nilai terenkripsi,
2. pembagian antara dua nilai terenkripsi, dan
3. memangkatkan nilai terenkripsi.

Misal diberikan polinomial $p(x) = ax^3 + bx^2 + cx + d$. Untuk mengenkripsi suatu polinomial, enkripsi homomorfik dilakukan sebagai berikut.

$$\begin{aligned}
E(x^3)^a \cdot E(x^2)^b \cdot E(x^1)^c \cdot E(x^0)^a &= (g^{x^3})^a \cdot (g^{x^2})^b \cdot (g^{x^1})^c \cdot (g^{x^0})^d \\
&= g^{ax^3} \cdot g^{bx^2} \cdot g^{cx^1} \cdot g^{dx^0} \\
&= g^{ax^3+bx^2+cx+d}
\end{aligned}$$

II.3 Kurva Eliptik dan Pemetaan Bilinear

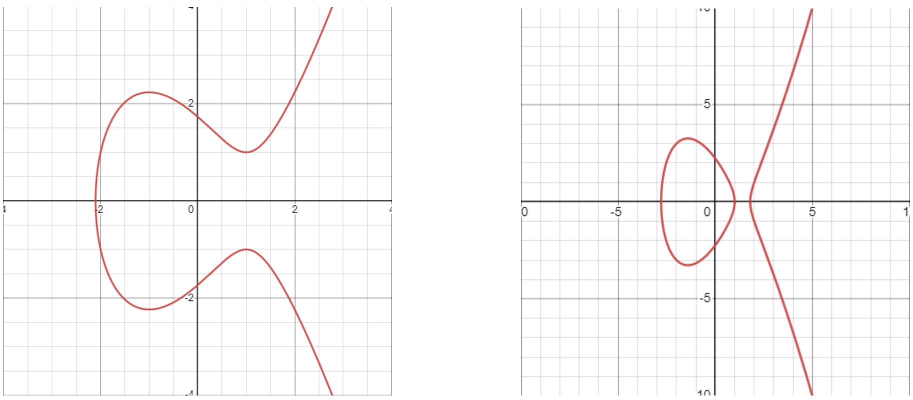
II.3.1 Kurva Eliptik

Sebuah *kurva eliptik* atas lapangan K dengan karakterisik $\neq 2, 3$ adalah himpunan solusi dari sebuah persamaan dengan bentuk

$$Y^2 = X^3 + AX + B.$$

Persamaan-persamaan dengan bentuk seperti diatas disebut dengan *persamaan Weierstrass*. Dua contoh kurva eliptik atas lapangan $\mathbb{R}$ diberikan sebagai berikut

$$E_1 : Y^2 = X^3 - 3X + 3 \quad \text{dan} \quad E_2 : Y^2 = X^3 - 6X + 5$$



(a) $E_1 : Y^2 = X^3 - 3X + 3$ (b) $E_2 : Y^2 = X^3 - 6X + 5$

Gambar 1: Kurva Eliptik

Satu sifat menarik dari kurva eliptik adalah adanya suatu cara natural yang dapat mengambil dua titik pada kurva dan "menjumlahkan" mereka untuk mendapatkan titik ketiga pada kurva yang sama. Cara menjumlahkan kedua titik pada kurva eliptik adalah sebagai berikut.

Diberikan dua titik P dan Q pada kurva eliptik E, tarik garis lurus L yang memotong P dan Q. Garis L ini akan memotong E pada tiga titik, yaitu P, Q, dan titik ketiga R. Ambil titik R, refleksikan terhadap garis sumbu-x untuk mendapatkan titik R'. Titik R' ini adalah hasil penjumlahan titik P dan Q. Untuk selanjutnya, notasi $\oplus$ digunakan sebagai operasi penjumlahan pada kurva eliptik.

Bagimana jika Q adalah P? Maka, garis L adalah garis singgung E pada P. Garis L akan berpotongan dengan titik lain sebut R. Kemudian refleksikan R terhadap sumbu-x, sehingga didapatkan titik R'. Maka, R' adalah hasil penjumlahan P dengan dirinya sendiri.

Kemudian, bagaimana jika $P = (a,b)$ ditambahkan dengan $P' = (a,-b)$? Garis L yang melalui P dan P' adalah garis vertikal dimana $x = a$, dan garis ini tidak berpotongan dengan titik ketiga. Solusinya adalah dengan menambahkan titik '$\mathcal{O}$' yang berada pada "ketidakhinggaan". Titik ini tidak ada pada bidang xy, tetapi kita anggap ada pada setiap garis vertikal. Sehingga $P \oplus P' = \mathcal{O}$.

Jika $P = (a,b)$, kita notasikan negasi dari P sebagai $\ominus P = (a,-b)$ atau $-P$. Sama seperti perkalian pada bilangan bulat, perkalian pada kurva eliptik ditulis sebagai

$$nP = \underbrace{P+P+P+\cdots+P}_{n \text{ kali}}$$

Definisi 2.12. Suatu kurva eliptik E adalah himpunan solusi dari persamaan *Weierstrass*

$$E : Y^2 = X^3 + AX + B,$$

bersama dengan titik $\mathcal{O}$ dimana nilai konstan A dan B harus memenuhi

$$4A^3 + 27B^2 \neq 0.$$

Kondisi $4A^3 + 27B^2 = \Delta_E$ dinamakan nilai diskriminan dari E. Kondisi $\Delta_E \neq 0$ adalah kondisi dimana $X^3 + AX + B$ tidak memiliki nilai akar yang berulang. Jika $\Delta_E = 0$, maka aturan penjumlahan yang sudah dijelaskan tadi tidak akan berfungsi dengan baik.

Misal E adalah kurva eliptik dengan $P = (a,b)$ dan $Q = (c,d)$ adalah titik pada E. Sekarang akan ditunjukkan bahwa kurva eliptik dengan aturan penjumlahan $(E,+)$ seperti di atas membentuk grup *abelian*.

- **Identitas** Ambil $Q = \mathcal{O}$, maka garis L adalah garis vertikal yang melalui P dan $P' = -P$. Refleksikan P' terhadap sumbu-x, didapatkan $(P')' = (a, -(-b)) = (a,b) = P$. Maka $\mathcal{O}$ adalah elemen identitas E.

- **Invers** Seperti yang sudah dibahas, bahwa P' adalah invers dari P.

- **Komutatif** Garis L yang menghubungkan P dengan Q adalah garis yang sama yang menghubungkan Q dengan P. Sehingga, E komutatif.

- **Asosiatif** Sifat asosiatif dapat dibuktikan dengan menggunakan algoritma penjumlahan di bawah ini.

Algoritma Penjumlahan Kurva Eliptik
Diberikan

$$E : Y^2 = X^3 + AX + B$$

kurva eliptik dan P_1 dan P_2 titik pada E.

(a) Jika $P_1 = \mathcal{O}$, maka $P_1 + P_2 = P_2$.

(b) Jika tidak, jika $P_2 = \mathcal{O}$, maka $P_1 + P_2 = P_1$.

(c) Jika tidak, tulis $P_1 = (x_1, y_1)$ dan $P_2 = (x_2, y_2)$.

(d) Jika $x_1 = x_2$ dan $y_1 = -y_2$, maka $P_1 + P_2 = \mathcal{O}$.

(e) Jika tidak, definisikan λ sebagai

$$\lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1} & \text{jika } P_1 \neq P_2, \\ \dfrac{3x_1^2 + A}{2y_1} & \text{jika } P_1 = P_2, \end{cases}$$

dan

$$x_3 = \lambda^2 - x_1 - x_2 \quad \text{dan} \quad y_3 = \lambda(x_1 - x_3) - y_1.$$

Maka $P_1 + P_2 = (x_3, y_3)$.

Untuk selanjutnya $E(K)$ digunakan untuk menyatakan kurva eliptik yang dibangun atas lapangan K (seperti $\mathbb{F}_p$) berkarakteristik $\neq 2,3$, dimana

$$E : Y^2 = X^3 + AX + B$$

dengan $A, B \in K$ yang memenuhi $4A^3 + 27B^2 \neq 0$.

$$E(K) = \{(x,y) : x,y \in K \text{ memenuhi } y^2 = x^3 + Ax + B\} \cup \{\mathscr{O}\}$$

II.3.2 Pemetaan Bilinear

Sebuah contoh pemetaan bilinear adalah perkalian titik pada ruang vektor $\mathbb{R}^n$,

$$\beta(\mathbf{v}, \mathbf{w}) = \mathbf{v} \cdot \mathbf{w} = v_1 w_1 + v_2 w_2 + \cdots + v_n w_n$$

dimana untuk setiap bilangan riil a_1, a_2, b_1, b_2,

$$\beta(a_1\mathbf{v_1} + a_2\mathbf{v_2}, \mathbf{w}) = a_1\beta(\mathbf{v_1}, \mathbf{w}) + a_2\beta(\mathbf{v_2}, \mathbf{w})$$
$$\beta(\mathbf{v}, b_1\mathbf{w_1} + b_2\mathbf{w_2}) = b_1\beta(\mathbf{v}, \mathbf{w_1}) + b_2\beta(\mathbf{v}, \mathbf{w_2})$$

Definisi 2.13. Diberikan grup siklik G_1, G_2, G_t dengan orde sama. Pemetaan bilinear dari $G_1 \times G_2$ ke G_t adalah fungsi $e : G_1 \times \rightarrow G_t$ untuk setiap $u \in G_1, v \in G_2, a, b \in \mathbb{Z}$,

$$e(u^a, v^b) = e(u,v)^{ab}.$$

Jika G adalah grup perkalian, $R, S \in G$, a, b bilangan bulat, dan e adalah pemetaan bilinear pada G, maka

$$e(R^a, S^b) = e(R,S)^{ab}$$

Pemetaan bilinear pada kurva eliptik memiliki dua titik sebagai *input* dan suatu angka sebagai *output*. Sebelum masuk ke pemetaan bilinear, kita perlu membahas tiga hal berkaitan pemetaan bilinear, yaitu titik-titik dengan orde berhingga, fungsi rasional, dan *divisor*.

Definisi 2.14. Suatu titik $P \in E$ dikatakan memiliki *orde* m jika

$$mP = \underbrace{P + P + \cdots + P = \mathcal{O}}_{m \text{ kali}},$$

dan $m'P \neq \mathcal{O}$ untuk setiap bilangan bulat $1 \leq m' < m$. Jika m ada, P dikatakan memiliki *orde berhingga*, jika tidak *orde tak berhingga*.

Definisi 2.15. Titik-titik $P \in E$ dengan orde m dinotasikan sebagai

$$\begin{aligned} E[m] = \{P \in E : mP = \mathcal{O}, \text{ dengan } m \text{ adalah} \\ \text{bilangan bulat positif terkecil yang memenuhi}\} \end{aligned}$$

Dapat dengan mudah dilihat bahwa jika P dan Q ada di $E[m]$, maka $P+Q$ dan $-P$ juga ada di $E[m]$, maka $E[m]$ adalah subgrup dari E. Jika koordinat P berada di lapangan K, maka kita tulis $E(K)[m]$.

Suatu polinomial rasional adalah rasio dari polinomial-polinomial

$$f(X) = \frac{a_0 + a_1X + a_2X^2 + \cdots + a_nX^n}{b_0 + b_1X + b_2X^2 + \cdots + b_mX^m}.$$

Dengan memperbolehkan bilangan kompleks, setiap polinomial dapat difaktorkan, sehingga polinomial rasional di atas menjadi

$$f(X) = \frac{a(X-\alpha_1)^{e_1}(X-\alpha_2)^{e_2}\dots(X-\alpha_r)^{e_r}}{b(X-\beta_1)^{d_1}(X-\beta_2)^{d_2}\dots(X-\beta_s)^{d_s}}.$$

Asumsi $\alpha_1,\dots,\alpha_r$ dan $\beta_1,\dots,\beta_s$ adalah nilai yang berbeda, karena jika tidak maka nilai yang sama pada pembilang dan penyebut dapat dihilangkan. Nilai $\alpha_1,\dots,\alpha_r$ disebut *zero dari* $f(X)$ dan $\beta_1,\dots,\beta_s$ disebut *pole dari* $f(X)$. Nilai eksponen $e_1,\dots,e_r$ dan $d_1,\dots,d_s$ adalah banyaknya *zero* dan *pole* yang berkaitan.

Definisikan *divisor* dari $f(X)$ sebagai

$$div(f(X)) = e_1[\alpha_1] + e_2[\alpha_2] + \cdots + e_r[\alpha_r] - d_1[\beta_1] - d_2[\beta_2] - \cdots - d_s[\beta_s],$$

singkatnya

$$div(f) = \sum_{P \in E} n_P[P]$$

Koefisien n_P adalah bilangan bulat dan hanya ada berhingga banyaknya yang tidak nol, jadi $div(f)$ adalah jumlah hingga. Fungsi divisor bergantung pada kurva eliptik.

Contoh 2.4. Misalkan E kurva eliptik yang memiliki faktor sebagai berikut

$$X^3 + AX + B = (X - \alpha_1)(X - \alpha_2)(X - \alpha_3).$$

Maka titik-titik $P_1 = (\alpha_1, 0), P_2 = (\alpha_2, 0), P_3 = (\alpha_3, 0)$ adalah titik-titik dengan orde 2 dimana memenuhi $2P_1 = 2P_2 = 2P_3 = \mathcal{O}$. Fungsi $X^3 + AX + B$ bernilai nol pada titik-titik tersebut dan *divisor* nya adalah

$$div(X^3 + AX + B) = [P_1] + [P_2] + [P_3] - 3[\mathcal{O}]$$

Lebih umum lagi, kita definisikan *divisor* pada E sebagai berikut

Definisi 2.16. *Divisor* pada E dinotasikan sebagai

$$D = \sum_{P \in E} n_P[P] \text{ dengan } n_P \in \mathbb{Z} \text{ dan } n_P = 0 \text{ untuk berhingga banyaknya } P.$$

Definisi 2.17. *Derajat dari divisor* adalah jumlah dari koefisiennya

$$deg(D) = deg\left(\sum_{P \in E} n_P[P]\right) = \sum_{P \in E} n_P.$$

Dalam kurva eliptik terdapat dua pemetaan bilinear yaitu, *Weil pairing* dan *Tate pairing*. Tugas akhir ini hanya menggunakan *Weil pairing*.

Weil pairing dinotasikan dengan e_m yang menerima dua titik $P, Q \in E[m]$ sebagai input dan menghasilkan akar kesatuan ke-m $e_m(P, Q)$. Bilinearitas dari *Weil pairing* diekspresikan dengan persamaan

$$e_m(P_1 + P_2, Q) = e_m(P_1, Q)e_m(P_2, Q)$$
$$e_m(P, Q_1 + Q_2) = e_m(P, Q_1)e_m(P, Q_2).$$

Definisi 2.18. Diberikan $P, Q \in E[m]$ yaitu P dan Q adalah titik dengan orde m pada grup E, f_P dan f_Q adalah fungsi rasional pada E yang memenuhi

$$div(f_P) = m[P] - m[\mathcal{O}] \quad \text{dan} \quad div(f_Q) = m[Q] - m[\mathcal{O}]$$

Weil pairing dari P dan Q adalah nilai dari

$$e_m(P, Q) = \frac{f_P(Q + S)}{f_P(S)} \bigg/ \frac{f_Q(P - S)}{f_Q(-S)},$$

dimana $S \in E$ memenuhi $S \notin \{\mathcal{O}, P, -Q, P-Q\}$ untuk memastikan nilai di ruas kanan pada persamaan di atas terdefinisi dan tak nol.

Sifat-sifat dari *Weil pairing*:

- Nilai dari *Weil pairing* memenuhi

$$e_m(P,Q)^m = 1 \text{ untuk setiap } P,Q \in E[m].$$

  Dengan kata lain, $e_m(P,Q)^m$ adalah akar kesatuan ke-m.

- *Weil pairing* bersifat bilinear, dimana

$$e_m(P_1+P_2,Q) = e_m(P_1,Q)e_m(P_2,Q) \text{ untuk setiap } P_1,P_2,Q \in E[m],$$

  dan

$$e_m(P,Q_1+Q_2) = e_m(P,Q_1)e_m(P,Q_2) \text{ untuk setiap } P,Q_1,Q_2 \in E[m].$$

- *Weil pairing* bersifat bergantian (*alternating*), dimana

$$e_m(P,P) = 1 \text{ untuk setiap } P \in E[m].$$

  Ini mengimplikasikan bahwa $e_m(P,Q) = e_m(Q,P)^{-1}$ untuk setiap $P,Q \in E[m]$.

- *Weil pairing* bersifat *nondegenerate*, dimana

$$\text{jika } e_m(P,Q) = 1 \text{ untuk setiap } Q \in E[m], \text{ maka } P = \mathcal{O}.$$

Algoritma menghitung *Weil pairing*

Misalkan E adalah kurva eliptik berkarakteristik $\neq 2,3$, $P = (x_P, y_P)$ dan $Q = (x_Q, y_Q)$ titik-titik tak nol pada E.

(a) Misalkan λ adalah kemiringan garis yang menghubungkan P dan Q, atau kemiringan garis singgung E jika $Q = P$. (Jika λ garis vertikal, maka $\lambda = \infty$). Definisikan fungsi $g_{P,Q}$ sebagai berikut:

$$g_{P,Q}(x,y) = \begin{cases} \dfrac{y - y_P - \lambda(x - x_P)}{x + x_P + x_Q - \lambda^2} & \text{jika } \lambda \neq \infty, \\ x - x_P & \text{jika } \lambda = \infty. \end{cases}$$

Maka

$$div(g_{P,Q}(x,y)) = [P] + [Q] - [P+Q] - [\mathcal{O}].$$

(b) Algoritma *Miller*. Misal $m \geq 1$ dan tulis representasi biner dari m menjadi

$$m = m_0 + m_1 \cdot 2 + \cdots + m_{n-1} 2^{n-1}$$

dengan $m_i \in \{0,1\}$ dan $m_{n-1} \neq 0$. Algoritma berikut menghasilkan fungsi $f_P(x,y)$ dengan *divisor* yang memenuhi

$$div(f_P) = m[P] - [mP] - (m-1)[\mathscr{O}],$$

dimana fungsi $g_{T,T}(x,y)$ dan $g_{T,P}(x,y)$ yang digunakan didefinisikan seperti pada (a).

[1] $T = P$ `dan` $f = 1$
[2] `Loop` $i = n-2$ `sampai dengan` 0
[3] $\quad f = f^2 \cdot g_{T,T}(x,y)$
[4] $\quad T = 2T$
[5] $\quad$ `If` $m_i = 1$ `then`
[6] $\qquad f = f \cdot g_{T,P}(x,y)$
[7] $\qquad T = T + P$
[8] $\quad$ `End If`
[9] `End` i `Loop`
[10] `Return` f

Khususnya, jika $P \in E[m]$, maka $div(f_P) = m[P] - m[\mathscr{O}]$.

Contoh 2.5. Ambil suatu kurva eliptik

$$y^2 = x^3 + 30x + 34 \text{ atas lapangan hingga } \mathbb{F}_{631}$$

Titik-titik

$$P = (36,60) \quad \text{dan} \quad Q = (121,387)$$

merupakan titik-titik dengan orde 5 pada $E(\mathbb{F}_{631})$. Untuk menghitung *Weil pairing*, kita membutuhkan titik S yang tidak berada pada subgrup yang diperluas oleh P dan Q. Ambil $S = (0,36)$. Titik S mempunyai orde 130. Maka algoritma *Miller* menghasilkan

$$\frac{f_P(Q+S)}{f_P(S)} = \frac{103}{219} = 473 \in \mathbb{F}_{631}$$

dan

$$\frac{f_Q(P-S)}{f_Q(-S)} = \frac{284}{204} = 88 \in \mathbb{F}_{631}.$$

Terakhir, dengan membagi kedua nilai di atas didapatkan

$$e_5(P,Q) = \frac{473}{88} = 242 \in \mathbb{F}_{631}$$

Dapat diperiksa bahwa $(242)^5 = 1$, sehingga $e_5(P,Q)$ adalah akar kesatuan ke-5 pada $\mathbb{F}_{631}$.

Kemudian, jika kita ambil $P_1 = (617,5)$ dan $Q_1 = (121,244)$. Perhitungan yang sama akan menghasilkan

$$\frac{f_{P_1}(Q_1+S)}{f_{P_1}(S)} = \frac{326}{523} = 219 \text{ dan } \frac{f_{Q_1}(P_1-S)}{f_{Q_1}(-S)} = \frac{483}{576} = 83.$$

Membagi kedua nilai tersebut dihasilkan

$$e_5(P_1,Q_1) = \frac{219}{83} = 512 \in \mathbb{F}_{631}$$

Ternyata $P_1 = 3P$ dan $Q_1 = 4Q$. Dapat diperiksa bahwa

$$e_5(P,Q)^{12} = 242^{12} = 512 = e_5(P_1,Q_1) = e_5(3P,4Q)$$

mengilustrasikan sifat bilinearitas dari *Weil pairing*. Perhatikan bahwa perhitungan dilakukan pada $\mathbb{F}_{631}$.

II.3.3 Pemetaan Distorsi dan Modifikasi *Weil pairing*

Weil pairing mempunyai satu sifat yaitu *alternating*, dimana $e_m(P,P) = 1$ untuk setiap P. Dalam tugas akhir ini, pemetaan bilinear dilakukan antara titik-titik $P_1 = aP$ dan $P_2 = bP$. Supaya hasil *Weil pairing* antara P_1 dan P_2 tidak selalu menghasilkan nilai 1, digunakan suatu pemetaan $\phi : E \to E$ dimana P dan $\phi(P)$ saling bebas di $E[m]$. Kemudian kita bisa menghitung

$$e_m(P_1,\phi(P_2)) = e_m(aP,\phi(bP)) = e_m(aP,b\phi(P)) = e_m(P,\phi(P))^{ab}.$$

Definisi 2.19. Misalkan $l \geq 3$ prima, E kurva eliptik, $P \in E[l]$ adalah titik dengan orde l, $\phi : E \to E$ pemetaan dari E ke E. Pemetaan ϕ disebut sebagai pemetaan distorsi l untuk P jika memiliki dua sifat berikut.

1. $\phi(nP) = n\phi(P)$ untuk setiap $n \geq 1$.

2. Nilai $e_l(P,\phi(P))$ adalah akar kesatuan primitif ke-l. Artinya

$$e_l(P,\phi(P))^r = 1 \quad \text{jika dan hanya jika} \quad r \text{ adalah kelipatan } l.$$

Definisi 2.20. Misalkan E kurva eliptik, $P \in E[l]$, ϕ adalah pemetaan distorsi l untuk P. *Weil pairing* termodifikasi $\hat{e}_l$ pada $E[l]$ (relatif pada ϕ) didefinisikan sebagai berikut.

$$\hat{e}_l(Q,Q') = e_l(Q,\phi(Q')).$$

Contoh *Weil pairing* termodifikasi dapat dilihat pada Contoh 2.6.

Proposisi 2.1. Misalkan E kurva eliptik

$$E : y^2 = x^3 + x$$

atas lapangan K dan misalkan K memiliki elemen $\alpha \in K$ memenuhi $\alpha^2 = -1$. Definisikan ϕ sebagai

$$\phi(x,y) = (-x,\alpha y) \quad \text{dan} \quad \phi(\mathscr{O}) = (O).$$

(a) Jika $P \in E(K)$, maka $\phi(P) \in E(K)$, sehingga ϕ adalah pemetaan dari $E(K)$ ke dirinya sendiri.

(b) Pemetaan ϕ memenuhi aturan penjumlahan pada E,

$$\phi(P_1+P_2) = \phi(P_1)+\phi(P_2) \quad \text{untuk setiap } P_1,P_2 \in E(K).$$

Khususnya, $\phi(nP) = n\phi(P)$ untuk setiap $P \in E(K)$ dan untuk setiap $n \geq 1$. Bukti lebih lanjut dapat dilihat pada [6].

Contoh 2.6. Ambil $E : y^2 = x^3 + x$ dan bilangan prima $p = 547$. Dengan uji coba didapatkan titik $P_0 = (2,253) \in E(\mathbb{F}_{547})$ dan

$$P = (67,481) = 4P_0 = 4(2,253) \in E(\mathbb{F}_{547})$$

adalah titik dengan orde 137.

Untuk mencari lebih banyak titik dengan orde 137, kita pindah ke lapangan yang lebih besar

$$\mathbb{F}_{547^2} = \mathbb{F}_{547}[x]/\langle x^2+1\rangle = \{a+bi : a,b \in \mathbb{F}_{547}, \quad \text{dimana } i^2 = -1\}.$$

Perhatikan bahwa x^2+1 tidak memiliki akar di F_{547}. Pemetaan distorsi memberikan

$$\phi(P) = (-67,481i) \in E(\mathbb{F}_{547^2}).$$

Untuk menghitung *Weil pairing* dari P dan $\phi(P)$, pilih secara acak titik

$$S = (256+110i,441+15i) \in E(\mathbb{F}_{547^2})$$

dan dengan algoritma *Miller*

$$\frac{f_P(\phi(P)+S)}{f_P(S)} = \frac{376+138i}{384+76i} = 510+96i,$$
$$\frac{f_{\phi(P)}(P-S)}{f_{\phi(P)}(-S)} = \frac{498+286i}{393+120i} = 451+37i.$$

Maka

$$\hat{e}_{137}(P,P) = e_{137}(P,\phi(P)) = \frac{510+96i}{451+37i} = 37+452i \in \mathbb{F}_{547^2}.$$

Dapat diperiksa bahwa $(37+425i)^{137} = 1$, sehingga $\hat{e}_{137}(P,P)$ adalah akar kesatuan primitif ke-137 pada $\mathbb{F}_{547^2}$.

II.4 Interpolasi Lagrange dan *QAP*

II.4.1 Interpolasi Lagrange

Diberikan himpunan pasangan titik (x_j, y_j) dengan $0 \le j \le k$. Polinomial interpolasi *Lagrange* $\mathscr{L}(x)$ mempunyai derajat $\le k$ dan melalui pasangan titik pada himpunan tersebut, $\mathscr{L}(x_j) = y_j$.

Definisi 2.21. Diberikan $k+1$ buah titik $\{x_0, x_1, \ldots, x_k\}$ dimana $x_j \neq x_m$ dengan $j \neq m$, polinomial interpolasi dalam bentuk *Lagrange*

$$\mathscr{L}(x) := \sum_{j=0}^{k} y_j \mathscr{L}_j(x)$$

adalah kombinasi linear dari polinomial-polinomial basis *Lagrange*

$$\mathscr{L}_j(x) := \prod_{\substack{0 \le m \le k \\ m \neq j}} \frac{x - x_m}{x_j - x_m} = \frac{x - x_0}{x_j - x_0} \cdots \frac{x - x_{j-1}}{x_j - x_{j-1}} \frac{x - x_{j+1}}{x_j - x_{j+1}} \cdots \frac{x - x_k}{x_j - x_k}.$$

II.4.2 QAP (*Quadratic Arithmetic Program*)

Quadratic Arithmetic Program *(QAP)* adalah suatu proses yang mentransformasi suatu kode dari fungsi menjadi representasi Matematika yang ketika diberikan input ke dalam kode, akan menghasilkan solusi yang sesuai. Diagram proses dari kode fungsi menjadi *zk-SNARK* dapat dilihat pada Gambar 2. Pada bagian ini kita akan membahas proses dari kode menjadi *QAP*.

$$\begin{array}{c} \textit{Code} \\ \big\downarrow \\ \textit{Algebraic Circuit} \\ \big\downarrow \\ \textit{R1CS} \\ \big\downarrow \\ \textit{QAP} \\ \big\downarrow \\ \textit{zk-SNARKs} \end{array}$$

Gambar 2: Konversi dari kode ke *zk-SNARKs*

II.4.2.1 Kode Fungsi

Misalkan diberikan suatu fungsi $f(x)$, pertama-tama kita akan mengubah fungsi ini menjadi suatu kode dalam bahasa pemrograman. Fungsi yang digunakan adalah fungsi polinomial dengan bahasa pemrogramannya adalah *Python*.

Contoh 2.7.

```
def f(x):
    y = x**3
    return x + y + 5
```

Pada tugas akhir ini, fungsi $f(x)$ yang digunakan berada pada lapangan hingga. Fungsi $x^3 + x + 5$ merupakan fungsi pada lapangan hingga $\mathbb{F}_{11}$.

II.4.2.2 Sirkuit Aljabar

Langkah kedua adalah mengubah kode tadi menjadi serangkaian pernyataan. Proses ini dinamakan *flattening*. Pernyataan yang dihasilkan dapat dibagi menjadi dua bentuk, yaitu:

1. $x = y$, atau
2. $x = y(op)z$

Operasi (op) disini dapat berupa penjumlahan, pengurangan, perkalian atau pembagian. Nilai y dan z dapat berupa variabel atau nilai konstan. Setiap pernyataan setelah melalui proses *flattening* dapat dianggap sebagai *gate* atau pintu dalam sirkuit aljabar.

Misalkan $f(x) = ax^3 + bx^2 + cx + d$ adalah fungsi yang digunakan, maka rangkaian pernyataan dari fungsi tersebut adalah sebagai berikut.

$$\begin{aligned} sym_1 =& \quad x * x \\ sym_2 =& sym_1 * x \\ sym_3 =& \quad a * sym_2 \\ sym_4 =& \quad x * x \\ sym_5 =& sym_4 * x \\ sym_6 =& \quad b * sym_4 \\ sym_7 =& sym_3 + sym_6 \\ sym_8 =& \quad c * x \\ sym_9 =& sym_7 + sym_8 \\ sym_{10} =& sym_9 + d \end{aligned}$$

Pernyataan terakhir biasa disebut sebagai out.

Contoh 2.8. Melanjutkan dari Contoh 2.7, setelah proses *flattening*, didapat pernyataan-pernyataan berikut:

$$\begin{aligned} sym_1 =& \quad x * x \\ y =& sym_1 * x \\ sym_2 =& \quad y + x \\ out =& sym_2 + 5 \end{aligned}$$

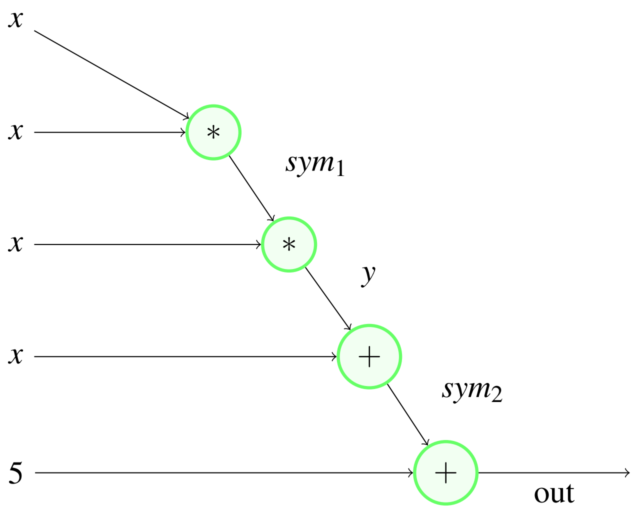
Pernyataan-pernyataan yang dihasilkan dapat direpresentasikan dalam sirkuit aljabar pada Gambar 3.

II.4.2.3 *Rank 1 Constraint System (R1CS)*

Selanjutnya sirkuit aljabar yang sudah didapatkan, diubah menjadi *R1SC*. *R1CS* adalah barisan dari tiga vektor (l, r, o) dan vektor v yang merupakan solusi dari *R1CS* yang memenuhi persamaan berikut

$$l \cdot v * r \cdot v - o \cdot v = 0$$

Notasi l, r, o yang dimaksud disini adalah (*left, right, output*) yang masing-masing merepresentasikan operan kiri, operan kanan, dan *output*. Vektor v disebut sebagai vektor dari nilai-nilai variabel.



Gambar 3: Sirkuit Aljabar

Operasi $*$ adalah perkalian biasa dan operasi $\cdot$ disini adalah perkalian titik antara dua vektor. Definisikan vektor v sebagai berikut

$$[one, x, out, sym_1, sym_2, \dots]$$

dengan masing-masing nilai adalah

1. *one* adalah variabel *dummy*,
2. x adalah solusi dari fungsi $f(x)$,
3. *out* adalah nilai dari $f(x)$ dengan x yang bersesuaian atau nilai pada *gate* terakhir,
4. sym_1, y, sym_2 adalah variabel-variabel dengan nilai pada setiap *gate* kecuali *gate* terakhir.

Variabel one, x, out disebut dengan variabel *input/output*, sedangkan $sym_1, sym_2, \dots$ disebut dengan variabel *intermediary*. Banyaknya *gate* dan variabel *intermediary* dapat berubah bergantung dengan fungsi $f(x)$ yang digunakan.

Kemudian definisikan L, R, O sebagai matriks sebagai berikut

$$L = \begin{bmatrix} l_1 \\ l_2 \\ \dots \\ l_d \end{bmatrix}, R = \begin{bmatrix} r_1 \\ r_2 \\ \dots \\ r_d \end{bmatrix}, O = \begin{bmatrix} o_1 \\ o_2 \\ \dots \\ o_d \end{bmatrix}.$$

dimana d adalah banyaknya *gate* pada sirkuit aljabar.

Selanjutnya definisikan vektor-vektor l_i, r_i, o_i sebagai vektor dengan panjang yang sama dengan vektor v dengan nilai 0 atau 1 sesuai dengan nilai yang muncul pada *gate* ke-i.

Contoh 2.9. Melanjutkan Contoh 2.8, misal $x = 3$ adalah solusi dari fungsi $f(x) = x^3 + x + 5$. Kemudian hitung nilai pada setiap pernyataan.

$$\begin{aligned} sym_1 &= x * x = 3 * 3 = 9 \\ y &= sym_1 * x = 9 * 3 = 27 \\ sym_2 &= y + x = 27 + 3 = 30 \\ out &= sym_2 + 5 = 30 + 5 = 35 \end{aligned}$$

Sehingga didapatkan vektor v sebagai berikut.

$$\begin{aligned} v &= [one, x, out, sym_1, y, sym_2] \\ &= [1, 3, 35, 9, 27, 30] \end{aligned}$$

Selanjutnya untuk mendapatkan matriks L, R, O, kita harus memperhatikan setiap pernyataan. Pada *gate* ke-1 pernyataan berupa

$$\begin{aligned} sym_1 &= x * x \\ sym_1 &= 3 * 3 \\ sym_1 &= 9 \end{aligned}$$

Nilai pada operan kiri adalah 3, nilai pada operan kanan adalah 3, dan nilai pada *output* adalah 9. Kemudian perhatikan vektor v

$$v = [1, 3, 35, 9, 27, 30].$$

Nilai 3 terdapat pada elemen kedua dari v, sehingga vektor l_1 adalah vektor dengan panjang sama dengan v tetapi memiliki nilai 0 untuk semua kecuali elemen keduanya bernilai 1.

$$l_1 = [0, 1, 0, 0, 0, 0]$$

Untuk operan kanan, hal yang sama juga terjadi, sehingga r_1 adalah

$$r_1 = [0, 1, 0, 0, 0, 0]$$

Untuk *output*, nilai 9 ada pada elemen keempat dari v, sehingga vektor o_1 bernilai 0 untuk semua kecuali elemen keempatnya bernilai 1.

$$o_1 = [0, 0, 0, 1, 0, 0]$$

Selanjutnya, untuk *gate* kedua, pernyataan berupa

$$
\begin{aligned}
y &= sym_1 * x \\
y &= 9 * 3 \\
y &= 27
\end{aligned}
$$

Nilai pada operan kiri adalah 9, nilai pada operan kanan adalah 3, dan nilai pada *output* adalah 27. Kemudian perhatikan vektor v

$$v = [1, 3, 35, 9, 27, 30].$$

Nilai 9 terdapat pada elemen keempat dari v, sehingga vektor l_2 adalah vektor dengan panjang sama dengan v tetapi memiliki nilai 0 untuk semua kecuali elemen keempatnya bernilai 1.

$$l_2 = [0, 0, 0, 1, 0, 0]$$

Untuk operan kanan, nilai 3 terdapat pada elemen kedua dari v, sehingga r_2 adalah

$$r_2 = [0, 1, 0, 0, 0, 0]$$

Untuk *output*, nilai 27 ada pada elemen kelima dari v, sehingga vektor o_2 bernilai 0 untuk semua kecuali elemen kelimanya bernilai 1.

$$o_2 = [0, 0, 0, 0, 1, 0]$$

Untuk *gate* ketiga, pernyataan berupa

$$
\begin{aligned}
sym_2 &= y + x \\
sym_2 &= 27 + 3 \\
sym_2 &= 30
\end{aligned}
$$

Berbeda dengan kedua pernyataan sebelumnya, karena operasi pada pernyataan ini berupa penjumlahan. Hal ini ditangani dengan menetapkan kedua nilai pada l dan menetapkan nilai 1 pada r. Sehingga didapatkan nilai pada pernyataan pada operan kiri adalah $y, x = 27, 3$ dan nilai 1 pada operan kanan. Nilai *output* adalah 30. Kemudian perhatikan vektor v

$$v = [1, 3, 35, 9, 27, 30].$$

Nilai $27, 3$ terdapat pada elemen kelima dan kedua dari v, sehingga vektor l_3 adalah vektor dengan panjang sama dengan v tetapi memiliki nilai 0 untuk semua kecuali elemen kedua dan kelimanya bernilai 1.

$$l_3 = [0, 1, 0, 0, 1, 0]$$

Untuk operan kanan, nilai 1 selalu berada pada elemen pertama dari v, sehingga r_3 adalah

$$r_3 = [1,0,0,0,0,0]$$

Untuk *output*, nilai 30 ada pada elemen keenam dari v, sehingga vektor o_3 bernilai 0 untuk semua kecuali elemen keenamnya bernilai 1.

$$o_3 = [0,0,0,0,0,1]$$

Untuk *gate* terakhir, pernyataan berupa

$$\begin{aligned} out &= sym_2 + 5 \\ out &= 30 + 5 \\ out &= 35 \end{aligned}$$

Karena operasi yang dilakukan sama seperti operasi sebelumnya, nilai pada operan kiri adalah $30,5$, nilai pada operan kanan adalah 1, dan nilai pada *output* adalah 35. Sedikit berbeda dengan sebelumnya, penjumlahan yang dilakukan adalah antara satu variabel dengan suatu konstanta. Nilai variabel ada pada vektor v sedangkan nilai konstanta tidak ada, sehingga nilai konstanta tersebut dikalikan dengan elemen pertama 1 pada v. Kemudian perhatikan vektor v

$$v = [1,3,35,9,27,30].$$

Nilai $30,5$ terdapat pada elemen keenam dan elemen pertama dikalikan 5 dari v, sehingga vektor 4 adalah vektor dengan panjang sama dengan v tetapi memiliki nilai 0 untuk semua kecuali elemen keenamnya bernilai 1 dan elemen pertama bernilai 5.

$$l_4 = [5,0,0,0,0,1]$$

Untuk operan kanan, nilai 1 selalu berada pada elemen pertama dari v, sehingga r_4 adalah

$$r_4 = [1,0,0,0,0,0]$$

Untuk *output*, nilai 35 ada pada elemen ketiga *out* dari v, sehingga vektor o_4 bernilai 0 untuk semua kecuali elemen ketiganya bernilai 1.

$$o_4 = [0,0,1,0,0,0]$$

Sehingga didapatkan

$$L = \begin{bmatrix} [0,1,0,0,0,0] \\ [0,0,0,1,0,0] \\ [0,1,0,0,1,0] \\ [5,0,0,0,0,1] \end{bmatrix}, R = \begin{bmatrix} [0,1,0,0,0,0] \\ [0,1,0,0,0,0] \\ [1,0,0,0,0,0] \\ [1,0,0,0,0,0] \end{bmatrix}, O = \begin{bmatrix} [0,0,0,1,0,0] \\ [0,0,0,0,1,0] \\ [0,0,0,0,0,1] \\ [0,0,1,0,0,0] \end{bmatrix}.$$

II.4.2.4 Mengubah *R1CS* menjadi *QAP*

Tahap terakhir pada *QAP* adalah mengubah matriks pada *R1CS* menjadi tiga buah vektor dengan elemennya adalah polinomial-polinomial berderajat $d-1$. Hal ini dilakukan dengan menggunakan interpolasi *Lagrange* dimana pasangan-pasangan titiknya adalah $1,2,\ldots,d$ sebagai nilai x dan nilai- nilai pada satu kolom pada masing-masing matriks sebagai satu polinomial.

$$L(x) = \left\{ L_i(x) = \sum_{j=1}^{d} l_j[i]\mathscr{L}_j(x) : 1 \le i \le n \right\}$$

$$R(x) = \left\{ R_i(x) = \sum_{j=1}^{n} r_j[i]\mathscr{L}_j(x) : 1 \le i \le n \right\}$$

$$O(x) = \left\{ O_i(x) = \sum_{j=1}^{n} o_j[i]\mathscr{L}_j(x) : 1 \le i \le n \right\}$$

dengan n banyaknya kolom matriks L,R,O dan

$$\mathscr{L}_j(x) = \prod_{\substack{1\le k\le d \\ k\ne j}} \frac{x-k}{j-k}$$

Selain ketiga himpunan polinomial tersebut, terdapat satu polinomial yang dihasilkan, yaitu polinomial *target* $t(x)$ dimana polinomial tersebut berbentuk

$$t(x) = \prod_{1\le i\le d} (x-i)$$

Dengan demikian, hasil dari *QAP* adalah tiga matriks $L(x),R(x),O(x)$ dan satu polinomial $t(x)$.

Contoh 2.10. Melanjutkan dari Contoh 2.9, untuk matriks L dilakukan interpolasi *Lagrange* untuk setiap kolomnya. Untuk kolom pertama $[0,0,0,5]^T$,

$$\begin{aligned}
l_1(x) &= \sum_{j=1}^{n} l_j[1]\mathscr{L}_j(x) \\
&= l_1[1]\cdot\mathscr{L}_1(x) + l_2[1]\cdot\mathscr{L}_2(x) + l_3[1]\cdot\mathscr{L}_3(x) + l_4[1]\cdot\mathscr{L}_4(x) \\
&= l_1[1]\cdot\frac{(x-2)(x-3)(x-4)}{(1-2)(1-3)(1-4)} + l_2[1]\cdot\frac{(x-1)(x-3)(x-4)}{(2-1)(2-3)(2-4)} \\
&+ l_3[1]\cdot\frac{(x-1)(x-2)(x-4)}{(3-1)(3-2)(3-4)} + l_4[1]\cdot\frac{(x-1)(x-2)(x-3)}{(4-1)(4-2)(4-3)}
\end{aligned}$$

$$
\begin{aligned}
&= 0 \cdot \frac{(x-2)(x-3)(x-4)}{(1-1)(1-3)(1-4)} + 0 \cdot \frac{(x-1)(x-3)(x-4)}{(2-1)(2-3)(2-4)} \\
&+ 0 \cdot \frac{(x-1)(x-2)(x-4)}{(3-1)(3-2)(3-4)} + 5 \cdot \frac{(x-1)(x-2)(x-3)}{(4-1)(4-2)(4-3)} \\
&= 0 + 0 + 0 + 5 \cdot \frac{x^3 - 6x^2 + 11x - 6}{6} \\
&= 10 \cdot (x^3 + 5x^2 + 5) \\
&= 10x^3 + 50x^2 + 50 \\
&= 10x^3 + 6x^2 + 6
\end{aligned}
$$

Kemudian ubah polinomial tersebut menjadi vektor dengan mengambil nilai koefisiennya, sehingga

$$l_1(x) = [10, 6, 0, 6]$$

Untuk kolom kedua $[1, 0, 1, 0]^T$,

$$
\begin{aligned}
l_2(x) &= \sum_{j=1}^{n} l_j[1] \mathscr{L}_j(x) \\
&= l_1[1] \cdot \mathscr{L}_1(x) + l_2[1] \cdot \mathscr{L}_2(x) + l_3[1] \cdot \mathscr{L}_3(x) + l_4[1] \cdot \mathscr{L}_4(x) \\
&= l_1[1] \cdot \frac{(x-2)(x-3)(x-4)}{(1-2)(1-3)(1-4)} + l_2[1] \cdot \frac{(x-1)(x-3)(x-4)}{(2-1)(2-3)(2-4)} \\
&+ l_3[1] \cdot \frac{(x-1)(x-2)(x-4)}{(3-1)(3-2)(3-4)} + l_4[1] \cdot \frac{(x-1)(x-2)(x-3)}{(4-1)(4-2)(4-3)} \\
&= 1 \cdot \frac{(x-2)(x-3)(x-4)}{(1-1)(1-3)(1-4)} + 0 \cdot \frac{(x-1)(x-3)(x-4)}{(2-1)(2-3)(2-4)} \\
&+ 1 \cdot \frac{(x-1)(x-2)(x-4)}{(3-1)(3-2)(3-4)} + 0 \cdot \frac{(x-1)(x-2)(x-3)}{(4-1)(4-2)(4-3)} \\
&= 1 \cdot \frac{x^3 - 9x^2 + 26x - 24}{-6} + 0 + 1 \cdot \frac{x^3 - 7x^2 + 14x - 8}{-2} + 0 \\
&= 9 \cdot (x^3 + 2x^2 + 4x + 9) + 5 \cdot (x^3 + 4x^2 + 3x + 3) \\
&= (9x^3 + 18x^2 + 36x + 81) + (5x^3 + 20x^2 + 15x + 15) \\
&= 14x^3 + 38x^2 + 51x + 96 \\
&= 3x^3 + 5x^2 + 7x + 8
\end{aligned}
$$

Kemudian ubah polinomial tersebut menjadi vektor dengan mengambil nilai koefisiennya, sehingga

$$l_2(x) = [3, 5, 7, 8]$$

Untuk kolom ketiga $[0,0,0,0]^T$,

$$\begin{aligned}
l_3(x) &= \sum_{j=1}^{n} l_j[1]\mathscr{L}_j(x) \\
&= l_1[1]\cdot\mathscr{L}_1(x) + l_2[1]\cdot\mathscr{L}_2(x) + l_3[1]\cdot\mathscr{L}_3(x) + l_4[1]\cdot\mathscr{L}_4(x) \\
&= l_1[1]\cdot\frac{(x-2)(x-3)(x-4)}{(1-2)(1-3)(1-4)} + l_2[1]\cdot\frac{(x-1)(x-3)(x-4)}{(2-1)(2-3)(2-4)} \\
&+ l_3[1]\cdot\frac{(x-1)(x-2)(x-4)}{(3-1)(3-2)(3-4)} + l_4[1]\cdot\frac{(x-1)(x-2)(x-3)}{(4-1)(4-2)(4-3)} \\
&= 0\cdot\frac{(x-2)(x-3)(x-4)}{(1-1)(1-3)(1-4)} + 0\cdot\frac{(x-1)(x-3)(x-4)}{(2-1)(2-3)(2-4)} \\
&+ 0\cdot\frac{(x-1)(x-2)(x-4)}{(3-1)(3-2)(3-4)} + 0\cdot\frac{(x-1)(x-2)(x-3)}{(4-1)(4-2)(4-3)} \\
&= 0+0+0+0 \\
&= 0
\end{aligned}$$

Kemudian ubah polinomial tersebut menjadi vektor dengan mengambil nilai koefisiennya, sehingga

$$l_3(x) = [0,0,0,0]$$

Untuk kolom keempat $[0,1,0,0]$,

$$\begin{aligned}
l_4(x) &= \sum_{j=1}^{n} l_j[1]\mathscr{L}_j(x) \\
&= l_1[1]\cdot\mathscr{L}_1(x) + l_2[1]\cdot\mathscr{L}_2(x) + l_3[1]\cdot\mathscr{L}_3(x) + l_4[1]\cdot\mathscr{L}_4(x) \\
&= l_1[1]\cdot\frac{(x-2)(x-3)(x-4)}{(1-2)(1-3)(1-4)} + l_2[1]\cdot\frac{(x-1)(x-3)(x-4)}{(2-1)(2-3)(2-4)} \\
&+ l_3[1]\cdot\frac{(x-1)(x-2)(x-4)}{(3-1)(3-2)(3-4)} + l_4[1]\cdot\frac{(x-1)(x-2)(x-3)}{(4-1)(4-2)(4-3)} \\
&= 0\cdot\frac{(x-2)(x-3)(x-4)}{(1-1)(1-3)(1-4)} + 1\cdot\frac{(x-1)(x-3)(x-4)}{(2-1)(2-3)(2-4)} \\
&+ 0\cdot\frac{(x-1)(x-2)(x-4)}{(3-1)(3-2)(3-4)} + 0\cdot\frac{(x-1)(x-2)(x-3)}{(4-1)(4-2)(4-3)} \\
&= 0 + 1\cdot\frac{x^3-8x^2+19x-12}{2} + 0 + 0 \\
&= 6\cdot(x^3-8x^2+19x-12) \\
&= 6x^3-48x^2+114x-72 \\
&= 6x^3+7x^2+4x+5
\end{aligned}$$

Kemudian ubah polinomial tersebut menjadi vektor dengan mengambil nilai koefisiennya, sehingga

$$l_4(x) = [6,7,4,5]$$

Untuk kolom kelima $[0,0,1,0]$,

$$\begin{aligned}
l_5(x) &= \sum_{j=1}^{n} l_j[1]\mathscr{L}_j(x) \\
&= l_1[1]\cdot\mathscr{L}_1(x) + l_2[1]\cdot\mathscr{L}_2(x) + l_3[1]\cdot\mathscr{L}_3(x) + l_4[1]\cdot\mathscr{L}_4(x) \\
&= l_1[1]\cdot\frac{(x-2)(x-3)(x-4)}{(1-2)(1-3)(1-4)} + l_2[1]\cdot\frac{(x-1)(x-3)(x-4)}{(2-1)(2-3)(2-4)} \\
&+ l_3[1]\cdot\frac{(x-1)(x-2)(x-4)}{(3-1)(3-2)(3-4)} + l_4[1]\cdot\frac{(x-1)(x-2)(x-3)}{(4-1)(4-2)(4-3)} \\
&= 0\cdot\frac{(x-2)(x-3)(x-4)}{(1-1)(1-3)(1-4)} + 0\cdot\frac{(x-1)(x-3)(x-4)}{(2-1)(2-3)(2-4)} \\
&+ 1\cdot\frac{(x-1)(x-2)(x-4)}{(3-1)(3-2)(3-4)} + 0\cdot\frac{(x-1)(x-2)(x-3)}{(4-1)(4-2)(4-3)} \\
&= 0+0+1\cdot\frac{x^3-7x^2+14x-8}{-2} + 0 \\
&= 5\cdot(x^3+4x^2+3x+3) \\
&= 5x^3+20x^2+15x+15 \\
&= 5x^3+9x^2+4x+4
\end{aligned}$$

Kemudian ubah polinomial tersebut menjadi vektor dengan mengambil nilai koefisiennya, sehingga

$$l_5(x) = [5,9,4,4]$$

Untuk kolom keenam $[0,0,0,1]$,

$$\begin{aligned}
l_5(x) &= \sum_{j=1}^{n} l_j[1]\mathscr{L}_j(x) \\
&= l_1[1]\cdot\mathscr{L}_1(x) + l_2[1]\cdot\mathscr{L}_2(x) + l_3[1]\cdot\mathscr{L}_3(x) + l_4[1]\cdot\mathscr{L}_4(x) \\
&= l_1[1]\cdot\frac{(x-2)(x-3)(x-4)}{(1-2)(1-3)(1-4)} + l_2[1]\cdot\frac{(x-1)(x-3)(x-4)}{(2-1)(2-3)(2-4)} \\
&+ l_3[1]\cdot\frac{(x-1)(x-2)(x-4)}{(3-1)(3-2)(3-4)} + l_4[1]\cdot\frac{(x-1)(x-2)(x-3)}{(4-1)(4-2)(4-3)} \\
&= 0\cdot\frac{(x-2)(x-3)(x-4)}{(1-1)(1-3)(1-4)} + 0\cdot\frac{(x-1)(x-3)(x-4)}{(2-1)(2-3)(2-4)}
\end{aligned}$$

$$
\begin{aligned}
&+0\cdot\frac{(x-1)(x-2)(x-4)}{(3-1)(3-2)(3-4)}+1\cdot\frac{(x-1)(x-2)(x-3)}{(4-1)(4-2)(4-3)}\\
&=0+0+0+1\cdot\frac{x^3-6x^2+11x-6}{6}\\
&=2\cdot(x^3-6x^2+11x-6)\\
&=2x^3-12x^2+22x-12\\
&=2x^3+10x^2+10
\end{aligned}
$$

Kemudian ubah polinomial tersebut menjadi vektor dengan mengambil nilai koefisiennya, sehingga

$$l_5(x)=[2,10,0,10]$$

Sehingga didapatkan

$$L(x)=\begin{bmatrix}[10,6,0,6]\\ [3,5,7,8]\\ [0,0,0,0]\\ [6,7,4,5]\\ [5,9,4,4]\\ [2,10,0,10]\end{bmatrix}$$

Lakukan hal yang sama untuk matriks R dan O, sehingga didapatkan

$$R(x)=\begin{bmatrix}[7,8,4,3]\\ [4,3,7,9]\\ [0,0,0,0]\\ [0,0,0,0]\\ [0,0,0,0]\\ [0,0,0,0]\end{bmatrix},O(x)=\begin{bmatrix}[0,0,0,0]\\ [0,0,0,0]\\ [2,10,0,10]\\ [9,7,3,4]\\ [6,7,4,5]\\ [5,9,4,4]\end{bmatrix}$$

Perhatikan bahwa semua perhitungan dilakukan pada lapangan hingga $\mathbb{F}_{11}$.

Hal terakhir yang dilakukan adalah menghitung polinomial *target*,

$$
\begin{aligned}
t(x)&=\prod_{1\le i\le d}(x-i)\\
&=(x-1)(x-2)(x-3)(x-4)\\
&=x^4-10x^3+35x^2-50x+24\\
&=x^4+x^3+2x^2+5x+2
\end{aligned}
$$

Ubah polinomial *target* menjadi vektor

$$t(x)=[1,1,2,5,2]$$

Bab III *Zero Knowledge Proof*

III.1 *Zero Knowledge Proof*

Zero knowledge proof adalah suatu metode yang digunakan dimana satu pihak (*prover*) ingin membuktikan kepada pihak lain (*verifier*) bahwa suatu pernyataan benar tanpa ada informasi lain yang keluar selain fakta bahwa pernyataan tersebut benar. Esensi dari *zero knowledge proof* adalah karena sangat mudah untuk membuktikan bahwa suatu pihak memiliki suatu informasi atau rahasia dengan mengungkapkannya, sehingga menjadi tantangan adalah bagaimana membuktikan kepemilikan informasi tersebut tanpa mengungkapkan informasi itu sendiri atau informasi lainnya.

Contoh 3.1. Misalkan terdapat dua orang A dan B. B memiliki kelainan buta warna merah hijau. A memiliki dua buah pena yang berwarna merah dan hijau dan A ingin membuktikan kepada B bahwa kedua pena tersebut memiliki warna yang berbeda tanpa memberitahu B pena mana yang merah dan hijau.

Pertama-tama, A memberikan kedua pena kepada B. B meletakkan kedua pena di belakangnya. Kemudian, B menunjukkan salah satu pena kepada A. B meletakkan pena di belakangnya lagi dan memilih untuk menunjukkan salah satu dari kedua pena. B bertanya "Apakah saya menukar pena?" Jika warna kedua pena memang berbeda, maka A dapat dengan mudah menjawabnya. Sebaliknya, jika A berbohong (kedua pena memiliki warna yang sama), maka A memiliki peluang 50% untuk menjawab dengan benar.

Proses ini dilakukan sebanyak yang diinginkan. Jika proses ini dilakukan sebanyak 10 kali, maka peluang bahwa A berbohong dan dapat menjawab semua pertanyaan dengan benar adalah $1/2^{10} = 1/1024$.

Contoh 3.2. Misalkan *prover* mengetahui suatu polinomial $p(x)$, dimana $p(x) = t(x) \cdot h(x)$ dan $t(x)$ adalah polinomial target. *Prover* ingin membuktikan kepada *verifier* bahwa ia mengetahui polinomial $p(x)$ tanpa memberitahu $p(x)$ itu sendiri. *Prover* dan *verifier* mengetahui $t(x)$ dan hanya *prover* yang mengetahui $p(x)$. *Verifier* mengambil acak suatu nilai r, kemudian menghitung $t(r)$ dan mengirimkan r kepada *prover*. Karena *prover* mengetahui $p(x)$ dan $t(x)$, maka *prover* dapat dengan mudah menghitung

$$h(x) = \frac{p(x)}{t(x)}.$$

Kemudian, *prover* menghitung $h(r)$ dan $p(r)$. Setelah itu mengenkripsi dan mengirimkan $E(h(r)), E(p(r))$ kepada *verifier*. $E(h(r))$ dan $E(p(r))$ ini disebut sebagai *proof* atau bukti dihasilkan oleh *prover* untuk kemudian dikirimkan kepada *verifier* dan diverifikasi oleh *verifier*. Setelah *verifier* menerima *proof* dari *prover*, *verifier* memeriksa

$$E(p(r)) = E(h(r))^{t(r)} g^{p(r)} \qquad = g^{h(r)t(r)} g^{p(r)} = g^{h(r)\cdot t(r)}$$

Jika memang benar bahwa *prover* mengetahui $p(x)$, maka $p(x)$ dapat dibagi habis oleh $t(x)$. Jika tidak, maka $h(x)$ akan memiliki sisa pembagian $s(x)$. Karena r dipilih acak oleh *verifier*, maka kecil kemungkinan bahwa sisa pembagiannya $s(x)$ dapat habis dibagi oleh $t(x)$.

Definisi 3.1. *Zero knowledge proof* memiliki sifat sebagai berikut.

1. ***Completeness***: jika pernyataan benar, maka *verifier* yang jujur (mengikuti proses dengan benar) akan diyakinkan oleh *prover* yang jujur.
2. ***Soundness***: jika pernyataan salah, maka *prover* yang curang (berbohong) tidak akan dapat meyakini *prover* yang jujur.
3. ***Zero-knowledge***: jika pernyataan benar, maka *verifier* tidak mengetahui informasi apapun selain fakta bahwa pernyataan tersebut benar.

III.2 *zk-SNARK*

Protokol *zk-SNARK* merupakan salah satu protokol peningkatan dari *zero knowledge proof*. Berbeda dengan *zero knowledge proof* dimana proses verifikasi bukti dari *prover* yang dilakukan oleh *verifier* dilakukan berulang kali, *zk-SNARK* melakukan verifikasi hanya dalam satu kali interaksi. Selain itu, waktu untuk memverifikasi terjadi dalam waktu yang cepat (sekian milidetik) dan bukti yang dihasilkan oleh *prover* juga tidak panjang.

Definisi 3.2. Selain ketiga sifat pada *zero knowledge proof*, *zk-SNARK* memiliki dua sifat tambahan yaitu

1. ***Succinct***: *proof* atau bukti yang dihasilkan oleh *prover* berukuran kecil (sekian *byte*) dan dapat diverifikasi dengan cepat (hanya sekian milidetik).
2. ***Non-Interactive***: proses menghasilkan *proof*, mengirimkan *proof*, dan memverifikasi *proof* dilakukan hanya sekali.

Dalam tugas akhir ini, yang menjadi rahasia atau informasi yang dimiliki oleh *prover* adalah *witness* w untuk fungsi $f(x)$ yang disinggung pada *QAP*. Hasil dari *QAP* berdasarkan fungsi $f(x)$ yaitu matriks L, R, O dan polinomial *target* $t(x)$ adalah *proof* yang dihasilkan oleh *prover*. *Proof* inilah yang dikirimkan oleh *prover* kepada *verifier* yang kemudian diverifikasi oleh *verifier*. *Verifier* melakukan verifikasi dengan memeriksa

$$\mathbf{L}(x) * \mathbf{R}(x) - \mathbf{O}(x) = h(x) * t(x)$$

dimana $\mathbf{L}(x), \mathbf{R}(x), \mathbf{O}(x)$ masing-masing adalah perkalian titik $L(x) \cdot v, R(x) \cdot v, O(x) \cdot v$.

Tetapi, hanya dengan mengirimkan nilai-nilai tersebut tidak menyembunyikan informasi *prover*. Dengan kata lain, *verifier* masih dapat mengetahui informasi yang dimiliki oleh *prover*. Sehingga tidak *zero-knowledge*. Hal ini dicegah dengan melakukan δ-*shift*, dimana suatu nilai ditambahkan dengan suatu nilai acak δ. Sehingga, bagian *zero-knowledge* dilakukan sebagai berikut.

$$(L(s)+\delta_l) \cdot (R(s)+\delta_r) - (O(s)+\delta_o) = t(s) \cdot (\Delta + h(s))$$

$$\Delta = \frac{L(s)R(s) - O(s) + \delta_r L(s) + \delta_l R(s) + \delta_l \delta_r - \delta_o - t(s)h(s)}{t(s)} \Rightarrow$$

$$\Delta = \frac{\delta_r L(s) + \delta_l R(s) + \delta_l \delta_r - \delta_o}{t(s)}$$

dengan $\delta_o = \delta_l * \delta_r$.

Skema *zk-SNARK* dibagi menjadi tiga tahap: *setup* yang dilakukan oleh pihak ketiga, *proving* yang dilakukan oleh *prover*, *verification* yang dilakukan oleh *verifier*. Bagian *zero-knowledge* diberi warna merah. Pada tugas akhir ini, dilakukan dua skema *zk-SNARKs* dengan dua enkripsi dan juga pemetaan bilinear, yaitu enkripsi homomorfik dengan pemetaan bilinear sederhana dan enkripsi pada titik kurva eliptik dengan *Weil pairing*. Bagian dengan enkripsi homomorfik dengan pemetaan bilinear sederhana diberi warna biru.

1. ***Setup***

   - pilih kurva eliptik E atas lapangan hingga $\mathbb{F}_p$ berkarakteristik $\neq 2, 3$, generator $g \in E(\mathbb{F}_p[m])$ dengan m, p bilangan prima, dan pemetaan bilinear *Weil pairing* dimodifikasi $\hat{e}_m(P, P)$ (e pemetaan bilinear sederhana, g nilai basis)
   - *QAP*: untuk suatu fungsi $f(u) = y$ ubah menjadi tiga buah matriks $L(x), R(x), O(x)$ dengan setiap barisnya adalah koefisien-koefisien dari polinomial berderajat $d-1$ (d adalah jumlah operasi atau pernyataan atau *gate*) dan masing-masing matriks memiliki n buah polinomial dan polinomial $t(x)$ berderajat $d-1$
   - ambil bilangan acak $s, \rho_l, \rho_r, \alpha_l, \alpha_r, \alpha_o, \beta, \gamma$ dari 0 sampai $m-1$
   - hitung $\rho_o = \rho_l \cdot \rho_r \pmod m$ dan $g_l = \rho_l \cdot g, g_r = \rho_r \cdot g, g_o = \rho_o \cdot g$

- tetapkan *proving key*:

$$\Big(\{s^k \cdot g\}_{0\leq k\leq (d-1)}, \{l_i(s)\cdot g_l, r_i(s)\cdot g_r, o_i(s)\cdot g_o\}_{i\in\{m+1,\ldots,n\}},$$
$$\{\alpha_l \cdot l_i(s)\cdot g_l, \alpha_r \cdot r_i(s)\cdot g_r, \alpha_o \cdot o_i(s)\cdot g_o\}_{i\in\{m+1,\ldots,n\}},$$
$$\{\beta \cdot l_i(s)\cdot g_l + \beta \cdot r_i(s)\cdot g_r + \beta \cdot o_i(s)\cdot g_o\}_{i\in\{m+1,\ldots,n\}},$$
$$t(s)\cdot g_l, t(s)\cdot g_r, t(s)\cdot g_o, \alpha_l \cdot t(s)\cdot g_l, \alpha_r \cdot t(s)\cdot g_r, \alpha_o \cdot t(s)\cdot g_o,$$
$$\beta \cdot t(s)\cdot g_l, \beta \cdot t(s)\cdot g_r, \beta \cdot t(s)\cdot g_o\Big)$$

$$\left(\left\{g^{s^k}\right\}_{0\leq k\leq (d-1)}, \left\{g_l^{l_i(s)}, g_r^{r_i(s)}, g_o^{o_i(s)}\right\}_{i\in\{m+1,\ldots,n\}},\right.$$
$$\left\{g_l^{\alpha_l \cdot l_i(s)}, g_r^{\alpha_r \cdot r_i(s)}, g_o^{\alpha_o \cdot o_i(s)}\right\}_{i\in\{m+1,\ldots,n\}},$$
$$\left\{g_l^{\beta \cdot l_i(s)} \cdot g_r^{\beta \cdot r_i(s)} \cdot g_o^{\beta \cdot o_i(s)}\right\}_{i\in\{m+1,\ldots,n\}},$$
$$g_l^{t(s)}, g_r^{t(s)}, g_o^{t(s)}, g_l^{\alpha_l \cdot t(s)}, g_r^{\alpha_r \cdot t(s)}, g_o^{\alpha_o \cdot t(s)},$$
$$\left.g_l^{\beta \cdot t(s)}, g_r^{\beta \cdot t(s)}, g_o^{\beta \cdot t(s)}\right)$$

- tetapkan *verification key*:

$$\Big(g, t(s)\cdot g_o, \{l_i(s)\cdot g_l, r_i(s)\cdot g_r, o_i(s)\cdot g_o\}_{i\in\{1,\ldots,m\}},$$
$$\alpha_l \cdot g, \alpha_r \cdot g, \alpha_o \cdot g, \gamma \cdot g, \beta \cdot \gamma \cdot g\Big)$$

2. ***Proving***

- untuk *input* u, hitung $f(u)$ untuk mendapatkan nilai- nilai variabel $v = v_i, i \in \{1,\ldots,n\}$ untuk setiap variabel *intermediary*
- dengan nilai variabel v yang sudah didapatkan, hitung $\mathbf{L}(x) = \sum_{i=1}^{n} v_i \cdot l_i(x)$ dan sama halnya dengan $\mathbf{R}(x), \mathbf{O}(x)$
- ambil bilangan acak $\delta_l, \delta_r, \delta_o$
- hitung $h(x) = \dfrac{\mathbf{L}(x)\mathbf{R}(x) - \mathbf{O}(x)}{t(x)} + \delta_r \mathbf{L}(x) + \delta_l \mathbf{R}(x) + \delta_l \delta_r t(x) - \delta_o$
- hitung

$$\mathbf{L}_p(s)\cdot g_l = \delta_l \cdot (t(s)\cdot g_l) + \sum_{i=m+1}^{n} v_i \cdot (l_i(s)\cdot g_l)$$

$$g_l^{\mathbf{L}_p(s)} = \left(g_l^{t(s)}\right)^{\delta_l} \cdot \prod_{i=m+1}^{n} \left(g_l^{l_i(s)}\right)^{v_i}$$

dan hal yang sama untuk $\mathbf{R}_p(s)\cdot g_r, \mathbf{O}_p(s)\cdot g_o$ $\left(g_r^{\mathbf{R}_p(s)}, g_o^{\mathbf{O}_p(s)}\right)$.

- hitung α-*shift*

$$\mathbf{L}'_p(s)\cdot g_l = \delta_l\cdot(\alpha_l\cdot t(s)\cdot g_l) + \sum_{i=m+1}^{n} v_i\cdot(\alpha_l\cdot l_i(s)\cdot g_l)$$

$$g_l^{\mathbf{L}'_p(s)} = \left(g_l^{\alpha_l\cdot t(s)}\right)^{\delta_l}\cdot\prod_{i=m+1}^{n}\left(g_l^{\alpha_l\cdot l_i(s)}\right)^{v_i}$$

dan hal yang sama untuk $\mathbf{R}'_p(s)\cdot g_r, \mathbf{O}'_p(s)\cdot g_o$ $\left(g_r^{\mathbf{R}'_p(s)}, g_o^{\mathbf{O}'_p(s)}\right)$.

- hitung

$$\begin{aligned}\mathbf{Z}(s)\cdot g = {} & \delta_l\cdot(\beta\cdot t(s)\cdot g_l) + \delta_r\cdot(\beta\cdot t(s)\cdot g_r) + \delta_o\cdot(\beta\cdot t(s)\cdot g_o)\\ & + \sum_{i=m+1}^{n} v_i\cdot(\beta\cdot l_i(s)\cdot g_l + \beta\cdot r_i(s)\cdot g_r + \beta\cdot o_i(s)\cdot g_o)\end{aligned}$$

$$\begin{aligned}g^{\mathbf{Z}(s)} = {} & \left(g_l^{\beta\cdot t(s)}\right)^{\delta_l}\left(g_r^{\beta\cdot t(s)}\right)^{\delta_r}\left(g_o^{\beta\cdot t(s)}\right)^{\delta_o}\\ & \cdot\prod_{i=m+1}^{n}\left(g_l^{\beta\cdot l_i(s)} g_r^{\beta\cdot r_i(s)} g_o^{\beta\cdot o_i(s)}\right)^{v_i}\end{aligned}$$

- hitung $h(s)\cdot g$ $\left(g^{h(s)}\right)$
- susun *proof*

$$\begin{aligned}\big(&\mathbf{L}_p(s)\cdot g_l, \mathbf{R}_p(s)\cdot g_r, \mathbf{O}_p(s)\cdot g_o, h(s)\cdot g,\\ &\mathbf{L}'_p(s)\cdot g_l, \mathbf{R}'_p(s)\cdot g_r, \mathbf{O}'_p(s)\cdot g_o, \mathbf{Z}(s)\cdot g\big)\end{aligned}$$

$$\begin{aligned}\Big(&g_l^{\mathbf{L}_p(s)}, g_r^{\mathbf{R}_p(s)}, g_o^{\mathbf{O}_p(s)}, g^{h(s)},\\ &g_l^{\mathbf{L}'_p(s)}, g_r^{\mathbf{R}'_p(s)}, g_o^{\mathbf{O}'_p(s)}, g^{\mathbf{Z}(s)}\Big)\end{aligned}$$

3. ***Verification***

- urai *proof* menjadi $(\mathbf{L}_p\cdot g_l, \mathbf{R}_p\cdot g_r, \mathbf{O}_p\cdot g_o, h\cdot g, \mathbf{L}'_p\cdot g_l, \mathbf{R}'_p\cdot g_r, \mathbf{O}'_p\cdot g_o, \mathbf{Z}\cdot g)$ $\left(g_l^{\mathbf{L}_p}, g_r^{\mathbf{R}_p}, g_o^{\mathbf{O}_p}, g^h, g_l^{\mathbf{L}'_p}, g_r^{\mathbf{R}'_p}, g_o^{\mathbf{O}'_p}, g^{\mathbf{Z}}\right)$
- hitung

$$\mathbf{L}_v(s)\cdot g_l = \sum_{i=1}^{m} v_i\cdot(l_i(s)\cdot g_l)$$

$$g_l^{\mathbf{L}_v(s)} = \prod_{i=1}^{m} v_i \left(g_l^{l_i(s)}\right)$$

dan lakukan hal yang sama untuk $\mathbf{R}_v(s) \cdot g_r, \mathbf{O}_v(s) \cdot g_o \left(g_r^{\mathbf{R}_v(s)}, g_o^{\mathbf{O}_v(s)}\right)$.

- Uji pembatasan variabel polinomial:
-
$$\hat{e}(\mathbf{L}_p \cdot g_l, \alpha_l \cdot g) = \hat{e}\left(\mathbf{L}'_p \cdot g_l, g\right)$$
$$e(g_l^{\mathbf{L}_p}, g^{\alpha_l}) = e(g_l^{\mathbf{L}'_p}, g)$$

dan hal yang sama untuk $\mathbf{R}_p \cdot g_r, \mathbf{O}_p \cdot g_o \left(g_r^{\mathbf{R}_p, g_o^{\mathbf{L}_O}}\right)$.

- Uji konsistensi nilai variabel:
$$\hat{e}(\mathbf{L}_p \cdot g_l + \mathbf{R}_p \cdot g_r + \mathbf{O}_p \cdot g_o, \beta \cdot \gamma \cdot g) = \hat{e}(\mathbf{Z} \cdot g, \gamma \cdot g)$$
$$e\left(g_l^{\mathbf{L}_p} \cdot g_r^{\mathbf{R}_p} \cdot g_o^{\mathbf{L}_O}, g^{\beta\gamma}\right) = e\left(g^{\mathbf{Z}}, g^{\gamma}\right)$$

- Uji kevalidan operasi:
$$\begin{aligned}\hat{e}(\mathbf{L}_p \cdot g_l + \mathbf{L}_v(s) \cdot g_l, \mathbf{R}_p \cdot g_r + \mathbf{R}_v(s) \cdot g_r) =& \hat{e}(t(s) \cdot g_o, h \cdot g) \\ & \cdot \hat{e}(\mathbf{O}_p \cdot g_o + \mathbf{O}_v(s) \cdot g_o, g)\end{aligned}$$

$$\begin{aligned}e\left(g_l^{\mathbf{L}_p} \cdot g_l^{\mathbf{L}_v(s)}, g_r^{\mathbf{R}_p} \cdot g_r^{\mathbf{R}_v(s)}\right) =& e\left(g_o^{t(s)}, g^h\right) \\ & \cdot e\left(g_o^{\mathbf{O}_p} \cdot g_o^{\mathbf{O}_v(s)}, g\right)\end{aligned}$$

Contoh 3.3.

1. ***Setup***

   - pilih grup siklik hingga

$$\begin{aligned}(G, \cdot) &= \{2^i (\text{mod } 23) | \forall i \in \mathbb{F}_{11}\} \\ &= \{2^0, 2^1, 2^2, 2^3, 2^4, 2^5, 2^6, 2^7, 2^8, 2^9, 2^{10}\} \\ &= \{1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024\} \\ &= \{1, 2, 4, 8, 16, 9, 18, 13, 3, 6, 12\} \\ &= \{1, 2, 3, 4, 6, 8, 9, 12, 13, 16, 18\}\end{aligned}$$

   - pilih *generator* 2

- pilih pemetaan bilinear

$$e : G \times G \to \mathbb{Z}_{23} : (g,h) \to 2^{log_2(g) \cdot log_2(h)} (\text{mod } 23)$$

$$\begin{aligned} e(3,4) &= 2^{log_2(3) \cdot log_2(4)} (\text{mod } 23) \\ &= 2^{8 \cdot 2} (\text{mod } 23) \\ &= 2^{16} (\text{mod } 23) \\ &= 65536 (\text{mod } 23) \\ &= 9 \end{aligned}$$

- untuk suatu fungsi $f(x) = x^3 + x + 5$ dengan 3 sebagai solusinya dan 35 hasil fungsi, jalankan *QAP* sehingga dihasilkan

$$L(x) = \begin{bmatrix} [10,6,0,6] \\ [3,5,7,8] \\ [0,0,0,0] \\ [6,7,4,5] \\ [5,9,4,4] \\ [2,10,0,10] \end{bmatrix}, R(x) = \begin{bmatrix} [7,8,4,3] \\ [4,3,7,9] \\ [0,0,0,0] \\ [0,0,0,0] \\ [0,0,0,0] \\ [0,0,0,0] \end{bmatrix}, O(x) = \begin{bmatrix} [0,0,0,0] \\ [0,0,0,0] \\ [2,10,0,10] \\ [9,7,3,4] \\ [6,7,4,5] \\ [5,9,44] \end{bmatrix}$$

  dan polinomial *target* $t(x) = [1,1,2,5,2]$. Perhitungan lebih detail dapat dilihat pada contoh bagian *QAP* di bab sebelumnya.

- ambil bilangan acak

$$\begin{aligned} s = 8, \alpha_l &= 5, \\ \rho_l = 6, \alpha_r &= 4, \\ \rho_r = 4, \alpha_o &= 2, \\ \beta = 8, \ \gamma &= 10 \end{aligned}$$

- hitung

$$\begin{aligned} \rho_o =& \rho_l \cdot \rho_r = 6 \cdot 4 = 24 = 2 (\text{mod } 11) \\ g_l =& \quad g^{\rho_l} = \ 2^6 = 64 = 18 (\text{mod } 23) \\ g_r =& \quad g^{\rho_r} = \ 2^4 = 16 = 16 (\text{mod } 23) \\ g_l =& \quad g^{\rho_o} = \ 2^2 = \ 4 = 4 (\text{mod } 23) \end{aligned}$$

$$\rho_o = \rho_l \cdot \rho_r = 6 \cdot 4 = 24 = 2 (\text{mod } 11)$$

- tetapkan *proving key*:

$$
\begin{aligned}
\Big(\{g^{s^k}\}_{0\le k\le(d-1)} =& \{2^{8^0}, 2^{8^1}, 2^{8^2}, 2^{8^3}\},\\
\{g_l^{l_i(s)}\}_{i\in\{m+1,\ldots,n\}} =& \{(18^{8^3})^6\cdot(18^{8^2})^7\cdot(18^{8^1})^4\cdot(18^{8^0})^5,\\
& (18^{8^3})^5\cdot(18^{8^2})^9\cdot(18^{8^1})^4\cdot(18^{8^0})^4,\\
& (18^{8^3})^2\cdot(18^{8^2})^{10}\cdot(18^{8^1})^0\cdot(18^{8^0})^{10}\},\\
\{g_r^{r_i(s)}\}_{i\in\{m+1,\ldots,n\}} =& \{(16^{8^3})^0\cdot(16^{8^2})^0\cdot(16^{8^1})^0\cdot(16^{8^0})^0,\\
& (16^{8^3})^0\cdot(16^{8^2})^0\cdot(16^{8^1})^0\cdot(16^{8^0})^0,\\
& (16^{8^3})^0\cdot(16^{8^2})^0\cdot(16^{8^1})^0\cdot(16^{8^0})^0\},\\
\{g_o^{o_i(s)}\}_{i\in\{m+1,\ldots,n\}} =& \{(4^{8^3})^9\cdot(4^{8^2})^7\cdot(4^{8^1})^3\cdot(4^{8^0})^4,\\
& (4^{8^3})^6\cdot(4^{8^2})^7\cdot(4^{8^1})^4\cdot(4^{8^0})^5,\\
& (4^{8^3})^5\cdot(4^{8^2})^9\cdot(4^{8^1})^4\cdot(4^{8^0})^4,\},\\
\{g_l^{\alpha_l\cdot l_i(s)}\}_{i\in\{m+1,\ldots,n\}} =& \{(18^{5\cdot 8^3})^6\cdot(18^{5\cdot 8^2})^7\cdot(18^{5\cdot 8^1})^4\cdot(18^{5\cdot 8^0})^5,\\
& (18^{5\cdot 8^3})^5\cdot(18^{5\cdot 8^2})^9\cdot(18^{5\cdot 8^1})^4\cdot(18^{5\cdot 8^0})^4,\\
& (18^{5\cdot 8^3})^2\cdot(18^{5\cdot 8^2})^{10}\cdot(18^{5\cdot 8^1})^0\cdot(18^{5\cdot 8^0})^{10}\},\\
\{g_r^{\alpha_r\cdot r_i(s)}\}_{i\in\{m+1,\ldots,n\}} =& \{(16^{4\cdot 8^3})^0\cdot(16^{4\cdot 8^2})^0\cdot(16^{4\cdot 8^1})^0\cdot(16^{4\cdot 8^0})^0,\\
& (16^{4\cdot 8^3})^0\cdot(16^{4\cdot 8^2})^0\cdot(16^{4\cdot 8^1})^0\cdot(16^{4\cdot 8^0})^0,\\
& (16^{4\cdot 8^3})^0\cdot(16^{4\cdot 8^2})^0\cdot(16^{4\cdot 8^1})^0\cdot(16^{4\cdot 8^0})^0\},\\
\{g_o^{\alpha_o\cdot o_i(s)}\}_{i\in\{m+1,\ldots,n\}} =& \{(4^{2\cdot 8^3})^9\cdot(4^{2\cdot 8^2})^7\cdot(4^{2\cdot 8^1})^3\cdot(4^{2\cdot 8^0})^4,\\
& (4^{2\cdot 8^3})^6\cdot(4^{2\cdot 8^2})^7\cdot(4^{2\cdot 8^1})^4\cdot(4^{2\cdot 8^0})^5,\\
& (4^{2\cdot 8^3})^5\cdot(4^{2\cdot 8^2})^9\cdot(4^{2\cdot 8^1})^4\cdot(4^{2\cdot 8^0})^4\},
\end{aligned}
$$

$$
\{g_l^{\beta\cdot l_i(s)}\cdot g_r^{\beta\cdot r_i(s)}\cdot g_o^{\beta\cdot o_i(s)}\}_{i\in\{m+1,\ldots,n\}} =
$$

$$
\begin{aligned}
& \{((18^{8\cdot 8^3})^6\cdot(18^{8\cdot 8^2})^7\cdot(18^{8\cdot 8^1})^4\cdot(18^{8\cdot 8^0})^5)\cdot\\
& ((16^{8\cdot 8^3})^0\cdot(16^{8\cdot 8^2})^0\cdot(16^{8\cdot 8^1})^0\cdot(16^{8\cdot 8^0})^0)\cdot\\
& ((4^{8\cdot 8^3})^9\cdot(4^{8\cdot 8^2})^7\cdot(4^{8\cdot 8^1})^3\cdot(4^{8\cdot 8^0})^4),\\
& ((18^{8\cdot 8^3})^5\cdot(18^{8\cdot 8^2})^9\cdot(18^{8\cdot 8^1})^4\cdot(18^{8\cdot 8^0})^4)\cdot
\end{aligned}
$$

$$
\begin{aligned}
&((16^{8\cdot 8^3})^0\cdot(16^{8\cdot 8^2})^0\cdot(16^{8\cdot 8^1})^0\cdot(16^{8\cdot 8^0})^0)\cdot\\
&((4^{8\cdot 8^3})^6\cdot(4^{8\cdot 8^2})^7\cdot(4^{8\cdot 8^1})^4\cdot(4^{8\cdot 8^0})^5),\\
&((18^{8\cdot 8^3})^2\cdot(18^{8\cdot 8^2})^{10}\cdot(18^{8\cdot 8^1})^0\cdot(18^{8\cdot 8^0})^{10})\cdot\\
&((16^{8\cdot 8^3})^0\cdot(16^{8\cdot 8^2})^0\cdot(16^{8\cdot 8^1})^0\cdot(16^{8\cdot 8^0})^0)\cdot\\
&((4^{8\cdot 8^3})^5\cdot(4^{8\cdot 8^2})^9\cdot(4^{8\cdot 8^1})^4\cdot(4^{8\cdot 8^0})^4)\},\\
g_l^{t(s)}=&(18^{8^4})^1\cdot(18^{8^3})^1\cdot(18^{8^2})^2\cdot(18^{8^1})^5\cdot(18^{8^0})^2,\\
g_r^{t(s)}=&(16^{8^4})^1\cdot(16^{8^3})^1\cdot(16^{8^2})^2\cdot(6^{8^1})^5\cdot(16^{8^0})^2,\\
g_o^{t(s)}=&(4^{8^4})^1\cdot(4^{8^3})^1\cdot(4^{8^2})^2\cdot(4^{8^1})^5\cdot(4^{8^0})^2,\\
g_l^{\alpha_l\cdot t(s)}=&(18^{5\cdot 8^4})^1\cdot(18^{5\cdot 8^3})^1\cdot(18^{5\cdot 8^2})^2\cdot(18^{5\cdot 8^1})^5\cdot(18^{5\cdot 8^0})^2,\\
g_r^{\alpha_r\cdot t(s)}=&(16^{4\cdot 8^4})^1\cdot(16^{4\cdot 8^3})^1\cdot(16^{4\cdot 8^2})^2\cdot(6^{4\cdot 8^1})^5\cdot(16^{4\cdot 8^0})^2,\\
g_o^{\alpha_o\cdot t(s)}=&(4^{2\cdot 8^4})^1\cdot(4^{2\cdot 8^3})^1\cdot(4^{2\cdot 8^2})^2\cdot(4^{2\cdot 8^1})^5\cdot(4^{2\cdot 8^0})^2,\\
g_l^{\beta\cdot t(s)}=&(18^{8\cdot 8^4})^1\cdot(18^{8\cdot 8^3})^1\cdot(18^{8\cdot 8^2})^2\cdot(18^{8\cdot 8^1})^5\cdot(18^{8\cdot 8^0})^2,\\
g_r^{\beta\cdot t(s)}=&(16^{8\cdot 8^4})^1\cdot(16^{8\cdot 8^3})^1\cdot(16^{8\cdot 8^2})^2\cdot(6^{8\cdot 8^1})^5\cdot(16^{8\cdot 8^0})^2,\\
g_o^{\beta\cdot t(s)}=&(4^{8\cdot 8^4})^1\cdot(4^{8\cdot 8^3})^1\cdot(4^{8\cdot 8^2})^2\cdot(4^{8\cdot 8^1})^5\cdot(4^{8\cdot 8^0})^2\Big)
\end{aligned}
$$

$$
\begin{aligned}
=\Big(&\{g^{s^k}\}_{0\le k\le 3}=\{2,3,6,18\},\\
&\{g_l^{l_i(s)}\}_{i\in\{4,\dots,6\}}=\{4,4,2\},\\
&\{g_r^{r_i(s)}\}_{i\in\{4,\dots,6\}}=\{1,1,1\},\\
&\{g_o^{o_i(s)}\}_{i\in\{4,\dots,6\}}=\{16,3,3\},\\
&\{g_l^{\alpha_l\cdot l_i(s)}\}_{i\in\{4,\dots,6\}}=\{12,12,9\},\\
&\{g_r^{\alpha_r\cdot r_i(s)}\}_{i\in\{4,\dots,6\}}=\{1,1,1\},\\
&\{g_o^{\alpha_o\cdot o_i(s)}\}_{i\in\{4,\dots,6\}}=\{3,9,9\},\\
\{g_l^{\beta\cdot l_i(s)}\cdot g_r^{\beta\cdot r_i(s)}\cdot &g_o^{\beta\cdot o_i(s)}\}_{i\in\{4,\dots,6\}}=\{16,8,18\}\\
g_l^{t(s)}=4,\ &g_r^{t(s)}=9,g_o^{t(s)}=3,\\
g_l^{\alpha_l\cdot t(s)}=12,\ &g_r^{\alpha_r\cdot t(s)}=6,g_o^{\alpha_o\cdot t(s)}=9,\\
g_l^{\beta\cdot t(s)}=9,\ &g_r^{\beta\cdot t(s)}=13,g_o^{\beta\cdot t(s)}=6\Big)
\end{aligned}
$$

- tetapkan *verification key*:

$$\begin{aligned}\Big(g^1 =&2,\\ g_o^{t(s)} =&(4^{8^4})^1\cdot(4^{8^3})^1\cdot(4^{8^2})^2\cdot(4^{8^1})^5\cdot(4^{8^0})^2,\\ \{g_l^{l_i(s)}\}_{i\in\{1,\dots,m\}} =&\{(18^{8^3})^{10}\cdot(18^{8^2})^6\cdot(18^{8^1})^0\cdot(18^{8^0})^6,\\ &(18^{8^3})^3\cdot(18^{8^2})^5\cdot(18^{8^1})^7\cdot(18^{8^0})^8,\\ &(18^{8^3})^0\cdot(18^{8^2})^0\cdot(18^{8^1})^0\cdot(18^{8^0})^0\},\\ \{g_r^{r_i(s)}\}_{i\in\{1,\dots,m\}} =&\{(16^{8^3})^7\cdot(16^{8^2})^8\cdot(16^{8^1})^4\cdot(16^{8^0})^3,\\ &(16^{8^3})^4\cdot(16^{8^2})^3\cdot(16^{8^1})^7\cdot(16^{8^0})^9,\\ &(16^{8^3})^0\cdot(16^{8^2})^0\cdot(16^{8^1})^0\cdot(16^{8^0})^0\},\\ \{g_o^{o_i(s)}\}_{i\in\{1,\dots,m\}} =&\{(4^{8^3})^0\cdot(4^{8^2})^0\cdot(4^{8^1})^0\cdot(4^{8^0})^0,\\ &(4^{8^3})^0\cdot(4^{8^2})^0\cdot(4^{8^1})^0\cdot(4^{8^0})^0,\\ &(4^{8^3})^2\cdot(4^{8^2})^{10}\cdot(4^{8^1})^0\cdot(4^{8^0})^{10},\},\\ g^{\alpha_l} = 2^5, g^{\alpha_r} =&2^4, g^{\alpha_o} = 2^2\\ g^{\gamma} = 2^{10}, g^{\beta\gamma} =& 2^{8\cdot 10}\Big)\end{aligned}$$

$$\begin{aligned}=\Big(g^1 = 2, g_o^{t(s)} =& 3,\\ \{g_l^{l_i(s)}\}_{i\in\{1,\dots,3\}} =&\{9,8,1\},\\ \{g_r^{r_i(s)}\}_{i\in\{1,\dots,3\}} =&\{4,4,1\},\\ \{g_o^{o_i(s)}\}_{i\in\{1,\dots,3\}} =&\{1,1,16\},\\ g^{\alpha_l} = 9, g^{\alpha_r} =&16, g^{\alpha_o} = 4\\ g^{\gamma} = 12, g^{\beta\gamma} =& 8\Big)\end{aligned}$$

2. ***Proving***

- untuk *witness* 3, didapatkan nilai-nilai variabel

$$v = [one, x, out, sym_1, y, sym_2] = [1, 3, 35, 9, 27, 30]$$

- hitung

$$\mathbf{L}(x) = 1\cdot(10x^3+6x^2+6)+3\cdot(3x^3+5x^2+7x+8)+$$

$$\begin{aligned}
& 35 \cdot (0) + 9 \cdot (6x^3 + 7x^2 + 4x + 5) + \\
& 27 \cdot (5x^3 + 9x^2 + 4x + 4) + 30 \cdot (2x^3 + 10x^2 + 10) \\
=& (10x^3 + 6x^2 + 6) + (9x^3 + 15x^2 + 21x + 24) + \\
& (0) + (54x^3 + 63x^2 + 36x + 45) + \\
& (54x^3 + 45x^2 + 20x + 20) + (16x^3 + 80x^2 + 80) \\
=& 114x^3 + 209x^2 + 77x + 175 \\
=& 4x^3 + 10 \\
=& [4, 0, 0, 10], \\
\mathbf{R}(x) =& 1 \cdot (7x^3 + 8x^2 + 4x + 3) + 3 \cdot (4x^3 + 3x^2 + 7x + 9) + \\
& 35 \cdot (0) + 9 \cdot (0) + 27 \cdot (0) + 30 \cdot (0) \\
=& (7x^3 + 8x^2 + 4x + 3) + (12x^3 + 9x^2 + 21x + 27) + 0 \\
=& 19x^3 + 17x^2 + 25x + 30 \\
=& 8x^3 + 6x^2 + 3x + 8 \\
=& [8, 6, 3, 8] \\
\mathbf{O}(x) =& 1 \cdot (0) + 3 \cdot (0) + \\
& 35 \cdot (2x^3 + 10x^2 + 10) + 9 \cdot (9x^3 + 7x^2 + 3x + 4) + \\
& 27 \cdot (6x^3 + 7x^2 + 4x + 5) + 30 \cdot (5x^3 + 9x^2 + 4x + 4) \\
=& (0) + (0) + \\
& (4x^3 + 20x^2 + 20) + (81x^3 + 63x^2 + 27x + 36) + \\
& (30x^3 + 35x^2 + 20x + 25) + (40x^3 + 72x^2 + 32x + 32) \\
=& 155x^3 + 190x^2 + 79x + 113 \\
=& x^3 + 3x^2 + 2x + 3 \\
=& [1, 3, 2, 3]
\end{aligned}$$

- ambil bilangan acak

$$\delta_l = 8, \delta_r = 6, \delta_o = 4$$

- hitung

$$\begin{aligned}
h(x) =& \frac{(4x^3 + 10) \cdot (8x^3 + 6x^2 + 3x + 8) - (x^3 + 3x^2 + 2x + 3)}{x^4 + x^3 + 2x^2 + 5x + 2} + \\
& 6 \cdot (4x^3 + 10) + 8 \cdot (8x^3 + 6x^2 + 3x + 8) + \\
& 8 \cdot 6 \cdot (x^4 + x^3 + 2x^2 + 5x + 2) - 2
\end{aligned}$$

$$
\begin{aligned}
=&\frac{(10x^6+2x^5+x^4+2x^3+5x^2+8x+3)-(x^3+3x^2+2x+3)}{x^4+x^3+2x^2+5x+2}+\\
&(2x^3+5)+(9x^3+4x^2+2x+9)+\\
&(4x^4+4x^3+8x^2+9x+8)-4\\
=&(10x^3+3x)+(2x^3+5)+(9x^3+4x^2+2x+9)+\\
&(4x^4+4x^3+8x^2+9x+8)-4\\
=&(2x^3+10x^2+3x+5)+(4x^4+2x^3+x^2+17)-4\\
=&4x^4+4x^3+3x-4\\
=&4x^4+4x^3+3x+7\\
=&[4,4,0,3,7]
\end{aligned}
$$

- hitung

$$
\begin{aligned}
g_l^{\mathbf{L}_p(s)} =&\left(g_l^{t(s)}\right)^{\delta_l}\cdot\prod_{i=4}^{6}\left(g_l^{l_i(s)}\right)^{v_i}\\
=&4^8\cdot 4^9\cdot 4^{27}\cdot 2^{30}\\
=&9\cdot 13\cdot 12\cdot 3\\
=&9\cdot 8\\
=&3
\end{aligned}
$$

$$
\begin{aligned}
g_r^{\mathbf{R}_p(s)} =&\left(g_r^{t(s)}\right)^{\delta_r}\cdot\prod_{i=4}^{6}\left(g_r^{r_i(s)}\right)^{v_i}\\
=&9^6\cdot 1^9\cdot 1^{27}\cdot 1^{30}\\
=&3\cdot 1\cdot 1\cdot 1\\
=&3\cdot 1\\
=&3
\end{aligned}
$$

$$
\begin{aligned}
g_o^{\mathbf{O}_p(s)} =&\left(g_o^{t(s)}\right)^{\delta_o}\cdot\prod_{i=4}^{6}\left(g_o^{o_i(s)}\right)^{v_i}\\
=&3^4\cdot 16^9\cdot 3^{27}\cdot 3^{30}\\
=&12\cdot 8\cdot 13\cdot 6\\
=&12\cdot 3\\
=&13
\end{aligned}
$$

- hitung α-*shift*

$$
g_l^{\mathbf{L}'_p(s)} =\left(g_l^{\alpha_l\cdot t(s)}\right)^{\delta_l}\cdot\prod_{i=4}^{6}\left(g_l^{\alpha_l\cdot l_i(s)}\right)^{v_i}
$$

$$
\begin{aligned}
&= 12^{8} \cdot 12^{9} \cdot 12^{27} \cdot 9^{30} \\
&= 8 \cdot 4 \cdot 18 \cdot 13 \\
&= 8 \cdot 16 \\
&= 13
\end{aligned}
$$

$$
\begin{aligned}
g_r^{\mathbf{R}'_p(s)} &= \left(g_r^{\alpha_r \cdot t(s)}\right)^{\delta_r} \cdot \prod_{i=4}^{6} \left(g_r^{\alpha_l rcdot r_i(s)}\right)^{v_i} \\
&= 6^{6} \cdot 1^{9} \cdot 1^{27} \cdot 1^{30} \\
&= 12 \cdot 1 \cdot 1 \cdot 1 \\
&= 12 \cdot 1 \\
&= 12
\end{aligned}
$$

$$
\begin{aligned}
g_o^{\mathbf{O}'_p(s)} &= \left(g_o^{\alpha_o \cdot t(s)}\right)^{\delta_o} \cdot \prod_{i=4}^{6} \left(g_o^{\alpha_o \cdot o_i(s)}\right)^{v_i} \\
&= 9^{4} \cdot 3^{9} \cdot 9^{27} \cdot 9^{30} \\
&= 6 \cdot 18 \cdot 8 \cdot 13 \\
&= 6 \cdot 9 \\
&= 8
\end{aligned}
$$

- hitung

$$
\begin{aligned}
g^{\mathbf{Z}(s)} &= \left(g_l^{\beta \cdot t(s)}\right)^{\delta_l} \left(g_r^{\beta \cdot t(s)}\right)^{\delta_r} \left(g_o^{\beta \cdot t(s)}\right)^{\delta_o} \cdot \\
&\prod_{i=4}^{6} \left(g_l^{\beta \cdot l_i(s)} g_r^{\beta \cdot r_i(s)} g_o^{\beta \cdot o_i(s)}\right)^{v_i} \\
&= 9^{8} \cdot 13^{6} \cdot 6^{4} \cdot 16^{9} \cdot 8^{5} \cdot 18^{8} \\
&= 13 \cdot 6 \cdot 8 \cdot 8 \cdot 16 \cdot 16 \\
&= 3 \cdot 1 \\
&= 3
\end{aligned}
$$

- hitung

$$
\begin{aligned}
g^{h(s)} &= (2^{8^4})^4 \cdot (2^{8^3})^4 \cdot (2^{8^2})^0 \cdot (2^{8^1})^3 \cdot (2^{8^0})^7 \\
&= 16^4 \cdot 18^4 \cdot 6^0 \cdot 3^3 \cdot 2^7 \\
&= 9 \cdot 4 \cdot 1 \cdot 4 \cdot 13 \\
&= 9
\end{aligned}
$$

- susun *proof*

$$\Big(g_l^{\mathbf{L}_p(s)} = 3, g_r^{\mathbf{R}_p(s)} = 3,$$
$$g_o^{\mathbf{O}_p(s)} = 13, g_l^{\mathbf{L}'_p(s)} = 13,$$
$$g_r^{\mathbf{R}'_p(s)} = 12, g_o^{\mathbf{O}'_p(s)} = 8,$$
$$g^{\mathbf{Z}(s)} = 3, g^{h(s)} = 9\Big)$$

3. ***Verification***

- urai *proof* menjadi

$$\Big(g_l^{\mathbf{L}_p} = 3, g_r^{\mathbf{R}_p} = 3,$$
$$g_o^{\mathbf{O}_p} = 13, g_l^{\mathbf{L}'_p} = 13,$$
$$g_r^{\mathbf{R}'_p} = 12, g_o^{\mathbf{O}'_p} = 8,$$
$$g^{\mathbf{Z}} = 3, g^{h} = 9\Big)$$

- hitung

$$\begin{aligned}
g_l^{\mathbf{L}_v(s)} =& \prod_{i=1}^{3} v_i \cdot (g_l^{l_i(s)}) \\
=& 9^1 \cdot 8^3 \cdot 1^{35} \\
=& 9 \cdot 6 \cdot 1 \\
=& 8 \\
g_r^{\mathbf{R}_v(s)} =& \prod_{i=1}^{3} v_i \cdot (g_r^{r_i(s)}) \\
=& 4^1 \cdot 4^3 \cdot 1^{35} \\
=& 4 \cdot 18 \cdot 1 \\
=& 3 \\
g_o^{\mathbf{O}_v(s)} =& \prod_{i=1}^{3} v_i \cdot (g_l^{o_i(s)}) \\
=& 1^1 \cdot 1^3 \cdot 16^{35} \\
=& 1 \cdot 1 \cdot 3 \\
=& 3
\end{aligned}$$

- uji pembatasan variabel polinomial:

$$
\begin{aligned}
e(g_l^{\mathbf{L}_p}, g^{\alpha_l}) =& e(g_l^{\mathbf{L}_p}, g) \\
e(3,9) =& e(13,2) \\
2^{log_2(3) \cdot log_2(9)} =& 2^{log_2(13) \cdot log_2(2)} \\
2^{8 \cdot 5} =& 2^{7 \cdot 1} \\
2^{40} =& 2^{7} \\
2^{7} =& 2^{7} \\
128 =& 128 \\
13 =& 13 e(g_r^{\mathbf{R}_p}, g^{\alpha_r}) = e(g_l^{\mathbf{L}_p}, g) \\
e(3,16) =& e(12,2) \\
2^{log_2(3) \cdot log_2(16)} =& 2^{log_2(12) \cdot log_2(2)} \\
2^{8 \cdot 4} =& 2^{10 \cdot 1} \\
2^{32} =& 2^{10} \\
2^{10} =& 2^{10} \\
1024 =& 1024 \\
12 =& 12 \\
e(g_o^{\mathbf{O}_p}, g^{\alpha_o}) =& e(g_o^{\mathbf{L}_p}, g) \\
e(13,4) =& e(8,2) \\
2^{log_2(13) \cdot log_2(4)} =& 2^{log_2(8) \cdot log_2(2)} \\
2^{7 \cdot 2} =& 2^{3 \cdot 1} \\
2^{14} =& 2^{3} \\
2^{3} =& 2^{3} \\
8 =& 8
\end{aligned}
$$

- uji konsistensi nilai variabel:

$$
\begin{aligned}
e\left(g_l^{\mathbf{L}_p} \cdot g_r^{\mathbf{R}_p} \cdot g_o^{\mathbf{L}_O}, g^{\beta\gamma}\right) =& e\left(g^{\mathbf{Z}}, g^{\gamma}\right) \\
e(3 \cdot 3 \cdot 13, 8) =& e(3,12) \\
e(2,8) =& e(3,12) \\
2^{log_2(2) \cdot log_2(8)} =& 2^{log_2(3) \cdot log_2(12)} \\
2^{1 \cdot 3} =& 2^{8 \cdot 10} \\
2^{3} =& 2^{80}
\end{aligned}
$$

$$
\begin{aligned}
2^3 =& 2^3 \\
8 =& 8
\end{aligned}
$$

- uji kevalidan operasi:

$$
\begin{aligned}
e\left(g_l^{\mathbf{L}_p} \cdot g_l^{\mathbf{L}_v(s)}, g_r^{\mathbf{R}_p} \cdot g_r^{\mathbf{R}_v(s)}\right) =& e\left(g_o^{t(s)}, g^h\right) \\
e(3 \cdot 8, 3 \cdot 3) =& e(3,9) \cdot e(13 \cdot 3, 2) \\
e(1,9) =& e(3,9) \cdot e(16,2) \\
2^{log_2(1) \cdot log_2(9)} =& 2^{log_2(3) \cdot log_2(9)} \cdot 2^{log_2(16) \cdot log_2(2)} \\
2^{0 \cdot 5} =& 2^{8 \cdot 5} \cdot 2^{4 \cdot 1} \\
2^0 =& 2^{40} \cdot 2^4 \\
1 =& 2^7 \cdot 2^4 \\
1 =& 128 \cdot 16 \\
1 =& 2048 \\
1 =& 1
\end{aligned}
$$

Bab IV Algoritma *zk-SNARK*

Pada bab ini, diberikan algoritma yang digunakan dalam *zk-SNARK* termasuk kurva eliptik. Semua algoritma ditulis menggunakan *pseudocode*.

IV.1 Operasi di Lapangan Hingga

Beberapa operasi di $\mathbb{F}_p$ yang digunakan pada tugas akhir ini: penjumlahan, perkalian, faktor sekutu terbesar, invers, pembagian, pemangkatan dan akar kuadrat. Selain itu, juga dibahas operasi perkalian dan pembagian untuk polinomial atas lapangan hingga $\mathbb{F}_p[x]$ dan gelanggang kelas residu $\mathbb{F}_p[x]/\langle f(x)\rangle$, juga dibahas mengenai akar kuadrat di gelanggang kelas residu $\mathbb{F}_p[x]/\langle f(x)\rangle$ untuk $f(x)$ polinomial tak tereduksi di $\mathbb{F}_p[x]$.

IV.1.1 Penjumlahan di $\mathbb{F}_p$

Penjumlahan di $\mathbb{F}_p$ cukup sederhana, jadi pada program tidak akan diberi fungsi khusus.

Algoritma IV.1 Penjumlahan di $\mathbb{F}_p$

Input: $a, b \in \mathbb{F}_p, p$ bilangan prima
Ensure: $(a+b) \pmod p$
1: **return** $(a+b) \bmod p$

IV.1.2 Perkalian di $\mathbb{F}_p$

Sama seperti penjumlahan, perkalian di $\mathbb{F}_p$ cukup sederhana dan tidak akan dibuat buat fungsi khusus dalam program.

Algoritma IV.2 Perkalian di $\mathbb{F}_p$

Input: $a, b \in \mathbb{F}_p, p$ bilangan prima
Ensure: $(a \cdot b) \pmod p$
1: **return** $(a \cdot b) \bmod p$

IV.1.3 Algoritma Euclid yang Diperluas

Seperti namanya, algoritma *Euclid* yang diperluas adalah perluasan dari algoritma *Euclid*. Algoritma *Euclid* digunakan untuk mencari faktor pembagi terbesar $\gcd(a,b)$ dari a dan b.

Algoritma IV.3 Algoritma *Euclid*

Input: $a, b \in \mathbb{F}_p$, p bilangan prima
Ensure: $\gcd(a,b)$

$r_0 \leftarrow a$
$r_1 \leftarrow b$
$m \leftarrow 1$
while $r_m \neq 0$ **do**
  $q_m \leftarrow \lfloor \frac{r_{m-1}}{r_m} \rfloor$
  $r_{m+1} \leftarrow r_{m-1} - q_m r_m$
  $m \leftarrow m+1$
end while
$m \leftarrow m-1$
return r_m

Algoritma *Euclid* yang diperluas tidak hanya untuk menghitung $\gcd(a,b)$, tetapi juga dapat menghasilkan s dan t sehingga $\gcd(a,b) = sa + tb$.

Algoritma IV.4 Algoritma *Euclid* yang Diperluas

Input: $a, b \in \mathbb{F}_p, p$ bilangan prima
Ensure: $\gcd(a,b)$, s (koefisien a), t (koefisien b)

$a_0 \leftarrow a$
$b_0 \leftarrow b$
$t_0 \leftarrow 0$
$t \leftarrow 1$
$s_0 \leftarrow 1$
$s \leftarrow 0$
$q \leftarrow \lfloor \frac{a_0}{b_0} \rfloor$
$r \leftarrow a_0 - qb_0$
while $r > 0$ **do**
    $temp \leftarrow t_0 - qt$
    $t_0 \leftarrow t$
    $t \leftarrow temp$
    $temp \leftarrow s_0 - qs$
    $s_0 \leftarrow s$
    $s \leftarrow temp$
    $a_0 \leftarrow b_0$
    $b_0 \leftarrow r$
    $q \leftarrow \lfloor \frac{a_0}{b_0} \rfloor$
    $r \leftarrow a_0 - qb_0$
end while
$r \leftarrow b_0$
return r, s, t

IV.1.4 Invers di $\mathbb{F}_p$

Algoritma yang digunakan pada invers di $\mathbb{F}_p$ adalah algoritma *Euclid* yang diperluas. Hanya saja, $\gcd(a,b) = 1$ dan $a^{-1} (\bmod b) = s (\bmod b)$.

IV.1.5 Pembagian di $\mathbb{F}_p$

Pembagian di $\mathbb{F}_p$ sama saja dengan mengalikan suatu bilangan dengan invers dari bilangan kedua. Misal $a, b \in \mathbb{F}_p$, maka $a/b = a \cdot b^{-1}$.

Algoritma IV.5 Algoritma pembagian di $\mathbb{F}_p$

Input: $a, b \in \mathbb{F}_p, p$ bilangan prima
Ensure: $a/b \pmod p$
return $a \cdot b^{-1} \pmod p$

IV.1.6 Pemangkatan di $\mathbb{F}_p$

Pemangkatan di $\mathbb{F}_p$ dilakukan dengan menggunakan metode kuadrat dan kali (*square and multiply*).

Algoritma IV.6 Algoritma Pemangkatan di $\mathbb{F}_p$ dengan Metode Kuadrat dan Kali

Input: $a \in \mathbb{F}_p, p$ bilangan prima, exponen n
Ensure: $a^n \pmod p$
$temp \leftarrow 1$
while $y > 0$ **do**
  if $y \equiv 1 \bmod 2$ **then**
    $temp \leftarrow temp \cdot x(\bmod p)$
  end if
  $n \leftarrow \lfloor n/2 \rfloor$
  $a \leftarrow a \cdot a(\bmod p)$
end while
return $temp \pmod p$

IV.1.7 Akar Kuadrat di $\mathbb{F}_p$

Akar kuadrat di $\mathbb{F}_p$ dilakukan dengan menggunakan algoritma *Tonelli-Shanks*. Sebelum masuk ke algoritma *Tonelli-Shanks*, pertama-tama kita harus mengetahui mengenai simbol *Jacobi* terlebih dahulu. Simbol *Jacobi* digunakan untuk mengetahui apakah a merupakan residu kuadratik modulo p, artinya terdapat suatu bilangan $x \in \mathbb{F}_p$ dimana $x^2 \equiv a \pmod p$. Jika tidak terdapat x yang memenuhi, maka a dikatakan non-residu kuadratik modulo p.

Definisi 4.1. Untuk suatu bilangan bulat a dan bilangan bulat ganjil n, simbol *Jacobi* (a/n) didefinisikan sebagai perkalian dari simbol *Legendre* berdasarkan faktor-faktor prima dari n.

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)^{\alpha_1} \left(\frac{a}{p_2}\right)^{\alpha_2} \cdots \left(\frac{a}{p_k}\right)^{\alpha_k},$$

dimana

$$n = p_1^{\alpha_1} p_2^{\alpha_2} \dots p_k^{\alpha_k}$$

adalah faktorisasi prima dari n.

Simbol *Legendre* (a/p) terdefinisi untuk bilangan bulat a dan semua bilangan prima ganjil p sebagai berikut

$$\left(\frac{a}{p}\right) = \begin{cases} 0 & \text{jika } a \equiv 0 (\text{mod } p), \\ 1 & \text{jika } a \not\equiv 0 (\text{mod } p) \text{ dan untuk beberapa bilangan bulat } x : a \equiv x^2 \pmod{p}, \\ -1 & \text{jika } a \not\equiv 0 (\text{mod } p) \text{ dan tidak ada } x \text{ demikian.} \end{cases}$$

Beberapa cara yang digunakan untuk mengevaluasi nilai dari simbol *Jacobi*.

1. Jika n ganjil dan $m_1 \equiv m_2 \pmod{p}$ maka

$$\left(\frac{m_1}{n}\right) = \left(\frac{m_1}{m_2}\right)$$

2. Jika n ganjil maka

$$\left(\frac{2}{n}\right) = \begin{cases} 1 & \text{jika } n \equiv \pm 1 \pmod{8} \\ -1 & \text{jika } n \equiv \pm 3 \pmod{8} \end{cases}$$

3. Jika n ganjil maka

$$\left(\frac{m_1 m_2}{n}\right) = \left(\frac{m_1}{n}\right)\left(\frac{m_2}{n}\right)$$

4. Jika m dan n ganjil maka

$$\left(\frac{m}{n}\right) = \begin{cases} -\left(\frac{n}{m}\right) & \text{jika } m \equiv n \equiv 3 \pmod{4} \\ \left(\frac{n}{m}\right) & \text{lainnya.} \end{cases}$$

Algoritma IV.7 Algoritma Simbol *Jacobi*

Input: a, n
Ensure: 1 (a residu kuadratik modulo n) atau -1 (a non-residu kuadratik modulo p)

```
temp ← 1
while a ≠ 0 do
    while a ≡ 0 (mod 2) do
        a ← a/2
        if n ≡ 3 (mod 8) atau n ≡ 5 (mod 8) then
            temp ← −temp
        end if
    end while
    swap = a
    a = p
    p = swap
    if a ≡ 3 (mod 4) dan n ≡ 3 (mod 4) then
        temp ← −temp
    end if
    a ← a (mod p)
end while
if p = 1 then
    return temp
else
    return 0
end if
```

Algoritma IV.8 Algoritma *Tonelli-Shanks*

Input: $a \in \mathbb{F}_p, p$ bilangan prima
Ensure: $x \in \mathbb{F}_p$ dimana $x^2 \equiv a \pmod p$

$q \leftarrow p-1$
$s \leftarrow 0$
while $q \equiv 0 \pmod 2$ **do**
    $q \leftarrow q/2$
    $s \leftarrow s+1$
end while
$z \leftarrow 2$
while true **do**
    if $z^{\frac{p-1}{2}} = 1$ **then**
        break
    end if
    $z \leftarrow z+1$
end while
$M \leftarrow s$
$c \leftarrow z^q$
$t \leftarrow a^q$
$R \leftarrow a^{\frac{q+1}{1}}$
while true **do**
    if $t = 0$ **then**
        return 0
    else if $t = 1$ **then**
        return R
    else
        $i \leftarrow 1$
        while $i < M$ **do**
            if $t^{2^i} = 1$ **then**
                break
            end if
            $i \leftarrow i+1$
            $b \leftarrow c^{2^{M-i-1}}$
            $M \leftarrow 1$
            $c \leftarrow b^2$
            $t \leftarrow t \cdot b^2 \pmod p$
            $R \leftarrow R \cdot b \pmod p$
        end while
    end if
end while

IV.1.8 Perkalian di $\mathbb{F}_p[x]$

Diberikan $p(x), q(x) \in \mathbb{F}_p[x]$, dengan $p(x) = a_0 + a_1 x + \cdots + a_{n-1}x^{n-1}$, $q(x) = b_0 + b_1 x + \cdots + b_{m-1}x^{m-1}$, dan $m \leq n$. Algoritma di bawah ini digunakan untuk mencari $t(x) = p(x) \cdot q(x)$. Dalam program, nilai koefisien dari polinomial dibalik menjadi $p(x) = a_0 x^{n-1} + \cdots + a_{n-2}x + a_{n-1}$.

Algoritma IV.9 Algoritma Perkalian di $\mathbb{F}_p[x]$

Input: $p(x), q(x) \in \mathbb{F}_p[x]$ dengan derajat $p(x)$ dan $q(x)$ masing-masing n dan m
Ensure: $t(x) = p(x) \cdot q(x)$

```
temp ← [0, 0, ..., 0] sebanyak n + m − 1
for 0 ≤ i ≤ n − 1 do
    for 0 ≤ i ≤ m − 1 do
        temp_{i+j} ← temp_{i+j} + a_i · b_j (mod p)
    end for
end for
return temp
```

IV.1.9 Pembagian di $\mathbb{F}_p[x]$

Diberikan $p(x), q(x) \in \mathbb{F}_p[x]$, dengan $p(x) = a_0 + a_1 x + \cdots + a_{n-1}x^{n-1}$, $q(x) = b_0 + b_1 x + \cdots + b_{m-1}x^{m-1}$, dan $m \leq n$ dimana $a_i, b_i \in \mathbb{F}_p$. Algoritma di bawah ini digunakan untuk mencari $t(x) = p(x)/q(x)$. Sebelum itu, terdapat algoritma normalisasi, algoritma ini digunakan untuk memastikan bahwa koefisien dari suku terbesar tidak 0.

Algoritma IV.10 Algoritma Normalisasi

Input: $p(x) \in \mathbb{F}_p[x]$
Ensure: $a_0 \neq 0$

```
while p = 0 dan a_0 = 0 do
    p ← [a_1, a_2, ..., a_{n-1}]
end while
return p
```

Algoritma IV.11 Algoritma Pembagian di $\mathbb{F}_p[x]$

Input: $p(x), q(x) \in \mathbb{F}_p[x]$ dengan derajat $p(x)$ dan $q(x)$ masing-masing n dan m, $m \leq n$
Ensure: $t(x) = p(x)/q(x)$

$diff \leftarrow n - m$
$temp \leftarrow [0, 0, \ldots, 0]$ sebanyak $n + m - 1$
if $diff < 0$ atau $q(x) = 0$ **then**
    return tidak bisa membagi dengan 0
end if
for $0 \leq i \leq diff + 1$ **do**
    if $len(p(x)) < len(q(x))$ **then**
        break
    end if
    if $p_i \neq 0$ **then**
        $temp1 \leftarrow a_i/b_0 \pmod p$
        $temp_{len(p(x))+i-(diff+1)} \leftarrow temp1$
        $ph \leftarrow [0$ sebanyak i, $temp_{len(p)+i-(diff+1):len(temp)}]$
        $pg \leftarrow q \cdot ph \pmod p$
        for $0 \leq j \leq len(pg)$ **do**
            $p_j \leftarrow p_j - pg_j \pmod p$
        end for
        $temp_{i+j} \leftarrow temp_{i+j} + a_i \cdot b_j \pmod p$
    end if
end for
$remainder \leftarrow p$
return normlisasi($temp$), normalisasi($remainder$)

IV.1.10 Perkalian di $\mathbb{F}_p[x]/\langle f(x) \rangle$

Polinomial tak tereduksi yang digunakan disini adalah $f(x) = x^2 + 1$ dan gelanggang kelas residu didefinisikan sebagai

$$\mathbb{F}_p[x]/\langle f(x) \rangle := \{a + bi : a, b \in \mathbb{F}_p\} \text{ dengan } i \text{ adalah akar } f(x).$$

Perkalian yang dilakukan serupa dengan perkalian pada bilangan kompleks. Misal $a, b \in \mathbb{F}_p[x]/\langle f(x) \rangle$ dimana $a = (a_0 + a_1 i)$ dan $b = (b_0 + b_1 i)$ $a_0, a_1, b_0, b_1 \in \mathbb{F}_p[x]$, maka

$$\begin{aligned} a \cdot b &= (a_0 + a_1 i) \cdot (b_0 + b_1 i) \\ &= b_0(a_0 + a_1 i) + b_1 i(a_0 + a_1 i) \\ &= a_0 b_0 + a_1 b_0 i + a_0 b_1 i + a_1 b_1 i^2 \end{aligned}$$

$$= a_0b_0 - a_1b_1 + (a_1b_0 + a_0b_1)i$$

Algoritma IV.12 Algoritma Perkalian di $\mathbb{F}_p[x]/\langle f(x)\rangle$

Input: $a, b \in \mathbb{F}_p[x]/\langle f(x)\rangle$
Ensure: $a \cdot b \in \mathbb{F}_p[x]/\langle f(x)\rangle$
1: $real \leftarrow a_0b_0 - a_1b_1$
2: $imaj \leftarrow a_1b_0 + a_0b_1$
3: **return** $(real, imaj \cdot i)$

IV.1.11 Pembagian di $\mathbb{F}_p[x]/\langle f(x)\rangle$

Seperti perkalian, pembagian di $\mathbb{F}_p[x]/\langle f(x)\rangle$ juga dilakukan sama seperti di bilangan kompleks. Misal $a, b \in \mathbb{F}_p[x]/\langle f(x)\rangle$ dimana $a = (a_0 + a_1i)$ dan $b = (b_0 + b_1i)$ $a_0, a_1, b_0, b_1 \in \mathbb{F}_p[x]$,

$$\begin{aligned}
\frac{a}{b} &= \frac{a_0 + a_1i}{b_0 + b_1i} \\
&= \frac{a_0 + a_1i}{b_0 + b_1i} \cdot \frac{b_0 - b_1i}{b_0 - b_1i} \\
&= \frac{(a_0b_0 + a_1b_1) + (a_1b_0 - a_0b_1)i}{b_0^2 + b_1^2} \\
&= \frac{a_0b_0 + a_1b_1}{b_0^2 + b_1^2} + \frac{a_1b_0 - a_0b_1}{b_0^2 + b_1^2}i
\end{aligned}$$

IV.1.12 Akar Kuadrat di $\mathbb{F}_p[x]/\langle f(x)\rangle$

Misal $z \in \mathbb{F}_p[x]/\langle f(x)\rangle$ dengan $a = a_0 + a_1i, b \neq 0$, maka akar dari z adalah

Algoritma IV.13 Algoritma Akar Kuadrat Bilangan Kompleks atas $\mathbb{F}_p[x]/\langle f(x)\rangle$

Input: polinomial tak tereduksi $f(x) = x^2 - \beta, \beta = -1, z = a + bi \in \mathbb{F}_p[x]/\langle f(x)\rangle$
Ensure: jika ada, $b = b_0 + b_1 i \in \mathbb{F}_p[x]/\langle f(x)\rangle$ memenuhi $b^2 = a$

if $a_1 = 0$ **then**
  return $\mathrm{SQRT}_p(a_0)$
end if
$\alpha \leftarrow a_0^2 - (-1) \cdot a_1^2$
$\gamma \leftarrow \alpha^{\frac{p-1}{2}}$
if $\gamma = -1$ **then**
  return false
end if
$\alpha \leftarrow \mathrm{SQRT}_p(\alpha)$
$\delta \leftarrow \dfrac{a_0 + \alpha}{2}$
$\gamma \leftarrow \delta^{\frac{p-1}{2}}$
if $\gamma = -1$ **then**
  $\delta \leftarrow \dfrac{a_0 - \alpha}{2}$
end if
$x_0 \leftarrow \mathrm{SQRT}_p(\delta)$
$x_1 \leftarrow \dfrac{a_1}{2x_0}$
$x \leftarrow x_0 + x_1 i$
return x

IV.2 Operasi di Kurva Eliptik

IV.2.1 Penjumlahan Titik Berbeda

Algoritma IV.14 Penjumlahan Titik Kurva Eliptik

Input: titik $P = (x_1, y_1)$, titik $Q = (x_2, y_2)$, kurva eliptik $E : y^2 = x^3 + Ax + B$
Ensure: $P + Q$

if $P = \mathcal{O}$ **then**
  return Q
else if $Q = \mathcal{O}$ **then**
  return P
else if $Q = P$ **then**
  $\lambda \leftarrow \dfrac{3x_1^2 + A}{2y_1} \pmod{p}$
else if $Q = -P$ **then**
  return $\mathcal{O}$
else
  $\lambda \leftarrow \dfrac{y_2 - y_1}{x_2 - x_1} \pmod{p}$
end if
$x_3 \leftarrow \lambda^2 - x_1 - x_2 \pmod{p}$
$y_3 \leftarrow \lambda(x_1 - x_3) - y_1 \pmod{p}$
return (x_3, y_3)

IV.2.2 Negasi Titik

Algoritma IV.15 Negasi Titik Kurva Eliptik

Input: titik $P = (x_1, y_1)$, kurva eliptik $E : y^2 = x^3 + Ax + B$
Ensure: $-P$

$x = x_1 \pmod{p}$
$y = -y_1 \pmod{p}$
return (x, y)

IV.2.3 Perkalian Titik

Perkalian titik $P = (x_1, y_1)$ pada kurva eliptik $E : y^2 = x^3 + Ax + B$ dilakukan dengan menggunakan metode gandakan-dan-jumlah (*double-and-add*).

Algoritma IV.16 Negasi Titik Kurva Eliptik

Input: titik $P = (x_1, y_1)$, kurva eliptik $E : y^2 = x^3 + Ax + B$, bilangan bulat positif n

Ensure: nP

$Q \leftarrow P$
$R \leftarrow \mathscr{O}$
while $n > 0$ **do**
    if $n \equiv 1 \pmod 2$ **then**
        $R \leftarrow R + Q$
    end if
    $Q \leftarrow 2Q$
    $n \leftarrow \lfloor n/2 \rfloor$
end while
return R

IV.2.4 *Weil Pairing*

Diberikan $P = (x_P, y_P)$ dan $Q = (x_Q, y_Q)$ titik-titik di $E(\mathbb{F}_p) : y^2 = x^3 + Ax + B, A, B \in \mathbb{F}_p$. *Weil pairing* dari P dan Q $e_m(P, Q)$ adalah

Algoritma IV.17 Algoritma *Weil Pairing*

Input:

1. titik $P = (x_P, y_P)$, titik $Q = (x_Q, y_Q)$, $P, Q \in E(\mathbb{F}_p)$
2. $E(\mathbb{F}_p) : y^2 = x^3 + Ax + B, A, B \in \mathbb{F}_p$
3. $m = m_0 + m_1 \cdot 2 + m_2 \cdot 2^2 + \cdots + m_{n-1} \cdot 2^{n-1}$ orde P dan Q dengan $m_i \in \{0, 1\}$ dan $m_{n-1} \neq 0$

Ensure: $e_m(P, Q)$

if $P = \mathscr{O}$ atau $Q = \mathscr{O}$ **then**
  return titik-titik tidak boleh $\mathscr{O}$
else if $Q = P$ **then**
  $\lambda \leftarrow \dfrac{3x_1^2 + A}{2y_1} \pmod{p}$
else if $Q = -P$ **then**
  return *lambda* $\leftarrow \infty$
else
  $\lambda \leftarrow \dfrac{y_2 - y_1}{x_2 - x_1} \pmod{p}$
end if
if $\lambda \neq \infty$ **then**
  $g_{P,Q} \leftarrow \dfrac{y - y_P - \lambda(x - x_P)}{x + x_P + x_Q - \lambda^2} \pmod{p}$
else
  $g_{P,Q} \leftarrow x - x_P \pmod{p}$
end if
$T = P$ ▷ Algoritma *Miller*
$f = 1$
for $n - 2 \leq i \leq 0$ **do**
  $f = f^2 \cdot g_{T,T} \pmod{p}$
  $T = 2T$
  if $m_i = 1$ **then**
    $f = f \cdot g_{T,P} \pmod{p}$
    $T = T + P$
  end if
end for
return f

IV.3 *QAP*

IV.3.1 Interpolasi Lagrange

Algoritma IV.18 Algoritma Interpolasi *Lagrange*

Input: $x = \{x_0, x_1, \ldots, x_n\}$ dan $y = \{y_0, y_1, \ldots, y_n\}$ koordinat titik yang bersesuaian, p bilangan prima

Ensure: $\mathscr{L}(x)$ polinomial interpolasi *Lagrange*

for $0 \leq i \leq n$ **do**
  $\mathscr{L}_i \leftarrow 1$
  for $0 \leq j \leq n$ **do**
    if $i \neq j$ **then**
      $\mathscr{L}_i \leftarrow \mathscr{L}_i \cdot \frac{x - x_j}{x_i - x_j}$
    end if
  end for
end for
$\mathscr{L} \leftarrow 0$
for $0 \leq k \leq n$ **do**
  $\mathscr{L} \leftarrow \mathscr{L} + \mathscr{L}_k \cdot y_k$
end for
return $\mathscr{L}$

IV.3.2 *QAP*

Algoritma IV.19 Algoritma Kode Fungsi-Sirkuit Aljabar

Input: $f = \{a_n, a_{n-1}, \ldots, a_0\}$ *array* dari fungsi $f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_0$, u bilangan sehingga $f(u) = y$, p bilangan prima
Ensure: matriks l, r, o dan vektor v

$v \leftarrow [1, u]$
$degree \leftarrow n$
$l \leftarrow []$
$r \leftarrow []$
$o \leftarrow []$
$initial\_sum \leftarrow u$
$total \leftarrow [0]$
for $1 \leq i \leq degree$ **do**
    if $a_i = 0$ **then**
        append $[a_i]$ **to** l
        append $[u]$ **to** r
        if a_i **not in** v **then**
            append a_i **to** v
        end if
        append $[a_i] \cdot u \pmod{p}$ **to** o
    end if
    $sums \leftarrow initial\_sum \cdot a_i \pmod{p}$
    if $sums$ **not in** v **then**
        append $sums$ **to** v
    end if
    for $0 \leq j \leq d - i - 1$ **do**
        append $[sums]$ **to** l
        append $[u]$ **to** r
        $sums \leftarrow sums \cdot u \pmod{p}$
        append $[sums]$ **to** o
        if $sums$ **not in** v **then**
            append $sums$ **to** v
        end if
    end for
    append sums **to** total
    if $i \neq 0$ **then**
        append $[sum(total[:-1]), total[-1]]$ **to** l
        append $[1]$ **to** r
        append $[sum](total)$ **to** o
    end if

if $sum(total)$ **not in** v **then**
append $sum(total)$ **to** v
end if
end for
if $f_0 \neq 0$ **then**
append $[sum(total[:-1])] \pmod{p}$ **to** l
append $[f_0]$ **to** $l[-1]$
append $[1]$ **to** r
append $[sum(l[-1]) \cdot r[-1][0]] \pmod{p}$ **to** o
append $o[-1][0]$ **to** $v[2]$
end if
return l, r, o, v

Algoritma IV.20 Algoritma Sirkuit Aljabar-*R1CS*

Input: matriks l, r atau o, vektor v
Ensure: matriks L, R atau O dengan nilai 0 atau 1
$result \leftarrow []$
for $0 \leq i \leq l$ **do**
$z \leftarrow [z_0, z_1, \ldots, z_i]_{i \in 0,1,\ldots,|v|}$ jika $v[i]$ **in** $l[i]$
end for
return $result$

Algoritma IV.21 Algoritma *R1CS-QAP*

Input: matriks L, R atau O dengan nilai 0 atau 1
Ensure: matriks $L(x), R(x)$ atau $O(x)$ dengan nilai koefisien dari polinomial-polinomial interpolasi *Lagrange*
$L(x) \leftarrow []$
for $0 \leq i \leq |L[0]|$ **do**
$x \leftarrow []$
$y \leftarrow []$
for $1 \leq j \leq |L|$ **do**
append i **to** x
append $L[i][j]$ **to** y
end for
$L[i](x) \leftarrow interpolasi\_Lagrange(x, y)$
append $L[i](x)$ **to** $L(x)$
end for
return $L(x)$

Algoritma IV.22 Algoritma *QAP*

Input: $f = \{a_n, a_{n-1}, \ldots, a_0\}$ *array* dari fungsi $f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_0$, u bilangan sehingga $f(u) = y$, p bilangan prima

Ensure: matriks $L(x), R(x), O(x)$ dengan nilai koefisien dari polinomial-polinomial interpolasi *Lagrange* dan polinomial target $t(x)$

$l, r, o, v \leftarrow function\_to\_circuit\_algebra(f, u, p)$
$l \leftarrow circuit\_algebra\_to\_rics(l, v)$
$r \leftarrow circuit\_algebra\_to\_rics(r, v)$
$o \leftarrow circuit\_algebra\_to\_rics(o, v)$
$L(x) \leftarrow circuit\_algebra\_to\_rics(l)$
$R(x) \leftarrow circuit\_algebra\_to\_rics(r)$
$O(x) \leftarrow circuit\_algebra\_to\_rics(o)$
$t(x) = 1$
for $1 \leq i \leq |v|$ **do**
    $t(x) \cdot (x - i)$
end for
return $L(x), R(x), O(x), t(x)$

IV.4 zk-SNARKs

IV.4.1 *Setup*

Algoritma IV.23 Algoritma *Setup zk-SNARKs*

Input: Beberapa input yang diperlukan:

1. bilangan prima p
2. kurva eliptik atas lapangan hingga $\mathbb{F}_p$ $E(\mathbb{F}_p) : y^2 = x^3 + Ax + B$
3. titik *generator* $g \in E(\mathbb{F}_p)$ dengan orde m
4. fungsi $f(x)$
5. u dimana $f(u) = y$

Ensure: *proving key verification key*

$L(x), R(x), O(x), t(x) \leftarrow QAP(f(x), u, p)$
$s \leftarrow random[1, m-1]$
$\rho_l \leftarrow random[1, m-1]$
$\rho_r \leftarrow random[1, m-1]$
$\alpha_l \leftarrow random[1, m-1]$
$\alpha_r \leftarrow random[1, m-1]$
$\alpha_o \leftarrow random[1, m-1]$
$\beta \leftarrow random[1, m-1]$
$\gamma \leftarrow random[1, m-1]$
$\rho_o \leftarrow \rho_l \cdot \rho_r$ (mod m)
$g_l \leftarrow \rho_l \cdot g$
$g_r \leftarrow \rho_r \cdot g$
$g_o \leftarrow \rho_o \cdot g$
$proving\_key \leftarrow$

$$\Big(\{s^k \cdot g\}_{0 \le k \le (d-1)}, \{l_i(x) \cdot g_l, r_i(x) \cdot g_r, o_i(x) \cdot g_o\}_{i \in \{m+1,\dots,n\}},$$
$$\{\alpha_l \cdot l_i(s) \cdot g_l, \alpha_r \cdot r_i(s) \cdot g_r, \alpha_o \cdot o_i(s) \cdot g_o\}_{i \in \{m+1,\dots,n\}},$$
$$\{\beta \cdot l_i(s) \cdot g_l + \beta \cdot r_i(s) \cdot g_r + \beta \cdot o_i(s) \cdot g_o\}_{i \in \{m+1,\dots,n\}},$$
$$t(s) \cdot g_l, t(s) \cdot g_r, t(s) \cdot g_o, \alpha_l \cdot t(s) \cdot g_l, \alpha_r \cdot t(s) \cdot g_r, \alpha_o \cdot t(s) \cdot g_o,$$
$$\beta \cdot t(s) \cdot g_l, \beta \cdot t(s) \cdot g_r, \beta \cdot t(s) \cdot g_o\Big)$$

verification_key $\leftarrow$

$$\Big(g, t(s)\cdot g_o, \{l_i(s)\cdot g_l, r_i(s)\cdot g_r, o_i(s)\cdot g_o\}_{i\in\{1,\ldots,m\}},$$
$$\alpha_l\cdot g, \alpha_r\cdot g, \alpha_o\cdot g, \gamma\cdot g, \beta\cdot\gamma\cdot g\Big)$$

return *proving_key*, *verification_key*

IV.4.2 Proving

Algoritma IV.24 Algoritma *Proving zk-SNARKs*

Input: Input yang diperlukan:

1. fungsi $f(x)$
2. *witness* w untuk menghitung $f(w)$
3. $proving\_key$
4. $verification\_key$

Ensure: $proof$

$l, r, o, v \leftarrow function\_to\_circuit\_algebra(f(x), w, p)$

$L(x), R(x), O(x), t(x) \leftarrow QAP(f(x), w, p)$

$\mathbf{L}(x) \leftarrow L(x) \cdot v$

$\mathbf{R}(x) \leftarrow L(x) \cdot v$

$\mathbf{O}(x) \leftarrow L(x) \cdot v$

$\delta_l \leftarrow random[1, m-1]$

$\delta_r \leftarrow random[1, m-1]$

$\delta_o \leftarrow random[1, m-1]$

$h(x) \leftarrow \dfrac{\mathbf{L}(x)\mathbf{R}(x) - \mathbf{O}(x)}{t(x)} + \delta_r \mathbf{L}(x) + \delta_l \mathbf{R}(x) + \delta_l \delta_r t(x) - \delta_o$

$\mathbf{L}_p(w) \cdot g_l \leftarrow \delta_l \cdot (t(w) \cdot g_l) + \sum_{i=m+1}^{n} v_i \cdot (l_i(w) \cdot g_l)$

$\mathbf{R}_p(w) \cdot g_r \leftarrow \delta_r \cdot (t(w) \cdot g_r) + \sum_{i=m+1}^{n} v_i \cdot (r_i(w) \cdot g_r)$

$\mathbf{O}_p(w) \cdot g_o \leftarrow \delta_o \cdot (t(w) \cdot g_o) + \sum_{i=m+1}^{n} v_i \cdot (o_i(w) \cdot g_o)$

$\mathbf{L}'_p(w) \cdot g_l \leftarrow \delta_l \cdot (\alpha_l \cdot t(w) \cdot g_l) + \sum_{i=m+1}^{n} v_i \cdot (\alpha_l \cdot l_i(w) \cdot g_l)$

$\mathbf{R}'_p(w) \cdot g_r \leftarrow \delta_r \cdot (\alpha_r \cdot t(w) \cdot g_r) + \sum_{i=m+1}^{n} v_i \cdot (\alpha_r \cdot r_i(w) \cdot g_r)$

$\mathbf{O}'_p(w) \cdot g_o \leftarrow \delta_o \cdot (\alpha_o \cdot t(w) \cdot g_o) + \sum_{i=m+1}^{n} v_i \cdot (\alpha_o \cdot o_i(w) \cdot g_o)$

$\mathbf{Z}(s) \cdot g \leftarrow$

$$\delta_l \cdot (\beta \cdot t(s) \cdot g_l) + \delta_r \cdot (\beta \cdot t(s) \cdot g_r) + \delta_o \cdot (\beta \cdot t(s) \cdot g_o) + \sum_{i=m+1}^{n} v_i \cdot (\beta \cdot l_i(s) \cdot g_l + \beta \cdot r_i(s) \cdot g_r + \beta \cdot o_i(s) \cdot g_o)$$

$proof \leftarrow$

$$\big(\mathbf{L}_p(s) \cdot g_l, \mathbf{R}_p(s) \cdot g_r, \mathbf{O}_p(s) \cdot g_o, h(s) \cdot g, \mathbf{L}'_p(s) \cdot g_l, \mathbf{R}'_p(s) \cdot g_r, \mathbf{O}'_p(s) \cdot g_o, \mathbf{Z}(s) \cdot g\big)$$

return $proof$

IV.4.3 Verification

Algoritma IV.25 Algoritma *Verification zk-SNARKs*

Input: $proof$, $verification\_key$

Ensure: true *prover* mengetahui rahasia (*witness* w benar), false selainnya

$\mathbf{L}_v(s) \cdot g_l \leftarrow \sum_{i=1}^{m} v_i \cdot (l_i(s) \cdot g_l)$

$\mathbf{R}_v(s) \cdot g_r \leftarrow \sum_{i=1}^{m} v_i \cdot (r_i(s) \cdot g_r)$

$\mathbf{O}_v(s) \cdot g_o \leftarrow \sum_{i=1}^{m} v_i \cdot (o_i(s) \cdot g_o)$

▷ Uji pembatasan variabel polinomial

if $\hat{e}(\mathbf{L}_p \cdot g_l, \alpha_l \cdot g) = \hat{e}\left(\mathbf{L}'_p \cdot g_l, g\right)$ **then**

  return false

end if

if $\hat{e}(\mathbf{R}_p \cdot g_r, \alpha_r \cdot g) = \hat{e}\left(\mathbf{R}'_p \cdot g_r, g\right)$ **then**

  return false

end if

if $\hat{e}(\mathbf{O}_p \cdot g_o, \alpha_o \cdot g) = \hat{e}\left(\mathbf{O}'_p \cdot g_o, g\right)$ **then**

  return false

end if

▷ Uji konsistensi nilai variabel

if $\hat{e}(\mathbf{L}_p \cdot g_l + \mathbf{R}_p \cdot g_r + \mathbf{O}_p \cdot g_o, \beta \cdot \gamma \cdot g) = \hat{e}(\mathbf{Z} \cdot g, \gamma \cdot g)$ **then**

  return false

end if

▷ Uji kevalidan operasi

if $\hat{e}(\mathbf{L}_p \cdot g_l + \mathbf{L}_v(s) \cdot g_l, \mathbf{R}_p \cdot g_r + \mathbf{R}_v(s) \cdot g_r) = \hat{e}(t(s) \cdot g_o, h \cdot g) \cdot \hat{e}(\mathbf{O}_p \cdot g_o + \mathbf{O}_v(s) \cdot g_o, g)$ **then**

  return false

end if

return true

IV.5 Hasil Running Program

IV.5.1 Enkripsi Homomorfik dan Pemetaan Linear Sederhana

```
proving_key, verification_key = setup()
```

```
proving_key
```

```
{'beta_polynomials': [16, 8, 18],
 'evaluation': [4, 8, 12, 16, 18, 6, 3, 2],
 'prover_left_polynomials': [4, 4, 2],
 'prover_output_polynomials': [16, 3, 3],
 'prover_right_polynomials': [1, 1, 1],
 'prover_shifted_left_polynomials': [12, 12, 9],
 'prover_shifted_output_polynomials': [3, 9, 9],
 'prover_shifted_right_polynomials': [1, 1, 1],
 'zero_knowledge': {'beta_left_target_polynomial': 9,
  'beta_output_target_polynomial': 6,
  'beta_right_target_polynomial': 13,
  'left_target_polynomial': 4,
  'output_target_polynomial': 3,
  'right_target_polynomial': 9,
  'shifted_left_target_polynomial': 12,
  'shifted_output_target_polynomial': 9,
  'shifted_right_target_polynomial': 6}}
```

```
verification_key
```

```
{'alpha_left': 9,
 'alpha_output': 4,
 'alpha_right': 16,
 'beta_gamma': 8,
 'gamma': 12,
 'generator': 2,
 'target_polynomials': 3,
 'verifier_left_polynomials': [9, 8, 1],
 'verifier_output_polynomials': [1, 1, 16],
 'verifier_right_polynomials': [4, 4, 1]}
```

Gambar 4: Setup Simple

```python
witness = 3
proof = prover_proof(proving_key, witness)
proof
```

```
{'prover_consistency_check_polynomial': 3,
 'prover_left_polynomial': 3,
 'prover_output_polynomial': 13,
 'prover_result_polynomial': 9,
 'prover_right_polynomial': 3,
 'prover_shifted_left_polynomial': 13,
 'prover_shifted_output_polynomial': 8,
 'prover_shifted_right_polynomial': 12}
```

```python
verified = verification(proof, verification_key)
```

```
Variable polynomials restriction check: True
Variable values consistency check:      True
Valid operations check:                 True
```

Gambar 5: Proving & Verification Simple

```python
for i in range(14):
  proof = prover_proof(proving_key, i)
  # print('proof', proof)
  verified = verification(proof, verification_key)
  print(i, verified)
  print('---------------------------------')
```

```
Variable polynomials restriction check: True
Variable values consistency check:      True
Valid operations check:                 False
0 False
---------------------------------
Variable polynomials restriction check: True
Variable values consistency check:      True
Valid operations check:                 False
1 False
---------------------------------
Variable polynomials restriction check: True
Variable values consistency check:      True
Valid operations check:                 False
2 False
---------------------------------
Variable polynomials restriction check: True
Variable values consistency check:      True
Valid operations check:                 True
3 True
---------------------------------
Variable polynomials restriction check: True
Variable values consistency check:      True
Valid operations check:                 False
4 False
---------------------------------
Variable polynomials restriction check: True
Variable values consistency check:      True
Valid operations check:                 False
5 False
---------------------------------
Variable polynomials restriction check: True
Variable values consistency check:      True
Valid operations check:                 False
6 False
---------------------------------
```

Gambar 6: Wrong Assignment Simple 1

```
Variable polynomials restriction check: True
Variable values consistency check:      True
Valid operations check:                 True
7 True
----------------------------------
Variable polynomials restriction check: True
Variable values consistency check:      True
Valid operations check:                 False
8 False
----------------------------------
Variable polynomials restriction check: True
Variable values consistency check:      True
Valid operations check:                 False
9 False
----------------------------------
Variable polynomials restriction check: True
Variable values consistency check:      True
Valid operations check:                 False
10 False
----------------------------------
11 False
----------------------------------
12 False
----------------------------------
13 False
----------------------------------
```

Gambar 7: Wrong Assignment Simple 2

IV.5.2 Kurva Eliptik dan *Weil pairing*

```
1 proving_key, verification_key = setup()
```

```
1 proving_key
```

```
{'beta_polynomials': [<__main__.Point_on_EC at 0x7f28e4fe19d0>,
  <__main__.Point_on_EC at 0x7f28e4fccbd0>,
  <__main__.Point_on_EC at 0x7f28e4ff9450>],
 'evaluation': [<__main__.Point_on_EC at 0x7f28e4fe1450>,
  <__main__.Point_on_EC at 0x7f28e4fe1a50>,
  <__main__.Point_on_EC at 0x7f28e4fe1410>,
  <__main__.Point_on_EC at 0x7f28e5114790>,
  <__main__.Point_on_EC at 0x7f28e4fe1f10>,
  <__main__.Point_on_EC at 0x7f28e4fe1fd0>,
  <__main__.Point_on_EC at 0x7f28e4fe1f50>,
  <__main__.Point_on_EC at 0x7f28ea5a1d90>],
 'prover_left_polynomials': [<__main__.Point_on_EC at 0x7f28e4fccd50>,
  <__main__.Point_on_EC at 0x7f28e4ff9350>,
  <__main__.Point_on_EC at 0x7f28e4ff9510>],
 'prover_output_polynomials': [<__main__.Point_on_EC at 0x7f28e4fe1f90>,
  <__main__.Point_on_EC at 0x7f28e4ff92d0>,
  <__main__.Point_on_EC at 0x7f28e4ff9250>],
 'prover_right_polynomials': [<__main__.O at 0x7f28e4fe1a90>,
  <__main__.O at 0x7f28e8c00fd0>,
  <__main__.O at 0x7f28e4ff90d0>],
 'prover_shifted_left_polynomials': [<__main__.Point_on_EC at 0x7f28e4ff9090>,
  <__main__.Point_on_EC at 0x7f28ea41de50>,
  <__main__.Point_on_EC at 0x7f28e4ff93d0>],
 'prover_shifted_output_polynomials': [<__main__.Point_on_EC at 0x7f28e4fe1810>,
  <__main__.Point_on_EC at 0x7f28e4fe1910>,
  <__main__.Point_on_EC at 0x7f28e4ff9410>],
 'prover_shifted_right_polynomials': [<__main__.O at 0x7f28e4ff91d0>,
  <__main__.O at 0x7f28e4ff9050>,
  <__main__.O at 0x7f28e4ff9210>],
 'zero_knowledge': {'beta_left_target_polynomial': <__main__.Point_on_EC at 0x7f28e4ff9590>,
  'beta_output_target_polynomial': <__main__.Point_on_EC at 0x7f28e4ff9750>,
  'beta_right_target_polynomial': <__main__.Point_on_EC at 0x7f28e4ff9790>,
  'left_target_polynomial': <__main__.Point_on_EC at 0x7f28e4ff9190>,
  'output_target_polynomial': <__main__.Point_on_EC at 0x7f28e4ff9390>,
  'right_target_polynomial': <__main__.Point_on_EC at 0x7f28e4ff9610>,
  'shifted_left_target_polynomial': <__main__.Point_on_EC at 0x7f28e4ff95d0>,
  'shifted_output_target_polynomial': <__main__.Point_on_EC at 0x7f28e4ff9650>,
  'shifted_right_target_polynomial': <__main__.Point_on_EC at 0x7f28e4ff9690>}}
```

Gambar 8: Setup ECC 1

```
1 verification_key
```

```
{'alpha_left': <__main__.Point_on_EC at 0x7f28e4ff9310>,
 'alpha_output': <__main__.Point_on_EC at 0x7f28e4ff9490>,
 'alpha_right': <__main__.Point_on_EC at 0x7f28e4ff9810>,
 'beta_gamma': <__main__.Point_on_EC at 0x7f28e4ff9890>,
 'gamma': <__main__.Point_on_EC at 0x7f28e4ff99d0>,
 'generator': <__main__.Point_on_EC at 0x7f28ea5a1d90>,
 'target_polynomials': <__main__.Point_on_EC at 0x7f28e4ff9850>,
 'verifier_left_polynomials': [<__main__.Point_on_EC at 0x7f28e4ff97d0>,
  <__main__.Point_on_EC at 0x7f28e4ff9710>,
  <__main__.O at 0x7f28e4ff98d0>],
 'verifier_output_polynomials': [<__main__.O at 0x7f28e4fe1e50>,
  <__main__.O at 0x7f28e4ff9550>,
  <__main__.Point_on_EC at 0x7f28e4ff9950>],
 'verifier_right_polynomials': [<__main__.Point_on_EC at 0x7f28e4ff96d0>,
  <__main__.Point_on_EC at 0x7f28e4ff94d0>,
  <__main__.O at 0x7f28e4ff9290>]}
```

Gambar 9: Setup ECC 2

```python
witness = 3
proof = prover_proof(proving_key, witness)
proof
```

```
{'prover_consistency_check_polynomial': <__main__.Point_on_EC at 0x7f1537d4fd90>,
 'prover_left_polynomial': <__main__.Point_on_EC at 0x7f1537e57110>,
 'prover_output_polynomial': <__main__.Point_on_EC at 0x7f153ebae6d0>,
 'prover_result_polynomial': <__main__.Point_on_EC at 0x7f1537e13410>,
 'prover_right_polynomial': <__main__.Point_on_EC at 0x7f1537d33950>,
 'prover_shifted_left_polynomial': <__main__.Point_on_EC at 0x7f1537ed9950>,
 'prover_shifted_output_polynomial': <__main__.Point_on_EC at 0x7f1537e02350>,
 'prover_shifted_right_polynomial': <__main__.Point_on_EC at 0x7f1537e57c50>}
```

```python
verified = verification(proof, verification_key)
```

```python
verified
```

```
True
```

Gambar 10: Proving & Verification ECC

```python
for i in range(100):
  proof = prover_proof(proving_key, i)
  verified = verification(proof, verification_key)
  print(i, verified)
```

```
0 False
1 False
2 False
3 True
4 False
5 False
6 False
7 False
8 False
9 False
10 False
11 False
12 False
13 False
14 False
15 False
16 False
17 False
18 False
19 False
20 False
21 False
22 False
23 False
24 False
25 False
26 False
27 False
28 False
29 False
30 False
31 False
32 False
```

Gambar 11: Wrong Assignment ECC 1

```
33 False
34 False
35 False
36 False
37 False
38 False
39 False
40 False
41 False
42 False
43 False
44 False
45 False
46 False
47 False
48 False
49 False
50 False
51 False
52 False
53 False
54 False
55 False
56 False
57 False
58 False
59 False
60 False
61 False
62 False
63 False
64 False
65 False
66 False
67 False
68 False
69 False
70 False
71 False
72 False
73 False
74 False
```

Gambar 12: Wrong Assignment ECC 2

```
75 False
76 False
77 False
78 False
79 False
80 False
81 False
82 False
83 False
84 False
85 False
86 False
87 False
88 False
89 False
90 False
91 False
92 False
93 False
94 False
95 False
96 False
97 False
98 False
99 False
```

Gambar 13: Wrong Assignment ECC 3

Bab V Simpulan dan Saran

Berdasarkan hasil yang diperoleh pada tugas akhir ini, diperoleh simpulan sebagai berikut.

V.1 Simpulan

1. Skema *zk-SNARK* dengan enkripsi homomorfik dan pemetaan bilinear sederhana dapat berjalan. Tetapi, masih terdapat kelemahan yaitu rahasia atau *witness* dari *prover* tidak eksklusif. Artinya, ada kemungkinan bahwa rahasia atau *witness* yang seharusnya tidak sesuai, lolos proses verifikasi. Contohnya ada pada Gambar 7, dimana nilai 7 yang bukan rahasia sebenarnya lolos verifikasi.

2. Skema *zk-SNARK* dengan kurva eliptik dan *Weil pairing* dapat mengatasi permasalahan pada enkripsi homomorfik dan pemetaan bilinear sederhana. Pada Gambar 11, 12, dan 13 dapat dilihat bahwa semua nilai tidak lolos proses verifikasi kecuali 3 yang merupakan rahasia atau *witness*.

3. Dalam prosesnya, pihak ketiga yang berperan dalam tahap *setup* dalam *zk-SNARK* diasumsikan sebagai pihak yang terpercaya.

V.2 Saran

Tentunya masih banyak hal yang dapat diperbaiki atau dikembangkan lebih lanjut lagi. Hal pertama yang dapat dilakukan adalah dengan mengembangkan proses *QAP* menjadi lebih baik, dimana *input* yang diterima tidak hanya harus satu input, tetapi bisa lebih dari satu input sehingga poin nomor 2 pada simpulan dapat teratasi lebih baik.

Hal kedua adalah dengan memanfaatkan sifat *transparent* dari *zk-STARK (Zero Knowledge Scalable Transparent Argument of Knowledge)*, dimana tidak perlu adanya pihak ketiga pada *setup*. Sehingga proses *zero knowledge* menjadi lebih terpercaya. Dengan tidak adanya pihak ketiga pada *setup* dan bukti (*proof*) yang dihasilkan berukuran kecil, proses *zero knowledge* dapat menjadi lebih cepat.

Hal ketiga adalah membuat algoritma yang lebih cepat dan efektif, baik untuk melakukan operasi-operasi pada lapangan hingga, gelaggang kelas residu, maupun pada kurva eliptik.

Pustaka

[1] Adj, G. & Rodriguez-Henriquez, F. (2013). *Square root computation over even extension fields*. *IEEE Transactions on Computers*, 63(11), 2829 - 2841. Diakses pada 30 Mei 2022, dari https://eprint.iacr.org/2012/685.pdf.

[2] Banerjee, A., Clear, M., Tewari, H. (2020). *Demystifying the Role of zk-SNARKs in Zcash*. *Cornell University*, 8. Diakses pada 30 Juni 2022, dari https://arxiv.org/abs/2008.00881.

[3] Buterin, V. (2016). *Quadratic Arithmetic Programs: from Zero to Hero*. Diakses pada 11 Oktober 2021, dari https://medium.com/@VitalikButerin/quadratic-arithmetic-programs-from-zero-to-hero-f6d558cea649.

[4] cygnusv. (2017). *Tate Pairing Example*. Diakses pada 5 Februari 2022, dari https://gist.github.com/cygnusv/f41ee4026dc56cec693a306b1926b472.

[5] Gabizon, A. (2017). *Explaining SNARKs Part V: From Computations to Polynomials*. Diakses pada 20 November 2021, dari https://electriccoin.co/blog/snark-explain5/.

[6] Hoffstein, J., Pipher, J., Silverman, J.H. (2008). *An Introduction to Mathematical Cryptography*. USA: Springer.

[7] Lidl, R. & Harald N. (1994). *Introduction to Finite Fields and Their Applications*. Cambridge: Cambridge University Press.

[8] Mayer, H. (2019). *zk-SNARK explained: Basis Principles*. *Coin Fabrik*, 8. Diakses pada 11 Oktober 2021, dari https://blog.coinfabrik.com/wp-content/uploads/2017/03/zkSNARK-explained_basic_principles.pdf.

[9] Nakamoto, S. (2009). *Bitcoin: A Peer-to-Peer Electronic Cash System*. Diakses pada 10 November 2021, dari https://bitcoin.org/bitcoin.pdf.

[10] Petkus, M. (2019). *Why and How zk-SNARK Works: Definitive Explanation*. *Cornell University*, 65. Diakses pada 11 Oktober 2021, dari https://arxiv.org/abs/1906.07221v1.

[11] Richter, M. (2019). *Pinocchio Short Signatures for Computation A Pen & Paper Example*. Diakses pada 11 Oktober 2021, dari https://leastauthority.com/static/slides/ZKsnarks_workshop_slides.pdf.

[12] Stinson, D. (2006). *Cryptography: Theory and Practice Third Edition*. Boca Raton: Chapman & Hall/CRC.

Lampiran

Di bawah ini terlampir tautan-tautan untuk program tugas akhir ini yang sudah dibuat. Semua kode dibuat menggunakan bahasa pemrograman *Python* dan menggunakan *Google Colaboratory*.

- *Elliptic Curve*: berisi kode mengenai operasi dalam kurva eliptik.
  Elliptic Curve
- *zk-SNARKS*: berisi kode mengenai *zk-SNARKs* dengan enkripsi homomorfik dan pemetaan bilinear sederhana.
  zk-SNARKS
- *zk-SNARKs ECC [main program]*: berisi kode mengenai *zk-SNARKs* dengan kurva eliptik dan *Weil pairing*.
  zk-SNARKs ECC [main program].