Bab 3

Matematika I - Aritmatika Modular

Aritmatika modular sangat berperan dalam kriptografi karena banyak digunakan dalam algoritma enkripsi, baik untuk enkripsi simetris maupun untuk *public key cryptography*. Dalam aritmatika modular, konsep gcd digunakan antara lain untuk operasi *inverse*. Gcd dapat dikalkulasi secara efisien menggunakan algoritma Euclid, algoritma sangat penting yang telah berusia lebih dari 2000 tahun. Bab ini menjelaskan gcd, algoritma Euclid dan aritmatika modular. Akan tetapi, sebelum itu, kita definisikan terlebih dahulu beberapa struktur aljabar yang banyak digunakan dalam aritmatika, yaitu *group*, *monoid*, *ring* dan *field*.

3.1 Group, Monoid, Ring dan Field

Definisi 1 (Group) *Suatu group G dengan operasi biner $*$ adalah suatu himpunan dengan struktur aljabar sebagai berikut:*

- *Jika $a, b \in G$ maka $(a * b) \in G$ (closure).*
- *$a * (b * c) = (a * b) * c$ (associativity).*
- *Terdapat elemen $e \in G$ dimana $a * e = a = e * a$ untuk setiap $a \in G$ (identity).*
- *Untuk setiap $a \in G$ terdapat $b \in G$ dengan $a * b = e = b * a$ (inverse).*

Untuk commutative group (dinamakan juga Abelian group), terdapat satu syarat lagi:

- $a * b = b * a$ *(commutativity).*

Sebagai contoh, pertambahan dengan bilangan bulat adalah *Abelian group* dengan himpunan semua bilangan bulat, operasi biner $+$, *identity* 0, dan $-a$ sebagai *inverse* untuk a.

Definisi 2 (Monoid) *Suatu monoid G dengan operasi biner $*$ adalah suatu himpunan dengan struktur aljabar sebagai berikut:*

- *Jika $a, b \in G$ maka $(a * b) \in G$ (closure).*
- $a * (b * c) = (a * b) * c$ *(associativity).*
- *Terdapat elemen $e \in G$ dimana $a * e = a = e * a$ untuk setiap $a \in G$ (identity).*

Untuk commutative monoid (dinamakan juga Abelian monoid), terdapat satu syarat lagi:

- $a * b = b * a$ *(commutativity).*

Sebagai contoh, perkalian dengan bilangan bulat adalah *Abelian monoid* dengan himpunan semua bilangan bulat, operasi biner perkalian $\cdot$, dan *identity* 1.

Definisi 3 (Ring) *Suatu ring R dengan operasi $+$ dan $\cdot$ dan elemen 0 dan 1 adalah suatu himpunan dengan struktur aljabar sebagai berikut:*

- *R dengan operasi $+$ mempunyai struktur Abelian group dengan identity 0.*
- *R dengan operasi $\cdot$ mempunyai struktur Abelian monoid dengan identity 1.*
- $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$ *(distributivity).*

Jadi yang dimaksud dengan *ring* dalam buku ini adalah *commutative ring with identity*. Operasi $a \cdot b$ kerap disingkat menjadi ab. Sebagai contoh dari *ring* adalah aritmatika bilangan bulat dengan pertambahan dan perkalian. Simbol $\mathbf{Z}$ digunakan sebagai notasi untuk himpunan semua bilangan bulat. Simbol $\mathbf{N}$ digunakan sebagai notasi untuk himpunan semua bilangan bulat non-negatif (disebut juga bilangan natural). Walaupun $\mathbf{N}$ bukan *ring* (karena *inverse* pertambahan tidak berlaku), $\mathbf{N}$ dengan prinsip *well-ordering* sangat berguna dalam pembuktian yang membutuhkan prinsip induksi.

Definisi 4 (Field) *Suatu field F adalah suatu ring dimana setiap $0 \neq a \in F$ mempunyai inverse perkalian a^{-1} dengan $a \cdot a^{-1} = 1$.*

Jadi suatu *field* adalah suatu *ring* R dimana $R \setminus \{0\}$ dengan operasi $\cdot$ membentuk *Abelian group*. Contoh dari *field* adalah aritmatika bilangan nyata dimana semua bilangan kecuali 0 memiliki *inverse* perkalian. Aritmatika bilangan rasional juga merupakan contoh dari *field*. Simbol yang digunakan sebagai notasi untuk himpunan semua bilangan nyata adalah $\mathbf{R}$, sedangkan simbol yang digunakan sebagai notasi untuk himpunan semua bilangan rasional adalah $\mathbf{Q}$. Tentunya $\mathbf{R}$ dan $\mathbf{Q}$ juga merupakan *ring*, akan tetapi $\mathbf{Z}$ bukan *field* karena *inverse* perkalian tidak berlaku untuk bilangan bulat.

3.2 Prinsip Induksi

Banyak teorema dalam aljabar yang pembuktiannya membutuhkan prinsip induksi. Prinsip induksi berlaku pada proposisi mengenai bilangan natural (bilangan bulat non-negatif), tetapi dapat juga digeneralisasi sehingga berlaku untuk proposisi mengenai bilangan ordinal. Dalam buku ini, kita cukup menggunakan prinsip induksi pada bilangan natural.

Teorema 1 (Prinsip Induksi) *Jika P merepresentasikan proposisi mengenai bilangan natural, kita gunakan notasi "$P(n)$" untuk merepresentasikan "proposisi P berlaku untuk bilangan natural n." Jika kita dapat buktikan bahwa:*

$$\begin{aligned} &(i) \quad P(0), \text{ dan} \\ &(ii) \quad P(n) \Longrightarrow P(n+1) \text{ untuk setiap } n \in \mathbf{N}. \end{aligned}$$

Maka $P(n)$ berlaku untuk setiap bilangan $n \in \mathbf{N}$.

Secara informal sangat masuk akal mengapa prinsip induksi berlaku. Kita umpamakan bahwa kita dapat buktikan (i) dan (ii). Dari (i) kita dapat buktikan $P(0)$. Dari $P(0)$ dan (ii) kita dapat buktikan $P(1)$, dan langkah ini dapat diulang m kali untuk membuktikan $P(m)$. Karena kita dapat membuktikan $P(m)$ untuk sembarang $m \in \mathbf{N}$, maka seharusnya $P(n)$ berlaku untuk setiap $n \in \mathbf{N}$. Akan tetapi secara matematis pembuktiannya agak sedikit lebih rumit karena melibatkan himpunan *infinite* yaitu $\mathbf{N}$. Kita tidak akan membahas pembuktian matematis yang formal karena membutuhkan pembahasan fondasi matematika.

$P(0)$ diatas kerap disebut sebagai *base case*. Selain *base case* $P(0)$, prinsip induksi dapat juga digunakan dengan *base case* $P(k)$ dimana k adalah bilangan natural bukan 0. Hasilnya adalah pembuktian $P(n)$ untuk setiap $n \in \mathbf{N}$ dimana $n \geq k$. Langkah induksi juga kerap dirumuskan sebagai berikut:

$$P(n-1) \Longrightarrow P(n) \text{ untuk setiap } n > 0 \text{ atau } n > k,$$

dimana $n \in \mathbf{N}$.

Selain teorema 1, ada dua bentuk lain dari prinsip induksi yang sering digunakan. Bentuk alternatif pertama adalah sebagai berikut (notasi $\forall$ berarti "untuk setiap"):

Teorema 2 *Jika kita dapat buktikan bahwa:*

$$\begin{array}{ll} (i) & P(0), \text{ dan} \\ (ii) & \forall n \in \mathbf{N} : (\forall m \leq n : P(m)) \Longrightarrow P(n+1). \end{array}$$

Maka $\forall n \in \mathbf{N} : P(n)$ *berlaku.*

Prinsip induksi sebagaimana dalam teorema 2 disebut juga *strong induction principle*. Perbedaan antara kedua prinsip hanya terletak pada *step case*: dengan *strong induction* kita dapat menggunakan lebih banyak asumsi dalam membuktikan $P(n+1)$. Meskipun prinsip *strong induction* sepertinya menghasilkan mekanisme yang lebih ampuh, kita dapat buktikan bahwa sebenarnya kedua prinsip ekuivalen. Untuk menunjukkan bahwa $P(n)$ dapat dibuktikan menggunakan *strong induction* jika $P(n)$ dapat dibuktikan menggunakan induksi (teorema 1) sangat mudah: pada *step case*, kita cukup melakukan instansiasi $m = n$. Untuk membuktikan sebaliknya, kita perlu membuktikan bahwa *step case* induksi dapat ditransformasi menjadi menjadi *step case* untuk *strong induction*. Kita gunakan notasi:

$$Q(n) = \forall m \leq n : P(m).$$

Jadi kita perlu membuktikan $Q(n)$ dengan asumsi $P(n)$. Kita buktikan ini menggunakan induksi. Untuk *base case* cukup mudah karena

$$\begin{array}{rcl} Q(0) & = & \forall m \leq 0 : P(m) \\ & = & P(0). \end{array}$$

Untuk *step case* kita perlu buktikan:

$$Q(n) \Longrightarrow Q(n+1).$$

Karena $Q(n) = \forall m \leq n : P(m)$ dan $P(n) \Longrightarrow P(n+1)$, maka

$$Q(n) \Longrightarrow P(n+1).$$

Dari $Q(n)$ dan $P(n+1)$ kita dapatkan $Q(n+1)$ dan selesailah pembuktian kita.

Bentuk alternatif kedua dari prinsip induksi adalah prinsip *well-ordering* sebagai berikut:

Teorema 3 *Jika* M *adalah subset non-kosong dari* $\mathbf{N}$ *maka* M *mempunyai elemen terkecil (least element).*

Kita buktikan kontrapositif dari prinsip ini, yaitu jika M tidak mempunyai elemen terkecil maka M adalah himpunan kosong. Untuk membuktikan bahwa M adalah himpunan kosong, kita buktikan bahwa $n \notin M$ untuk setiap $n \in \mathbf{N}$. Kita buktikan ini menggunakan prinsip *strong induction*. Untuk *base case* sangat mudah karena jika $0 \in M$ maka 0 adalah elemen terkecil dalam M, jadi $0 \notin M$. Untuk *step case*, kita umpamakan $\forall m \leq n : m \notin M$ dan kita harus tunjukkan bahwa $(n+1) \notin M$. Jika $(n+1) \in M$ maka $n+1$ adalah elemen terkecil dalam M karena setiap bilangan natural yang lebih kecil dari $n+1$ tidak berada dalam M. Jadi $(n+1) \notin M$ dan selesailah pembuktian kita.

3.3 GCD

Kita mulai penjelasan gcd dengan teorema mengenai pembagian:

Teorema 4 (Pembagian) *Untuk setiap pasangan bilangan bulat a dan b dengan $b > 0$, terdapat pasangan unik bilangan bulat q dan r yang mematuhi persamaan:*

$$a = qb + r \text{ dengan } 0 \leq r < b. \tag{3.1}$$

Teorema ini dapat dibuktikan menggunakan prinsip *well ordering* (teorema 3). Untuk itu kita gunakan dua himpunan "standard" yaitu:

- $\mathbf{Z}$, himpunan dari semua bilangan bulat (*integers*).
- $\mathbf{N}$, himpunan dari semua bilangan bulat non-negatif (*natural numbers*).

Pertama, kita buat:

$$\begin{aligned} S &= \{a - nb | n \in \mathbf{Z}\} = \{a, a+b, a-b, a+2b, a-2b, \ldots\}, \\ S^+ &= S \cap \mathbf{N} = \{a \in S | a \geq 0\}. \end{aligned}$$

Jadi S merupakan himpunan semua bilangan bulat yang berbeda kelipatan b dari a. Karena sebagian dari elemen himpunan S adalah bilangan bulat non-negatif, S^+ merupakan subset non-kosong dari $\mathbf{N}$. Jadi kita dapat gunakan prinsip *well-ordering*, yang mengatakan bahwa S^+ mempunyai elemen terkecil, sebut saja r, yang berdasarkan definisi S, mempunyai bentuk $r = a - qb$ dengan q berupa bilangan bulat. Ini membuktikan bahwa untuk sepasang a dan b dengan $b > 0$, terdapat pasangan q dan r yang mematuhi persamaan

$$a = qb + r.$$

Karena $r \in S^+$, maka $0 \leq r$. Karena r adalah elemen terkecil S^+, maka $r < b$, sebab konsekuensi $r \geq b$ adalah S^+ mempunyai elemen $r - b$ yang lebih kecil dari r, sesuatu yang tidak mungkin apabila r adalah elemen terkecil. Jadi

$$0 \leq r < b.$$

Untuk menunjukkan bahwa pasangan q dan r unik, kita umpamakan bahwa

$$\begin{aligned} a &= qb + r \\ &= q'b + r' \end{aligned}$$

dengan

$$0 \leq r < b \text{ dan } 0 \leq r' < b,$$

jadi $r - r' = (q' - q)b$. Jika $q \neq q'$ maka $|q' - q| \geq 1$, yang berarti $|r - r'| \geq |b| = b$, sesuatu yang tidak mungkin karena $0 \leq r < b$ dan $0 \leq r' < b$ berarti perbedaan antara r dan r' lebih kecil dari b. Oleh karena itu $q' = q$ dan akibatnya $r' = r$.

Teorema mengenai pembagian diatas menjadi dasar dari konsep *residue* untuk bilangan bulat sebagai berikut. Membagi persamaan 3.1 dengan b, kita dapatkan:

$$\frac{a}{b} = q + \frac{r}{b} \text{ dengan } 0 \leq \frac{r}{b} < 1.$$

Kita bisa lihat bahwa q adalah bagian bulat dari a/b, jadi q merupakan bilangan bulat terbesar yang $\leq a/b$. Jadi q, yang juga disebut *quotient*, dapat dikalkulasi dengan mudah karena merupakan hasil pembagian dibulatkan kebawah. Setelah q didapat, r, yang juga disebut *remainder* atau *residue* dapat dikalkulasi menggunakan rumus $r = a - qb$.

Teorema diatas berlaku untuk $b > 0$. Bagaimana dengan $b < 0$? Untuk $b < 0$, teorema diatas berlaku untuk $-b$, jadi untuk setiap pasangan bilangan bulat a dan b dengan $b < 0$ terdapat pasangan unik bilangan bulat q' dan r yang mematuhi persamaan:

$$a = q'(-b) + r \text{ dengan } 0 \leq r < -b.$$

Dengan $q = -q'$ kita dapatkan $a = qb + r$. Jadi teorema bisa direvisi menjadi:

Teorema 5 (Pembagian) *Untuk setiap pasangan bilangan bulat a dan b dengan $b \neq 0$ terdapat pasangan unik q dan r yang mematuhi persamaan:*

$$a = qb + r \text{ dengan } 0 \leq r < |b|. \tag{3.2}$$

Untuk $b < 0$,

$$\frac{a}{b} = q + \frac{r}{b} \text{ dan } 0 \geq \frac{r}{b} > -1.$$

Jadi untuk $b < 0$, q merupakan bilangan bulat terkecil $\geq a/b$ (q dibulatkan keatas).

Untuk setiap pasangan bilangan bulat a dan b, jika terdapat bilangan bulat q sehingga $a = qb$, maka b membagi a, dan b disebut pembagi (*divisor* atau faktor) dari a dengan notasi $b|a$. Notasi $b \not| a$ digunakan jika b bukan pembagi a.

Definisi 5 (GCD) *Jika $d|a$ dan $d|b$ maka d adalah pembagi persekutuan (common divisor) dari a dan b. Untuk setiap pasangan bilangan bulat a dan b kecuali jika $a = b = 0$, pembagi persekutuan terbesar (greatest common divisor atau* gcd*) dari a dan b adalah bilangan bulat unik d dimana:*

1. *d merupakan pembagi persekutuan dari a dan b,*
2. *jika c merupakan pembagi persekutuan dari a dan b, maka $c \leq d$.*

Dalam beberapa cabang matematika yang lebih abstrak, syarat 2 diubah dengan $c|d$ menggantikan $c \leq d$ sehingga d dan $-d$ keduanya merupakan $\gcd(a, b)$. Menurut teori abstrak mengenai struktur *ring*, jika d merupakan $\gcd(a, b)$ dan u adalah sebuah *unit*[1] dalam struktur *ring*, maka ud juga merupakan $\gcd(a, b)$, jadi gcd belum tentu unik jadi bukan merupakan fungsi. Untuk bilangan bulat, kita gunakan versi diatas agar gcd unik dan positif. Kita akan gunakan gcd versi abstrak dalam pembahasan beberapa konsep dalam teori *ring*.

3.4 Algoritma Euclid

Satu cara untuk mendapatkan $\gcd(a, b)$ adalah dengan membuat daftar semua faktor dari a, membuat daftar semua faktor dari b, dan kemudian mencari faktor terbesar yang ada dalam kedua daftar. Akan tetapi, untuk bilangan yang sangat besar, membuat daftar faktor bukanlah sesuatu yang mudah. Ada cara yang jauh lebih efisien untuk mendapatkan $\gcd(a, b)$ yaitu dengan menggunakan algoritma Euclid (*Euclidean algorithm*), yang seperti halnya dengan *Chinese Remainder Theorem* merupakan algoritma penting yang berusia lebih dari 2000 tahun.

Teorema yang digunakan sebagai dasar dari algoritma Euclid adalah sebagai berikut:

Teorema 6 (Algoritma Euclid)

$$\textit{Jika } a = qb + r \textit{ maka } \gcd(a, b) = \gcd(b, r).$$

Untuk meyakinkan kita sendiri bahwa teorema diatas benar, kita tahu bahwa jika $a = qb+r$ maka setiap pembagi persekutuan b dan r juga membagi $qb+r = a$. Juga, karena $r = a - qb$, setiap pembagi persekutuan a dan b juga membagi r. Akibatnya setiap pembagi persekutuan a dan b juga merupakan pembagi persekutuan b dan r, dan setiap pembagi persekutuan b dan r juga merupakan pembagi persekutuan a dan b, jadi $\gcd(a, b) = \gcd(b, r)$.

Algoritma Euclid menggunakan rumus diatas secara berulang untuk mendapatkan gcd, yaitu dengan memperkecil kedua bilangan yang dijadikan patokan untuk gcd setiap kali mengulang, tanpa merubah nilai gcd itu sendiri.

[1] Untuk struktur *ring* bilangan bulat, ada dua *unit*: 1 dan -1.

Hasil dari komputasi gcd didapat saat kedua patokan untuk gcd tidak dapat diperkecil lagi.

Untuk melakukan komputasi $d = \gcd(a, b)$, pertama dilakukan *preprocessing* sebagai berikut:

1. Jika $a = 0$ maka $d = |b|$ dan jika $b = 0$ maka $d = |a|$ (gcd tidak dapat dikomputasi jika $a = b = 0$).

2. Karena $\gcd(a, b) = \gcd(-a, b) = \gcd(a, -b) = \gcd(-a, -b)$, kita dapat mentransformasi komputasi menjadi $d = \gcd(|a|, |b|)$, jadi kedua bilangan menjadi positif.

3. Karena $\gcd(a, b) = \gcd(b, a)$, kita dapat saling tukar a dan b jika $a < b$, dengan hasil $a \geq b$.

4. Jika $a = b$ maka $d = a$.

Setelah *preprocessing*, jika $a \neq 0$, $b \neq 0$, dan $a \neq b$, maka komputasi sudah dirubah menjadi $d = \gcd(a, b)$ dengan $a > b > 0$.

Langkah berikutnya adalah membagi a dengan b menggunakan algoritma pembagian, mendapatkan *quotient* q_1 dan *residue* r_1 ($\gcd(a, b) = \gcd(b, r_1)$ berdasarkan teorema 6):

$$a = q_1 b + r_1 \text{ dengan } 0 \leq r_1 < b.$$

Jika $r_1 = 0$ maka b membagi a, jadi $d = b$ dan kita selesai. Jika $r_1 \neq 0$ kita bagi b dengan r_1 mendapatkan *quotient* q_2 dan *residue* r_2 ($\gcd(a, b) = \gcd(r_1, r_2)$):

$$b = q_2 r_1 + r_2 \text{ dengan } 0 \leq r_2 < r_1.$$

Jika $r_2 = 0$ maka r_1 membagi b, jadi $d = r_1$ dan kita selesai. Jika $r_2 \neq 0$ kita teruskan ($\gcd(a, b) = \gcd(r_2, r_3)$):

$$r_1 = q_3 r_2 + r_3 \text{ dengan } 0 \leq r_3 < r_2,$$

dan seterusnya jika $r_3 \neq 0$, sampai kita dapatkan $r_n = 0$ (sampai dengan langkah ini kita mengetahui bahwa $\gcd(a, b) = \gcd(r_{n-2}, r_{n-1}) = \gcd(r_{n-1}, r_n)$):

$$r_{n-2} = q_n r_{n-1} + r_n \text{ dengan } r_n = 0.$$

Dengan $r_n = 0$, kita tahu bahwa r_{n-1} membagi r_{n-2}, jadi $d = r_{n-1}$ dan kita selesai. Tidak terlalu sulit untuk melihat bahwa algoritma Euclid akan berhenti pada suatu r_n dengan $n \geq 0$, karena tidak mungkin terdapat deretan

$$r_1 > r_2 > r_3 > \ldots$$

yang tidak berhenti (dimana setiap r_i merupakan bilangan bulat positif).

Sebagai contoh, mari kita kalkulasi $\gcd(1485, 1745) = \gcd(1745, 1485)$:

$$\begin{aligned} 1745 &= 1 \cdot 1485 + 260 \\ 1485 &= 5 \cdot 260 + 185 \\ 260 &= 1 \cdot 185 + 75 \\ 185 &= 2 \cdot 75 + 35 \\ 75 &= 2 \cdot 35 + 5 \\ 35 &= 7 \cdot 5 + 0. \end{aligned}$$

Jadi $\gcd(1485, 1745) = 5$.

Sebagai konsekuensi dari algoritma Euclid, kita dapatkan $gcd(a, b)$ sebagai kombinasi linear dari a dan b:

Teorema 7 *Untuk setiap pasangan bilangan bulat a dan b, kecuali $a = b = 0$, terdapat pasangan bilangan bulat u dan v yang mematuhi:*

$$\gcd(a, b) = au + bv.$$

Pasangan u dan v dapat kita cari menggunakan persamaan-persamaan yang digunakan dalam algoritma Euclid. Menggunakan contoh diatas:

$$\begin{aligned} 5 &= 75 - 2 \cdot 35 \\ &= 75 - 2 \cdot (185 - 2 \cdot 75) \\ &= -2 \cdot 185 + 5 \cdot 75 \\ &= -2 \cdot 185 + 5 \cdot (260 - 185) \\ &= 5 \cdot 260 - 7 \cdot 185 \\ &= 5 \cdot 260 - 7 \cdot (1485 - 5 \cdot 260) \\ &= -7 \cdot 1485 + 40 \cdot 260 \\ &= -7 \cdot 1485 + 40 \cdot (1745 - 1485) \\ &= -47 \cdot 1485 + 40 \cdot 1745. \end{aligned}$$

Kita dapatkan pasangan $u = -47$ dan $v = 40$ sebagai solusi. Pasangan u dan v tidak unik karena kita bisa tambahkan atau kurangkan kombinasi linear a dan b yang mempunyai nilai 0 tanpa mempengaruhi nilai d. Jadi jika $u'a + v'b = 0$ dengan $u' \neq 0$ atau $v' \neq 0$ (contohnya $u' = b$ dan $v' = -a$), kita dapatkan $d = (u + u')a + (v + v')b$ dengan $u + u' \neq u$ atau $v + v' \neq v$ (pasangan $u + u'$ dan $v + v'$ tidak sama dengan pasangan u dan v). Sebagai contoh:

$$5 = 1698 \cdot 1485 + (-1445) \cdot 1745$$

Algoritma Euclid dapat direvisi agar sekaligus mendapatkan pasangan u dan v disamping mendapatkan gcd d. Algoritma yang sudah direvisi disebut juga *extended Euclidean algorithm*. Untuk $a, b > 0$, *extended Euclidean algorithm* mencari $d = \gcd(a, b), u$ dan v dengan $d = au + bv$. Agar yakin bahwa

algoritma benar, kita gunakan konsep *loop invariant*, yaitu suatu proposisi yang berlaku pada setiap putaran. *Loop invariant* yang digunakan adalah

$$\gcd(a,b) = \gcd(A,B), A = au + bv \text{ dan } B = as + bt.$$

Langkah-langkah untuk algoritma adalah sebagai berikut:

1. $A \leftarrow a, B \leftarrow b, u \leftarrow 1, v \leftarrow 0, s \leftarrow 0, t \leftarrow 1$.
2. $q \leftarrow A$ div B.
3. $r \leftarrow A - qB$.
4. $A \leftarrow B, B \leftarrow r, U \leftarrow u, V \leftarrow v$.
5. $u \leftarrow s, v \leftarrow t, s \leftarrow U - qs, t \leftarrow V - qt$.
6. Jika $B \neq 0$ kita ulangi dari langkah 2.
7. $d \leftarrow A$ dan kita selesai.

Operasi div adalah operasi pembagian dibulatkan kebawah. Setelah langkah 1, *loop invariant* berlaku karena $A = a, B = b, u = 1, v = 0, s = 0, t = 1$ berarti

$$\begin{aligned} \gcd(a,b) &= \gcd(A,B), \\ au + bv &= a \cdot 1 + b \cdot 0 = a = A, \text{ dan} \\ as + bt &= a \cdot 0 + b \cdot 1 = b = B. \end{aligned}$$

Kita harus tunjukkan bahwa jika *loop invariant* berlaku pada langkah 2, *loop invariant* juga berlaku pada langkah 6. Untuk keperluan ini, kita gunakan notasi A', B', u', v', s', t' sebagai nilai A, B, u, v, s, t pada saat langkah 2 dimulai. Langkah 2 dan 3 mendapatkan *quotient* q dan *residue* r, jadi $A' = qB' + r$. Menggunakan teorema 6, kita dapatkan

$$\gcd(a,b) = \gcd(A',B') = \gcd(B',r).$$

Langkah 4 membuat $A = B'$ dan $B = r$, jadi

$$\gcd(a,b) = \gcd(B',r) = \gcd(A,B)$$

setelah langkah 5 (langkah 5 tidak mengubah A dan B). Jadi $\gcd(a,b) = \gcd(A,B)$ berlaku pada langkah 6.

$$\begin{aligned} A &= B' \text{ (dari langkah 4)} \\ &= as' + bt' \text{ (dari } \textit{loop invariant} \text{ pada langkah 2)} \\ &= au + bv \text{ (karena langkah 5 membuat } u = s' \text{ dan } v = t'). \end{aligned}$$

Jadi $au + bv = A$ berlaku pada langkah 6.

$$\begin{aligned} B &= r \text{ (dari langkah 4)} \\ &= A' - qB' \\ &= au' + bv' - qB' \text{ (dari } \textit{loop invariant} \text{ pada langkah 2)} \\ &= au' + bv' - q(as' + bt') \text{ (dari } \textit{loop invariant} \text{ pada langkah 2)} \\ &= a(u' - qs') + b(v' - qt') \\ &= as + bt \text{ (langkah 5 membuat } s = u' - qs' \text{ dan } t = v' - qt'). \end{aligned}$$

Jadi $as + bt = B$ berlaku pada langkah 6. Akibatnya seluruh *loop invariant* berlaku pada langkah 6. Pada langkah 7, kita dapatkan $B = 0$ jadi

$$\gcd(a, b) = \gcd(A, B) = \gcd(A, 0) = A = d$$

dan

$$d = A = au + bv.$$

Jadi algoritma benar mengkalkulasi $d = \gcd(a, b)$ dan mendapatkan u, v dengan $d = au + bv$.

Teorema 8 *Untuk setiap pasangan bilangan bulat a dan b kecuali $a = b = 0$ dengan $d = \gcd(a, b)$ dan bilangan bulat c, persamaan:*

$$c = ax + by \text{ dengan } (x, y \in \mathbf{Z})$$

mempunyai solusi jika dan hanya jika ($\Longleftrightarrow$) c merupakan kelipatan d.

Untuk membuktikan bahwa c harus merupakan kelipatan d, kita gunakan fakta bahwa d membagi a dan b, alhasil d membagi $c = ax + by$. Untuk membuktikan bahwa setiap kelipatan d (sebut saja $c = de$ dengan e bilangan bulat apa saja) merupakan solusi, teorema 7 memberikan $d = au + bv$ untuk sepasang bilangan bulat u dan v, akibatnya $c = de = aue + bve$, jadi dengan $x = ue$ dan $y = ve$ kita dapatkan $c = ax + by$.

Definisi 6 (Koprima) *Sepasang bilangan bulat a dan b disebut koprima (coprime atau relatively prime) jika* $\gcd(a, b) = 1$.

Teorema 9 *Sepasang bilangan bulat a dan b koprima jika dan hanya jika ($\Longleftrightarrow$) ada pasangan bilangan bulat x dan y yang mematuhi persamaan:*

$$ax + by = 1.$$

Untuk membuktikan bahwa ada pasangan bilangan bulat x dan y dimana persamaan $ax+by = 1$ berlaku jika a dan b koprima, cukup menggunakan teorema 7 dengan $u = x$ dan $v = y$. Untuk membuktikan bahwa jika $ax+by = 1$ berarti a dan b koprima, kita gunakan teorema 8 dengan $c = 1$, jadi karena d harus membagi c, maka $d = 1$, yang berarti a dan b koprima.

3.5 Aritmatika Modular

Saat membahas analisa statistik (lihat 2.3.2), contoh *Caesar cipher* digunakan dengan enkripsi dan dekripsi yang rumusnya bersifat aritmatika. Aritmatika adalah matematika pertambahan dan perkalian dengan kemungkinan operasi *inverse* (pembalikan). Untuk aritmatika bilangan bulat, hanya 1 dan -1 yang mempunyai *inverse* perkalian[2], jadi struktur aritmatika bilangan bulat bukan *field* tetapi *ring*. *Domain* aritmatika bilangan bulat bersifat *infinite* (besarnya *domain* bukan merupakan bilangan bulat).

Lain dengan aritmatika bilangan bulat, aritmatika bilangan rasional (*rational numbers*) dan aritmatika bilangan nyata (*real numbers*) mempunyai struktur *field* dimana setiap bilangan kecuali 0 mempunyai *inverse* (setiap elemen yang mempunyai *inverse* disebut *unit*). Dalam struktur *field*, konsep gcd tidak ada artinya karena setiap bilangan kecuali 0 adalah pembagi untuk semua bilangan.

Aritmatika yang banyak digunakan dalam kriptografi adalah apa yang disebut aritmatika modular (*modular arithmetic*). Dalam aritmatika modular, *domain* yang digunakan adalah *subset* dari bilangan bulat dan bersifat *finite* (terbatas, besarnya *domain* merupakan bilangan bulat). Setiap bilangan mempunyai *inverse* pertambahan, dan jika setiap bilangan kecuali 0 mempunyai *inverse* perkalian maka struktur aritmatika disebut *finite field*. Digunakannya aritmatika modular dalam kriptografi adalah karena adanya *inverse* perkalian (terutama jika struktur berupa *field*) dan *domain* yang bersifat *finite*.

Karena *finite field* juga berupa *field*, konsep gcd tidak ada artinya dalam struktur *finite field*. Tetapi gcd dengan bilangan bulat (yang mempunyai struktur *ring*) banyak digunakan dalam membahas struktur *finite field*.

Domain dari aritmatika modular adalah $\{0, 1, 2, \ldots, n-1\}$, dimana n adalah besarnya *domain*. Aritmatika disebut aritmatika modulo n, dengan pertambahan dan perkalian seperti aritmatika biasa jika menghasilkan bilangan yang termasuk dalam *domain*. Jika hasil merupakan bilangan diluar *domain*, maka bilangan harus dikurangi dengan kelipatan n sampai menghasilkan bilangan dalam *domain*.

Tabel 3.1 menunjukkan contoh aritmatika modulo 7. Untuk $5 + 5$, hasilnya adalah 3 karena 10 dikurangi 7 menghasilkan 3.

Proses pengurangan kelipatan n dapat direpresentasikan dengan operasi mod. Menggunakan persamaan 3.1 dari teorema pembagian dengan $b = n$:

$$a = nq + r,$$

operasi mod dapat didefinisikan sebagai berikut:

[2]Selanjutnya jika tidak disebutkan pertambahan atau perkalian maka *inverse* berarti *inverse* perkalian.

Ekspressi	Hasil
$2+3$	5
$5+5$	3
$5 \cdot 6$	2
-3	4
4^{-1}	2

Tabel 3.1: Contoh aritmatika modulo 7

Definisi 7 (mod)

$$a \bmod n = r = a - nq,$$

dengan kata lain a mod n adalah *remainder* atau *residue* dari pembagian a oleh n.

Jadi operasi pertambahan dan perkalian modulo n dapat dipandang sebagai operasi aritmatika bilangan bulat yang dilanjutkan dengan operasi mod pada hasil operasi bilangan bulat. Rumus untuk pertambahan $x + y$ menjadi:

$$x + y = x + y \bmod n. \tag{3.3}$$

dimana operasi disebelah kanan persamaan menggunakan aritmatika bilangan bulat. Rumus untuk perkalian $x \cdot y$ menjadi:

$$x \cdot y = xy \bmod n. \tag{3.4}$$

Dengan menggunakan rumus perkalian untuk aritmatika modulo 7, $5 \cdot 6$ menghasilkan $30 \bmod 7 = 2$ ($a = xy = 30$, $n = 7$, $q = 4$).

Definisi untuk *inverse* pertambahan $-b$ adalah:

$$b + -b = 0.$$

Jadi $-b$ adalah bilangan bulat yang mematuhi persamaan:

$$(b + -b) \bmod n = 0 \text{ dan } 0 \leq -b < n.$$

Karena $0 \leq b < n$, rumus untuk $-b$ menjadi:

$$-b = n - b. \tag{3.5}$$

Jadi -3 menghasilkan $7 - 3 = 4$.

Definisi untuk *inverse* perkalian b^{-1} adalah:

$$b \cdot b^{-1} = 1. \tag{3.6}$$

Jadi b^{-1} adalah bilangan bulat yang mematuhi persamaan:

$$(b \cdot b^{-1}) \bmod n = 1 \text{ dengan } 0 \leq b^{-1} < n$$

Jadi 4^{-1} menghasilkan 2 karena $4 \cdot 2$ menghasilkan $8 \bmod 7 = 1$ ($a = 4 \cdot 2 = 8$, $n = 7$, $q = 1$). Kita akan melihat bagaimana kita dapat mengkalkulasi *inverse* perkalian. Kita definisikan dahulu konsep *congruent modulo n*.

Definisi 8 (Congruence) *Untuk setiap pasangan bilangan bulat a dan b, a congruent dengan b modulo n, dengan notasi*

$$a \equiv b \pmod{n},$$

jika a dan b mempunyai residue yang sama jika dibagi oleh n.

Menggunakan teorema 4 untuk pembagian, ada q' dan q'' dengan:

$$a = q'n + r, \text{ dan}$$

$$b = q''n + r$$

(kedua persamaan menggunakan r karena a dan b mempunyai *residue* yang sama). Jadi jika $a \equiv b \pmod{n}$ maka ada bilangan bulat $q = q' - q''$ yang mematuhi persamaan $a - b = qn$ (perbedaan antara a dan b adalah kelipatan n).

Konsep yang sering digunakan untuk menjelaskan aritmatika modular adalah konsep *congruence classes*. Relasi *congruent modulo n* adalah relasi ekuivalen yang mempartisi himpunan dari semua bilangan bulat ($\mathbf{Z}$) menjadi n kelas ekuivalen yang disebut juga *congruence classes*:

$$\begin{aligned}
[0] &= \{\ldots, -2n, -n, 0, n, 2n, \ldots\}, \\
[1] &= \{\ldots, -2n+1, -n+1, 1, n+1, 2n+1, \ldots\}, \\
[2] &= \{\ldots, -2n+2, -n+2, 2, n+2, 2n+2, \ldots\}, \\
&\ldots \\
[n-1] &= \{\ldots, -n-1, -1, n-1, 2n-1, 3n-1, \ldots\}.
\end{aligned}$$

Rumus untuk *congruence class* $[i]$ dengan modulus n adalah:

$$[i] = \{j \in \mathbf{Z} | i \equiv j \pmod{n}\} = \{jn + i | j \in \mathbf{Z}\}. \tag{3.7}$$

Dengan modulus n, hanya ada n kelas, tidak ada kelas lain. Sebagai contoh

$$[n] = \{\ldots, -n, 0, n, 2n, 3n, \ldots\} = [0].$$

Secara umum

$$[a] = [b] \Longleftrightarrow a \equiv b \pmod{n}. \tag{3.8}$$

Himpunan *congruence classes* mempunyai struktur *quotient ring* (akan dibahas di bab 5), dengan notasi $\mathbf{Z}/n\mathbf{Z}$ untuk bilangan bulat n. Sebagai contoh, untuk $n = 7$ ada 7 kelas:

$$\begin{aligned}
[0] &= \{\ldots, -14, -7, 0, 7, 14, \ldots\},\\
[1] &= \{\ldots, -13, -6, 1, 8, 15, \ldots\},\\
[2] &= \{\ldots, -12, -5, 2, 9, 16, \ldots\},\\
[3] &= \{\ldots, -11, -4, 3, 10, 17, \ldots\},\\
[4] &= \{\ldots, -10, -3, 4, 11, 18, \ldots\},\\
[5] &= \{\ldots, -9, -2, 5, 12, 19, \ldots\},\\
[6] &= \{\ldots, -8, -1, 6, 13, 20, \ldots\}.
\end{aligned}$$

Setiap elemen dalam kelas adalah representatif kelas, jadi setiap bilangan yang berbeda kelipatan 7 dari 2 (contohnya -5, 2 dan 9) merupakan representatif dari [2].

Aritmatika dapat didefinisikan terhadap kelas. Aritmatika dilakukan dahulu terhadap representatif kelas (elemen mana saja dalam kelas dapat digunakan). Hasil aritmatika bilangan kemudian digunakan untuk menentukan kelas yang merupakan hasil aritmatika kelas. Sebagai contoh, untuk $[2] + [3]$, kita dapat tambahkan -5 (yang merupakan representatif [2]) dengan 3 (yang merupakan representatif [3]) untuk mendapatkan -2, yang merupakan representatif dari [5]. Jadi $[2] + [3] = [5]$. Secara formal, rumus untuk pertambahan adalah:

$$[a] + [b] = [a + b], \tag{3.9}$$

rumus untuk *inverse* pertambahan adalah:

$$-[a] = [-a], \tag{3.10}$$

dan rumus untuk perkalian adalah:

$$[a] \cdot [b] = [a \cdot b]. \tag{3.11}$$

Tidak semua kelas mempunyai *inverse* perkalian, kita tunda pembahasan rumus untuk mencari *inverse* perkalian. Untuk meyakinkan bahwa operasi telah didefinisikan dengan baik (*well-defined*), kita gunakan teorema:

Teorema 10 *Untuk modulus $n > 1$, jika $a' \equiv a$ dan $b' \equiv b$, maka $a' + b' \equiv a + b$, $-a' \equiv -a$ dan $a' \cdot b' \equiv a \cdot b$.*

Pembuktian teorema adalah sebagai berikut: jika $a' \equiv a \pmod{n}$ maka terdapat bilangan bulat k sehingga $a' = a + kn$. Demikian juga untuk $b' \equiv b \pmod{n}$ terdapat l sehingga $b' = b + ln$. Jadi

$$\begin{aligned}
a' + b' = a + b + (k + l)n &\equiv a + b \pmod{n},\\
-a' = -a - kn &\equiv -a \pmod{n},\\
a' \cdot b' = a \cdot b + (al + bk + kln)n &\equiv a \cdot b \pmod{n}.
\end{aligned}$$

Teorema 10 menyatakan bahwa operasi terhadap kelas tidak tergantung representatif kelas yang dipilih, jadi operasi telah didefinisikan dengan baik.

Selanjutnya tidak terlalu sukar untuk menunjukkan bahwa aritmatika *congruence classes* mempunyai struktur aljabar *ring* dengan elemen [0] dan [1]. *Closure* untuk + dan · didapat karena bilangan apapun yang dihasilkan aritmatika bilangan bulat akan menghasilkan satu diantara *congruence classes* yang ada.
Associativity untuk +:

$$\begin{aligned}[a] + ([b] + [c]) &= [a] + [b + c] \\ &= [a + (b + c)] \\ &= [(a + b) + c] \\ &= [a + b] + [c] \\ &= ([a] + [b]) + [c].\end{aligned}$$

Identity untuk +:

$$[a] + [0] = [a + 0] = [a].$$

Commutativity untuk +:

$$[a] + [b] = [a + b] = [b + a] = [b] + [a].$$

Inverse untuk +:

$$[a] + (-[a]) = [a] + [-a] = [a + (-a)] = [0].$$

Associativity untuk ·:

$$\begin{aligned}[a] \cdot ([b] \cdot [c]) &= [a] \cdot [b \cdot c] \\ &= [a \cdot (b \cdot c)] \\ &= [(a \cdot b) \cdot c] \\ &= [a \cdot b] \cdot [c] \\ &= ([a] \cdot [b]) \cdot [c].\end{aligned}$$

Identity untuk ·:

$$[a] \cdot [1] = [a \cdot 1] = [a].$$

Commutativity untuk ·:

$$[a] \cdot [b] = [a \cdot b] = [b \cdot a] = [b] \cdot [a].$$

Distributivity:

$$\begin{aligned}
[a] \cdot ([b] + [c]) &= [a] \cdot [b + c] \\
&= [a \cdot (b + c)] \\
&= [(a \cdot b) + (a \cdot c)] \\
&= [a \cdot b] + [a \cdot c] \\
&= ([a] \cdot [b]) + ([a] \cdot [c]).
\end{aligned}$$

Kembali ke aritmatika modular, kita dapat menggunakan konsep aritmatika *congruence classes* untuk menjelaskannya. Karena aritmatika modular tidak tergantung pada representatif kelas yang dipilih, untuk setiap kelas kita dapat memilih representatif $0 \leq r < n$, yaitu *residue* modulo n. Jadi aritmatika modular dapat dianggap sebagai aritmatika *congruence classes* dengan *residue* modulo n digunakan sebagai representatif setiap kelas.

Sebagai contoh, untuk mencari hasil aritmatika modulo 7, kita bisa cari dahulu hasil aritmatika bilangan bulat, kemudian cari kelas dari hasil aritmatika bilangan bulat, dan ahirnya cari bilangan r dalam kelas yang sama yang mematuhi $0 \leq r < 7$ (r adalah *residue* modulo 7 representatif kelas). Contoh kongkrit, untuk *inverse* pertambahan 3, hasil aritmatika bilangan bulat adalah -3, dan hasil pencarian r dalam kelas yang berisi -3 dengan $0 \leq r < 7$ menghasilkan $r = 4$. (Kita dapat juga mengkalkulasi $r = -3 \bmod 7 = 4$.)

Teorema berikut kita gunakan sebagai dasar cara mengkalkulasi *inverse* perkalian dalam aritmatika modular.

Teorema 11 (Inverse) *Suatu bilangan a mempunyai inverse modulo n jika dan hanya jika ($\Longleftrightarrow$)* $\gcd(a, n) = 1$.

Untuk membuktikan teorema ini, definisi *inverse* mengatakan bahwa jika a mempunyai *inverse* (sebut saja x) berarti $ax \equiv 1 \pmod{n}$. Jadi ada bilangan bulat q yang mematuhi persamaan $ax - 1 = qn$ atau $ax - nq = 1$. Menggunakan teorema 9 dengan $b = n$ dan $y = -q$ berarti a koprima dengan n jadi $\gcd(a, n) = 1$. Sebaliknya jika $\gcd(a, n) = 1$, maka berdasarkan teorema 7, ada pasangan u dan v yang mematuhi $1 = au + nv$, jadi $au \equiv 1 \pmod{n}$, yang berarti a mempunyai *inverse* yaitu u. *Extended Euclidean algorithm* (lihat 3.4) dapat digunakan untuk mendapatkan *inverse* u.

Tentunya jika n merupakan bilangan prima, untuk setiap $a \not\equiv 0 \pmod{n}$ (termasuk $0 < a < n$) kita dapatkan $\gcd(a, n) = 1$, sehingga setiap $a \not\equiv 0 \pmod{n}$ mempunyai *inverse*. Jadi aritmatika modulo bilangan prima n mempunyai struktur *finite field*.

3.6 Ringkasan

Bab ini dimulai dengan pembahasan struktur-struktur aljabar, antara lain *group*, *monoid*, *ring* dan *field*. Struktur-struktur tersebut banyak digunakan dalam matematika untuk kriptografi. Prinsip induksi dibahas karena diperlukan untuk pembuktian berbagai teorema. Konsep gcd dan kalkulasinya menggunakan algoritma *Euclid* juga dibahas dengan rinci. Konsep aritmatika modular dijelaskan menggunakan struktur aljabar dan konsep gcd. Antara lain, *inverse* dalam aritmatika modular dapat dikalkulasi menggunakan *extended Euclidean algorithm*.