II4021 Kriptografi

Enkripsi Homomorfik

Oleh: Rinaldi M

Program Studi Sistem dan Teknologi Informasi
Sekolah Teknik Elektro dan Informatika
ITB - 2026

Homomorphic Encryption

Komputasi pada cipherteks

- Bagaimana cara memanipulasi cipherteks sehingga ketika didekripsi hasilnya sama seperti mengubah plainteksnya?
- Cara konvensional:

  (i) dekripsi cipherteks menjadi plainteks,

  (ii) ubah plainteks seperti yang diinginkan,

  (iii) enkripsi kembali plainteks hasil perubahan menjadi cipherteks yang baru.

  $\rightarrow$ Tidak aman jika hasil Langkah (ii) berhasil disadap oleh pihak lawan

  Contoh: kasus e-voting

Enkripsi Homomorfik (EH)

- *Homomorphic encryption*

- EH melakukan komputasi pada cipherteks tanpa harus mendekripsi cipherteks tersebut terlebih dahulu,

- namun hasil komputasi pada cipherteks, bila didekripsi, memberikan hasil yang tepat sama bila komputasi yang serupa dilakukan pada plainteksnya.

- Misalkan E menyatakan fungsi enkripsi. Fungsi enkripsi E dikatakan **homomorfik** jika diberikan E(x) dan E(y), maka orang dapat memperoleh E(x ◊ y) tanpa mendekripsi x dan y untuk beberapa operasi ◊ (Rivest, 2002).

Homomorfik aditif vs Homomorfik multiplikatif

- Sebuah algoritma enkripsi homomorfik dikatakan aditif (*additive*) jika

$$E(x + y) = E(x) \otimes E(y)$$

E adalah fungsi enkripsi, *x* dan *y* adalah plainteks, dan $\otimes$ menyatakan operasi yang bergantung pada *cipher* yang digunakan (operasi +, $\times$, atau operasi lainnya).

- Sebuah algoritma enkripsi homomorfik dikatakan multiplikatif jika

$$E(x \cdot y) = E(x) \otimes E(y)$$

E adalah fungsi enkripsi, *x* dan *y* adalah plainteks, dan $\otimes$ menyatakan operasi yang bergantung pada *cipher* yang digunakan (operasi +, $\times$, atau operasi lainnya).

Jenis-jenis enkripsi homomorfik

1. **Enkripsi homomorfik sebagian** (*partially homomorphic encryption*) Enkripsi homomorfik sebagian hanya memungkinkan satu operasi aritmetika pada cipherteks, yaitu operasi penjumlahan saja atau operasi perkalian saja.

2. **Enkripsi homomorfik penuh** (*fully homomorphic encryption*), operasi penjumlahan dan operasi perkalian dapat dilakukan terhadap cipherteks

Enkripsi homomorfik sebagian

- Enkripsi homomorfik sebagian hanya memiliki satu operasi saja pada cipherteks, yaitu operasi penjumlahan saja atau operasi perkalian saja, namun tidak keduanya.

- Jadi, enkripsi homomorfik hanya memiliki satu sifat, yaitu sifat aditif atau sifat multiplikatif. Dua buah cipherteks hanya dapat dijumlahkan saja atau dikalikan saja.

- Tiga buah algoritma enkripsi homomorfik sebagian: (1) algoritma RSA, (2) algoritma ElGamal, dan (3) algoritma Paillier.

Enkripsi Homomorfik dengan Algoritma RSA

- Algoritma RSA bersifat multiplikatif, yaitu hasil kali dua buah cipherteks apabila didekripsi hasilnya sama dengan mengalikan kedua plainteksnya

  Sifat multiplikatif: $E(x \cdot y) = E(x) \cdot E(y)$ dan $D(E(x) \cdot E(y)) = x \cdot y$

- Algoritma RSA:

  Enkripsi: $E(m) = c = m^e \bmod n$

  Dekripsi: $D(c) = m = c^d \bmod n$

- Misalkan

  plainteks: m_1 dan m_2,

  cipherteks: c_1 dan c_2,

  $E(m_1) = c_1 = m_1^e \bmod n$

  $E(m_2) = c_2 = m_2^e \bmod n$

- Kalikan kedua cipherteks:

$$E(m_1) \cdot E(m_2) = c_1 \cdot c_2 = (m_1^e \bmod n) \cdot (m_2^e \bmod n)$$
$$= (m_1 \cdot m_2)^e \bmod n$$

Sedangkan $E(m_1 \cdot m_2) = (m_1 \cdot m_2)^e \bmod n$.

- Hasil ini memperlihatkan bahwa algoritma RSA memiliki sifat multiplikatif, yaitu

$$E(m_1 \cdot m_2) = E(m_1) \cdot E(m_2)$$

- Sekarang, akan ditunjukkan bahwa hasil dekripsi terhadap $c_1 \cdot c_2$ sama dengan perkalian kedua plainteksnya, yaitu

$$D(c_1 \cdot c_2) = (m_1 \cdot m_2) \bmod n.$$

Bukti:

$$D(c_1 \cdot c_2) = (c_1 \cdot c_2)^d \bmod n$$
$$= ((m_1 \cdot m_2)^e)^d \bmod n$$
$$= (m_1 \cdot m_2)^{ed} \bmod n$$

Karena $ed = k\phi(n) + 1$ (lihat penurunan rumus RSA), maka

$$= (m_1 \cdot m_2)^{k\phi(n)+1} \bmod n$$
$$= (m_1 \cdot m_2)\,(m_1 \cdot m_2)^{k\phi(n)} \bmod n$$

Karena $a^{k\phi(n)} \equiv 1 \pmod{n}$, maka $(m_1 \cdot m_2)^{k\phi(n)} \equiv 1 \pmod{n}$, sehingga

$$= (m_1 \cdot m_2) \cdot 1 \bmod n$$
$$= (m_1 \cdot m_2) \bmod n$$

Oleh karena itu, kita berhasil menunjukkan bahwa

$$D(c_1 \cdot c_2) = (m_1 \cdot m_2) \bmod n$$

Contoh 1: Misalkan kunci publik Alice adalah (n = 3337, e = 79) dan kunci privatnya d = 1019. Misalkan m_1 = 2671 dan m_2 = 1800.

$E(m_1) = c_1 = m_1^e \bmod n = 2671^{79} \bmod 3337 = 2081$

$E(m_2) = c_2 = m_2^e \bmod n = 1800^{79} \bmod 3337 = 338$

$E(m_1) \cdot E(m_2) = c_1 \cdot c_2 = 2081 \cdot 338 = 703378$

$D(c_1 \cdot c_2) = (c_1 \cdot c_2)^d \bmod n = (703378)^{1019} \bmod 3337 = 2520$

Hasil terakhir ini, 2520, sama dengan mengalikan 2671 dengan 1800 dalam modulus 3337, yaitu

$m_1 \cdot m_2 = (2671 \cdot 1800) \bmod 3337 = 4807800 \bmod 3337 = 2520$

Jadi, $D(c_1 \cdot c_2) = m_1 \cdot m_2 = 2520$, artinya hasil kali dua buah cipherteks apabila didekripsi hasilnya sama dengan mengalikan kedua plainteksnya.

Untuk menunjukkan RSA adalah enkripsi homomorfik yang bersifat multiplikatif, maka

$E(m_1 \cdot m_2) = E(2520) = 2520^{79} \bmod 3337 = 2608$

$= E(m_1) \cdot E(m_2) = (2081 \cdot 338) \bmod 3337 = 2608$

yang menunjukkan bahwa $E(m_1 \cdot m_2) = E(m_1) \cdot E(m_2)$

Enkripsi Homomorfik dengan Algoritma ElGamal

- Seperti RSA, algoritma ElGamal juga bersifat multiplikatif.

  Enkripsi: $(a, b) = (g^k \bmod p, y^k m \bmod p)$

  Dekripsi: $m = (b/a^x) \bmod p$

- Misalkan plainteks: m_1 dan m_2, cipheteksnya: (a_1, b_1) dan (a_2, b_2)

  $E(m_1) = (a_1, b_1) = (g^{k1} \bmod p, y^{k1} m_1 \bmod p)$

  $E(m_2) = (a_2, b_2) = (g^{k2} \bmod p, y^{k2} m_2 \bmod p)$

- Kalikan cipherteks (a_1, b_1) dan (a_2, b_2) sebagai berikut:

$$
\begin{aligned}
E(m_1) \cdot E(m_2) &= (a_1, b_1) \cdot (a_2, b_2) \\
&= (a_1 \cdot a_2,\ b_1 \cdot b_2) \\
&= (g^{k1} \cdot g^{k2},\ y^{k1} m_1 \cdot y^{k2} m_2) \bmod p \\
&= (g^{k1+k2},\ y^{k1+k2} \cdot m_1 \cdot m_2) \bmod p
\end{aligned}
$$

- Sedangkan

$$E(m_1 \cdot m_2) = (a, b) = (g^k, y^k \cdot m_1 \cdot m_2) \bmod p, \text{ dalam hal ini } k = k_1 + k_2$$
$$= (g^{k1 + k2}, y^{k1 + k2} \cdot m_1 \cdot m_2) \bmod p$$
$$= E(m_1) \cdot E(m_2)$$

- Hasil ini memperlihatkan algoritma ElGamal memiliki sifat multiplikatif, yaitu

$$E(m_1 \cdot m_2) = E(m_1) \cdot E(m_2)$$

- Sekarang, akan ditunjukkan bahwa hasil dekripsi terhadap hasil kali dua cipherteks sama dengan perkalian kedua plainteksnya:

$$D((a_1, b_1) \cdot (a_2, b_2)) = (m_1 \cdot m_2) \bmod p$$

Bukti:

$$D((a_1, b_1) \cdot (a_2, b_2)) = D(g^{k1+k2},\ m_1 \cdot m_2\, y^{k1+k2})$$

$$= \frac{m_1 \cdot m_2 \cdot y^{k1+k2}}{(g^{k1+k2})^x} \bmod p$$

Mengingat $g^x \equiv y \pmod p$, maka

$$= \frac{m_1 \cdot m_2 \cdot (g^x)^{k1+k2}}{(g^{k1+k2})^x} \bmod p$$

$$= \frac{m_1 \cdot m_2 \cdot (g^{k1+k2})^x}{(g^{k1+k2})^x} \bmod p$$

$$= m_1 \cdot m_2 \bmod p$$

Oleh karena itu, kita berhasil menunjukkan bahwa

$$D((a_1, b_1) \cdot (a_2, b_2)) = (m_1 \cdot m_2) \bmod p$$

Enkripsi Homomorfik dengan Algoritma Paillier

- Algoritma Paillier termasuk ke dalam algoritma kriptografi kunci-publik. Algoritma ini dikembangkan oleh Pascal Paillier pada tahun 1999 (Paillier, 1999).

- Keamanan algoritma ini didasarkan pada sulitnya memecahkan persoalan residu ke-*n* (*composite residuosity problem*). Persoalan residu kelas ke-*n* adalah sebagai berikut:

  Diberikan bilangan komposit *n* dan bilangan bulat *z*. Bilangan *z* dikatakan residu ke-*n* modulo n^2 jika terdapat sebuah nilai *y* sedemikian sehingga $z \equiv y^n \pmod{n^2}$.

  - Contoh: $n = 8$ dan $z = 33$. Maka, carilah nilai *y* sedemikian sehingga $33 \equiv y^8 \pmod{64}$. Nilai *y* yang memenuhi adalah $y = 5$, sebab $33 \equiv 5^8 \pmod{64}$. Persoalan ini akan semakin sukar bila *n* dan *z* bernilai besar.

Pascal Paillier

https://www.researchgate.net/profile/Pascal-Paillier

https://twitter.com/pascal_paillier

https://raais.co/speakers-2020-pascal-paillier-zama

Prosedur Membangkitkan Pasangan Kunci

- Misalkan Alice ingin membangkitkan pasangan kunci privat dan kunci publiknya. Prosedur yang dilakukan Alice adalah sebagai berikut:
  1. Alice memilih dua buah bilangan prima sembarang, p dan q yang memenuhi syarat PBB(pq, $(p-1)(q-1)$) = 1. PBB = pembagi bersama terbesar = *greatest common divisor* = *gcd*.
  2. Alice menghitung $n = p \cdot q$ dan λ = KPK($p-1$, $q-1$). KPK = kelipatan persekutuan terkecil atau *lcm* = *lowest common multiple*.
  3. Pilih sembarang bilangan bulat g, dengan $g < n^2$.
  4. Hitung $\mu = (L(g^{\lambda} \bmod n^2))^{-1} \bmod n$, dengan fungsi $L(x) = (x-1)/n$.

- Hasil dari prosedur di atas:

  Kunci publik Alice adalah pasangan (g, n).

  Kunci privat Alice adalah (λ, μ). .

Contoh 2: Alice menghitung kunci privat dan kunci publiknya sebagai berikut:

a) Misalkan $p = 7$ dan $q = 11$. Keduanya memenuhi syarat PBB(pq, $(p-1)(q-1)$) = PBB(77, 60) = 1.
b) Hitung $n = p \cdot q = 77$ dan λ = KPK($p-1$, $q-1$) = KPK(6, 10) = 30.
c) Pilih g secara acak, $g < n^2$, misalkan $g = 5652$
d) Hitung $\mu = (L(g^{\lambda} \bmod n^2))^{-1} \bmod n$, dengan fungsi $L(x) = (x-1)/n$.

$L(g^{\lambda} \bmod n^2) = L(5652^{30} \bmod 77^2) = L(5652^{30} \bmod 5929) = L(3928)$

$L(3928) = (3928 - 1)/77 = 3927/77 = 51$

$\mu = 51^{-1} \pmod{77} = 74$

- Diperoleh kunci publik dan kunci privat Alice adalah sebagai berikut:

Kunci publik: $(g, n) = (5652, 77)$

Kunci privat: $(\lambda, \mu) = (30, 74$

Prosedur Enkripsi

1. Misalkan m adalah pesan yang akan dienkripsi, dengan syarat $0 \leq m < n$.
2. Pilih bilangan bulat acak r dengan syarat $0 \leq r < n$ dan PBB(r, n)= 1.
3. Hitung cipherteks dari m dengan persamaan $c = g^m \cdot r^n \bmod n^2$. Dalam hal ini, c adalah residu ke-n dalam modulus n^2, dilambangkan dengan $[c]_g$.

Prosedur Dekripsi

1. Misalkan c cipherteks yang akan didekripsi.
2. Hitung plainteks dari c dengan persamaan $m = L(c^\lambda \bmod n^2) \cdot \mu \bmod n$ atau dengan persamaan lain yaitu

$$m = \frac{L(c^\lambda \bmod n^2)}{L(g^\lambda \bmod n^2)} \bmod n^2$$

Contoh 3: Misalkan pesan yang dienkripsi adalah $m = 42$. Pilih $r = 23$, karena $0 \leq r < n$ dan PBB(r, n) = 1.

a) Enkripsi: cipherteks dihitung menggunakan kunci publik (g, n) = (5652, 77) dan dengan persamaan:

$$c = g^m \cdot r^n \cdot \bmod n^2$$
$$= 5652^{42} \cdot 23^{77} \bmod 77^2$$
$$= 5652^{42} \cdot 23^{77} \bmod 5929$$
$$= 5652^{42} \bmod 5929 \cdot 23^{77} \bmod 5929$$
$$= 4019 \cdot 606 \bmod 5929$$
$$= 4624$$

- Jadi, cipherteks untuk pesan $m = 42$ adalah $c = 4624$.

b) Dekripsi: plainteks dihitung dengan kunci privat (λ, μ) = (30, 74) dan dengan persamaan:

$$m = L(c^{\lambda} \bmod n^2) \cdot \mu \bmod n$$
$$= L(4624^{30} \bmod 77^2) \cdot 74 \bmod 77$$
$$= L(4624^{30} \bmod 5929) \cdot 74 \bmod 77$$
$$= L(4624^{30} \bmod 5929) \cdot 74 \bmod 77$$
$$= L(4852) \cdot 74 \bmod 77$$
$$= (4852 - 1)/77 \cdot 74 \bmod 77$$
$$= 63 \cdot 74 \bmod 77$$
$$= 42$$

Sifat Homomorfik di dalam Algoritma Paillier

- Algoritma Paillier merupakan algoritma enkripsi homomorfik yang bersifat aditif, yaitu jika dua buah cipherteks dikalikan maka hasil dekripsinya sama dengan penjumlahan kedua plainteksnya.

  Sifat aditif: $E(x + y) = E(x) \cdot E(y)$ dan $D(E(x) \cdot E(y)) = x + y$

- Misalkan dua buah plainteks adalah m_1 dan m_2, hasil enkripsinya masing-masing adalah c_1 dan c_2, yang dalam hal ini,

  $$E(m_1, r_1) = c_1 = g^{m1} \cdot r_1^{\,n} \bmod n^2 \qquad (1)$$
  $$E(m_2, r_2) = c_2 = g^{m2} \cdot r_2^{\,n} \bmod n^2 \qquad (2)$$

  Kalikan cipherteks c_1 dan c_2 sebagai berikut:

  $$E(m_1, r_1) \cdot E(m_2, r_2) = c_1 \cdot c_2 = g^{m1} \cdot r_1^{\,n} \cdot g^{m2} \cdot r_2^{\,n} \bmod n^2$$
  $$= g^{m1 + m2} (r_1 \cdot r_2)^n \bmod n^2 \qquad (3)$$

- Sedangkan

$$E(m_1 + m_2, r) = g^m r^n \mod n^2, \text{ dalam hal ini } m = m_1 + m_2 \text{ dan } r = r_1 \cdot r_2$$
$$= g^{m1 + m2} (r_1 \cdot r_2)^n \mod n^2$$
$$= E(m_1, r_1) \cdot E(m_2, r_2)$$

- Hasil ini memperlihatkan algoritma Paillier memiliki sifat aditif, yaitu

$$E(m_1 + m_2) = E(m_1) \cdot E(m_2)$$

- Sekarang, akan ditunjukkan bahwa hasil dekripsi terhadap $c_1 \cdot c_2$ sama dengan penjumlahan kedua plainteksnya, yaitu akan dibuktikan:

$$D(c_1 \cdot c_2) = (m_1 + m_2) \bmod n \qquad (4)$$

Bukti*:*

Tinjau $(1 + n) \in Z^*{}_n{}^2$. ($Z^*{}_n{}^2$ = himpunan bilangan bulat tak-negatif = $\{1, 2, \ldots, n^2\}$)

Perpangkatan $(1 + n)$ di dalam $Z^*{}_n{}^2$ adalah sebagai berikut:

$$(1 + n)^2 = 1 + 2n + n^2 \equiv 1 + 2n \pmod{n^2}$$

$$(1 + n)^3 = 1 + 3n + 3n^2 + n^3 \equiv 1 + 3n \pmod{n^2}$$

$$(1 + n)^q = 1 + qn + [\text{suku-suku berderajat lebih tinggi}] \equiv 1 + qn \pmod{n^2} \qquad (5)$$

Dekripsi $c_1 \cdot c_2$ adalah

$$D(c_1 \cdot c_2) = L((c_1 \cdot c_2)^{\lambda} \bmod n^2) \cdot \mu \bmod n$$

Berdasarkan persamaan (3),

$$(c_1 \cdot c_2)^{\lambda} \equiv g^{\lambda(m1 + m2)} \cdot (r_1 \cdot r_2)^{\lambda n} \mod n^2 \equiv g^{\lambda (m1 + m2)} \mod n^2 \qquad (6)$$

menurut Teorema Carmichael (O'Keeffe, 2008).

Kemudian, $g^{\lambda} \equiv (1 + n)^{\lambda[g]}{}_{(1 + n)} \mod n^2$. Substitusikan ini ke dalam (6), diperoleh

$$(c_1 \cdot c_2)^{\lambda} \equiv (1 + n)^{\lambda[g]}{}_{(1 + n)} \cdot (m1 + m2) \mod n^2 \qquad (7)$$

Menurut (5),

$$(1 + n)^{\lambda[g]}{}_{(1 + n)} \cdot (m1 + m2) \equiv 1 + (m1 + m2) \cdot \lambda[g](_{1 + n)} \cdot n \mod n^2$$

Jadi,

$$L((c_1 \cdot c_2)^{\lambda} \mod n^2) \equiv (m1 + m2) \cdot \lambda[g]_{(1 + n)} \cdot \mod n \qquad (8)$$

sedangkan μ adalah balikan dari $\lambda[g]_{(1 + n)}$, atau $\mu = (\lambda[g]_{(1 + n)})^{-1}$.

Maka, dengan mengalikan (8) dengan μ diperoleh

$$D(c_1 \cdot c_2) = L((c_1 \cdot c_2)^{\lambda} \bmod n^2) \cdot \mu \bmod n$$
$$= (m_1 + m_2) \cdot \lambda[g]_{(1+n)} \cdot (\lambda[g]_{(1+n)})^{-1} \bmod n$$
$$= (m_1 + m_2) \bmod n$$

Jadi, kita berhasil menunjukkan bahwa

$$D(c_1 \cdot c_2) = (m_1 + m_2) \bmod n.$$

Contoh 4: Dari Contoh 3 telah diperoleh hasil enkripsi $m_1 = 42$ dengan $r_1 = 23$ adalah cipherteks $c_1 = 4624$. Misalkan $m_2 = 29$ dan $r_2 = 30$ dan setelah dihitung cipherteksnya adalah $c_2 = 1539$.

Perkalian kedua cipherteks adalah

$$c_1 \cdot c_2 = 4624 \cdot 1539 = 7116336 \bmod 5929 = 1536$$

Hasil ini sama dengan hasil enkripsi penjumlahan kedua buah plainteksnya sbb:

$$m_1 + m_2 = 42 + 29 = 71 \text{ dan } r = r_1 \cdot r_2 = (23)(30) = 690$$

$$\begin{aligned} E(m_1 + m_2 , r) = E(71, 690) &= g^m \cdot r^n \cdot \bmod n^2 \\ &= 5652^{71} \cdot (690)^{77} \bmod 77^2 \\ &= 5652^{71} \bmod 5929 \cdot 690^{77} \bmod 5929 \\ &= 3540 \cdot 2774 \bmod 5929 \\ &= 1536 \end{aligned}$$

Dekripsi $c_1 \cdot c_2$ adalah

$$D(c_1 \cdot c_2) = L((c_1 \cdot c_2)^{\lambda} \bmod n^2) \cdot \mu \bmod n$$
$$= L(1536^{30} \bmod 77^2) \cdot 74 \bmod 77$$
$$= L(1536^{30} \bmod 5929) \cdot 74 \bmod 77$$
$$= L(1536^{30} \bmod 5929) \cdot 74 \bmod 77$$
$$= L(155) \cdot 74 \bmod 77$$
$$= (155 - 1)/77 \cdot 74 \bmod 77$$
$$= 2 \cdot 74 \bmod 77$$
$$= 148 \bmod 77$$
$$= 71$$

Hasil dekripsi di atas sama dengan penjumlahan dua buah plainteks dalam modulus 77, yaitu

$$(m1 + m2) \bmod n = (42 + 29) \bmod 77 = 71 \bmod 77 = 71$$

Jadi, $D(c_1 \cdot c_2) = (m_1 + m_2) \bmod n = 71$.

Penggunaan Enkripsi Homomorfik Algoritma Paillier di dalam *E-voting*

- Misalkan terdapat 3 orang kandidat: X, Y, dan Z, dan tiga orang pemilih: V_1, V_2, V_3 .

	X	Y	Z
V_1	1	0	0
V_2	0	1	0
V_3	1	0	0
Total	2	1	0

Plainteks

	X	Y	Z
V_1	C_{1X}	C_{1Y}	C_{1Z}
V_2	C_{2X}	C_{2Y}	C_{2Z}
V_3	C_{3X}	C_{3Y}	C_{3Z}
Total	C_X	C_Y	C_Z

Cipherteks

Gambar 18.1 Tabel pemungutan suara di dalam *e-voting* (Rivest, 2002)

- $C_{1X,}$ C_{1Y}, dan lain-lain menyatakan suara yang diberikan oleh pemilih. Suara ini dienkripsi dengan menggunakan kunci publik panitia pemilihan. Kunci privat hanya diketahui oleh panitia pemilihan.

- Dengan algoritma Paillier, 0 yang sama tidak selalu dienkripsi menjadi cipherteks yang sama (tergantung nilai *r* yang digunakan, sebab *r* selalu berbeda setiap mengenkripsi plainteks).

- Dengan demikian, pihak lawan tidak dapat menyimpulkan apakah dua orang pemilih memberikan suara yang sama dengan melihat apakah cipherteksnya sama. Dengan demikian kerahasiaan pemilih terjamin.

- Untuk alasan ini, maka RSA tidak cocok untuk aplikasi *e-voting* (Rivest, 2012), sebab plainteks yang sama selalu dienkripsi menjadi cipherteks yang sama (ingat $c = m^e \bmod n$, karena kunci publik selalu sama untuk mengenkripsi setiap suara, maka angka 0 dan 1 selalu dienkripsi menjadi cipherteks yang sama pula).

- Untuk menghitung total suara setiap kandidat, maka semua cipherteks pada setiap kolom dikalikan:

$$C_X = C_{1X} \cdot C_{2X} \cdot C_{3X}$$
$$C_Y = C_{1Y} \cdot C_{2Y} \cdot C_{3Y}$$
$$C_Z = C_{1Z} \cdot C_{2Z} \cdot C_{3Z}$$

	X	Y	Z
V_1	C_{1X}	C_{1Y}	C_{1Z}
V_2	C_{2X}	C_{2Y}	C_{2Z}
V_3	C_{3X}	C_{3Y}	C_{3Z}
Total	C_X	C_Y	C_Z

- Menurut sifat homomorfik pada algoritma Paillier, perkalian cipherteks sama dengan menjumlahkan plainteksnya:

$$C_X = M_{1X} + M_{2X} + M_{3X}$$
$$C_Y = M_{1Y} + M_{2Y} + M_{3Y}$$
$$C_Z = M_{1Z} + M_{2Z} + M_{3Z}$$

	X	Y	Z
V_1	1	0	0
V_2	0	1	0
V_3	1	0	0
Total	2	1	0

M_{ij} menyatakan plainteks pada baris *i* dan kolom *j*.

- Dengan melakukan komputasi hanya pada cipherteks maka suara yang diberikan oleh pemilih dapat terjamin kerahasiaannya.

- Begitu pula total suara sementara seorang kandidat juga tetap rahasia sehingga pemilih lain maupun panitia pemilihan tidak mengetahui hasil sementara ini.

Enkripsi Homomorfik Penuh

- Enkripsi homomorfik penuh (*fully homomorphic encryption* atau FHE) artinya enkripsi yang memiliki sifat aditif dan multiplikatif sekaligus.

- Hasil penjumlahan atau perkalian dua buah cipherteks bila didekripsi hasilnya sama dengan penjumlahan atau perkalian plainteksnya.

- Algoritma kriptografi kunci-publik yang telah kita bicarakan sejauh ini (RSA, ElGamal, dan Paillier) tidak memenuhi kriteria enkripsi homomorfik penuh.

- Selama 30 tahun setelah Rivest, Adleman, dan Dertouzous menyajikan tantangan konstruksi skema FHE, namun tantangan ini tetap tidak terpecahkan (Wu, 2015).

- Tahun 2005 Boneh, Goh, dan Nissim mengembangkan skema FHE, namun skema tersebut hanya mendukung satu operasi perkalian saja dan banyak operasi penjumlahan cipherteks.

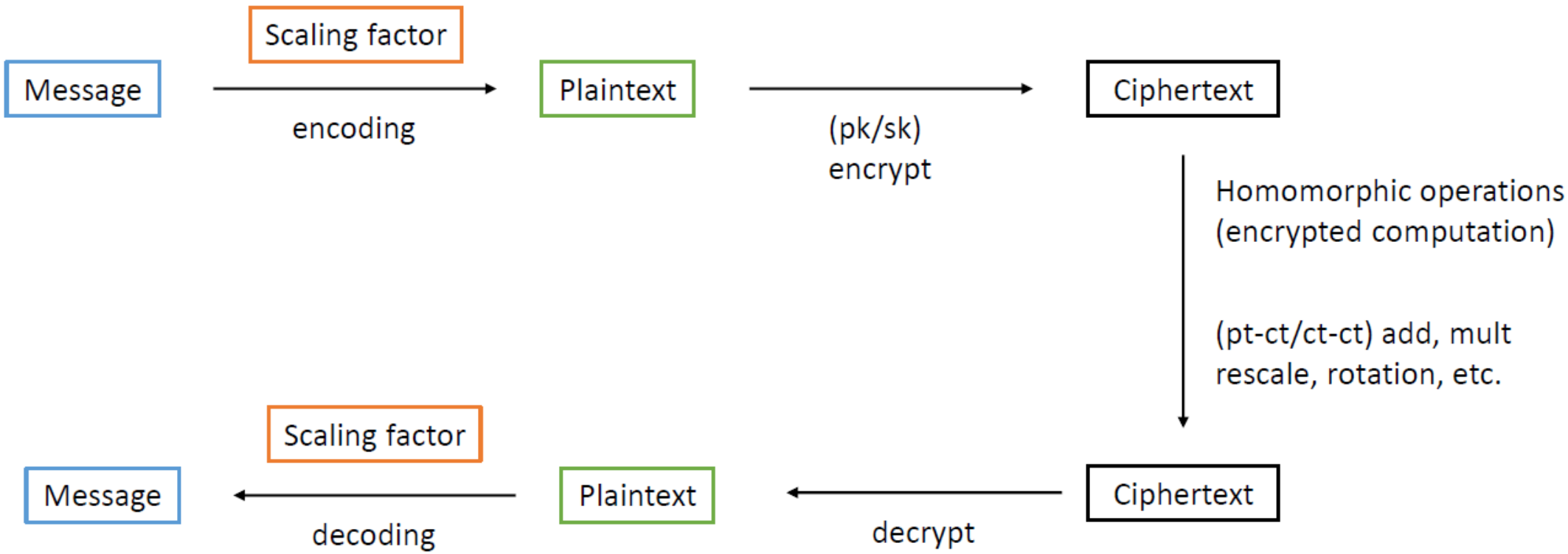
- Barulah pada tahun 2009 konstruksi skema FHE yang mendukung banyak operasi perkalian dan penjumlahan dikembangkan oleh Gentry (2009) dengan menggunakan kriptografi berbasis teori *lattice* (di luar pembahasan kuliah ini)

- Pada tahun 2017, Cheon, Kim, Kim, dan Song memperkenalkan skema enkripsi homomorfik penuh yang dinamakan skema CKKS.

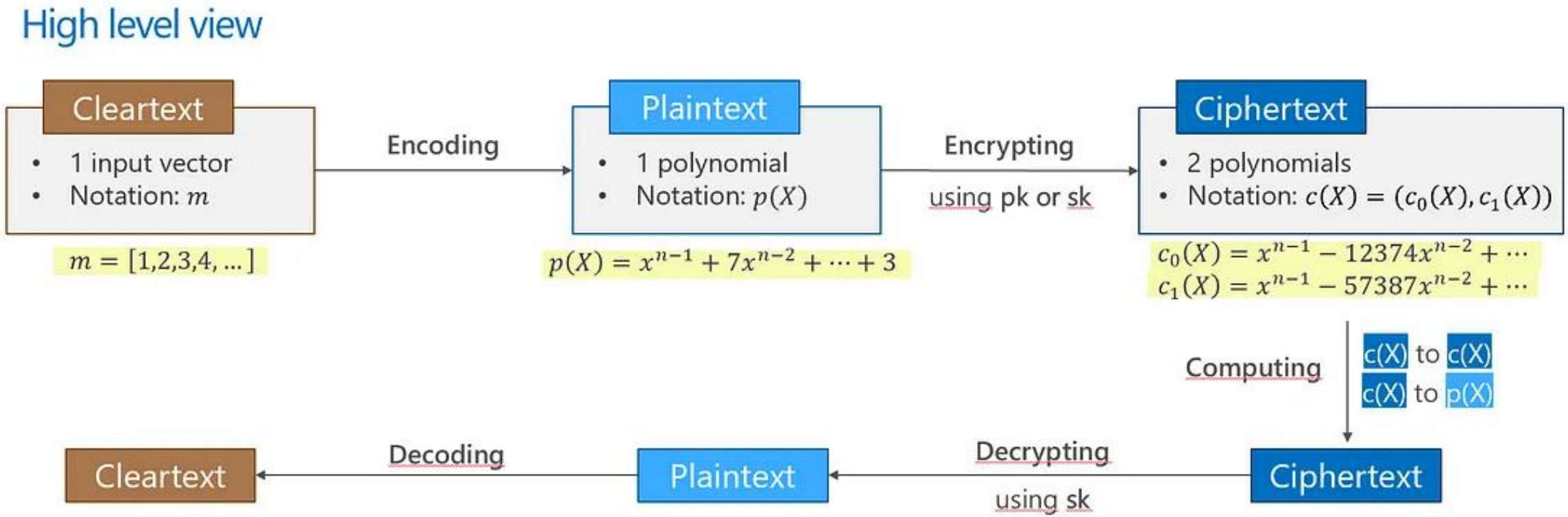
- Skema CKKS mendukung operasi pertambahan serta perkalian pada cipherteks dengan prosedur *rescaling* baru untuk mengatur nilai dari plainteks.

- Prosedur tersebut memotong sebuah cipherteks menjadi modulus yang lebih kecil yang menyebabkan nilai plainteks mendekati nilai aslinya

Algorithms in CKKS



Sumber: Introduction to CKKS (a.k.a. Approximate Homomorphic Encryption), by Yongsoo Song



Sumber: https://towardsdatascience.com/homomorphic-encryption-intro-part-2-he-landscape-and-ckks-8b32ba5b04dd

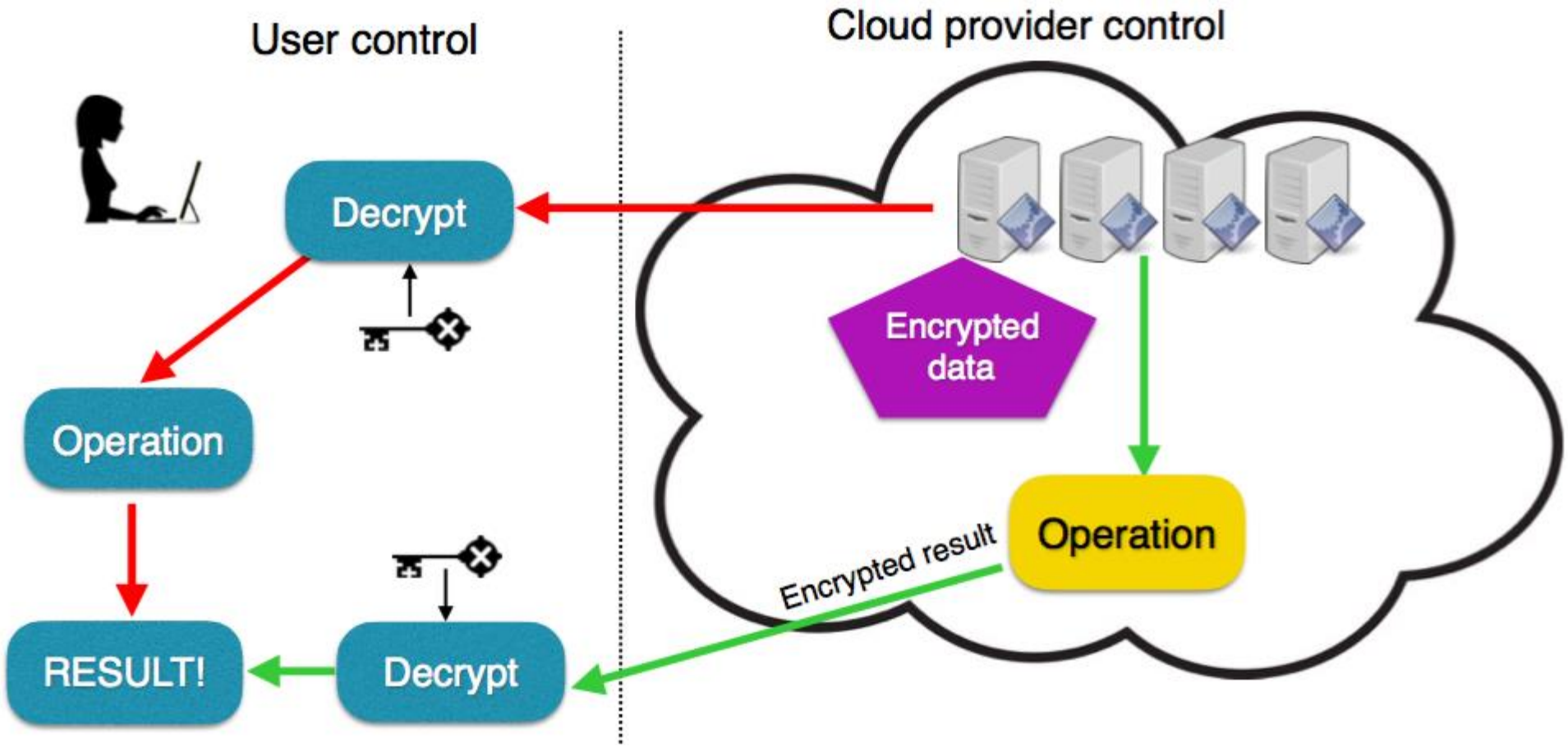
- Baca tulisan tentang CKKS dan contohnya lebih mendetil:

  https://blog.openmined.org/ckks-explained-part-1-simple-encoding-and-decoding/

Homomorphic Encryption di dalam Cloud Cryptography

- *Cloud cryptography* adalah penerapan teknik kriptografi untuk melindungi data, komunikasi, dan layanan yang berada di lingkungan *cloud computing*.
- Tujuannya adalah menjaga:
  1. Kerahasiaan (*confidentiality*) → data tidak bisa dibaca pihak tidak berwenang
  2. Integritas (*integrity*) → data tidak diubah tanpa izin
  3. Autentikasi (*authentication*) → memastikan identitas pengguna/sistem
  4. *Non-repudiation* → pengirim tidak dapat menyangkal tindakan yang dilakukan
- *Cloud cryptography* digunakan karena pada sistem *cloud*, data disimpan di *server* milik penyedia layanan *cloud* dan diakses melalui internet, sehingga risiko penyadapan, pencurian data, dan akses ilegal menjadi lebih besar.

Sumber: ChatGPT


User control
Cloud provider control
Decrypt
Operation
RESULT!
Decrypt
Encrypted data
Operation
Encrypted result

Beberapa contoh layanan dan jenis *cloud computing* yang umum digunakan.

1. Cloud Storage

Digunakan untuk menyimpan file secara online.

Contoh:

- Google Drive
- Dropbox

2. Cloud Computing Platform

Menyediakan server, database, jaringan, dan komputasi melalui internet.

Contoh:

- Amazon Web Services (AWS)
- Google Cloud Platform
- Microsoft Azure
- Oracle Cloud

- Bagaimanaca cara *cloud* dapat melakukan operasi terhadap data tanpa pernah melihat isi asli data tersebut?
- Jawabannya adalah dengan menggunakan *homomorphic encryption*.
- *Homomorphic encryption* sangat penting pada *cloud* karena pengguna sering menyimpan data rahasia di *cloud*, tetapi tetap memanfaatkan kemampuan komputasi *cloud*.

Penggunaan Homomorphic Encryption di Cloud

1. Cloud Data Processing

- Pengguna menyimpan data terenkripsi di cloud.
- Cloud dapat melakukan analisis, pencarian, statistik, machine learning, dll, tanpa mendekripsi data.

2. Cloud Healthcare

- Rumah sakit ingin menggunakan cloud untuk analisis medis tetapi data pasien bersifat rahasia.
- Langkah:
  - Data pasien dienkripsi
  - Dikirim ke cloud
  - Cloud menghitung statistik medis
  - Hasil terenkripsi dikirim kembali
  - Rumah sakit mendekripsi hasil

Cloud tidak pernah melihat:

- nama pasien,
- hasil tes,
- riwayat penyakit.

3. Cloud Machine Learning

- Dataset sensitive seperti data keuangan, biometrik, kesehatan, dll, dengan homomorphic encryption model AI dapat dilatih pada data terenkripsi, prediksi dapat dilakukan tanpa membuka data.
- Ini disebut: privacy-preserving machine learning.

4. Secure Cloud Voting

- Pada sistem e-voting berbasis cloud, suara dienkripsi, cloud menghitung total suara terenkripsi, hasil akhir baru didekripsi oleh otoritas.
- Sehingga isi suara individu tetap rahasia.

- **5. Financial Cloud Computing**
- Bank ingin memakai cloud untuk menghitung risiko, analisis kredit, fraud detection, tetapi data nasabah sangat sensitif.
- Homomorphic encryption memungkinkan komputasi dilakukan di cloud, tanpa membocorkan saldo atau transaksi nasabah.