BAB II
KAJIAN LITERATUR

2.1 Sertifikat

Menurut KBBI, sertifikat adalah tanda atau surat keterangan (pernyataan) tertulis atau tercetak dari orang yang berwenang yang dapat digunakan sebagai bukti pemilikan atau suatu kejadian [20]. Sertifikat juga berperan sebagai bukti formal atas kualifikasi akademis, keahlian profesional, dan pengembangan keterampilan [21]. Dengan kata lain, sertifikat dapat diartikan sebagai sebuah dokumen yang digunakan untuk membuktikan hak dari seseorang terhadap sesuatu hal. Sertifikat digunakan dalam berbagai bidang, seperti pendidikan, kepemilikan, pelatihan, maupun sertifikasai profesi. Sertifikat dapat dibagi menjadi dua jenis, yaitu sertifikat fisik dan sertifikat digital.

Sertifikat digital atau sertifikat elektronik (*e-certificate*), adalah sertifikat yang disimpan dalam bentuk *softcopy*, biasanya berupa format *file* PDF atau pun format citra seperti JPEG dan PNG. Sertifikat digital dikelola secara digital dan dapat diakses dengan cepat dan praktis melalui media tertentu seperti jaringan internet [2]. Sertifikat digital juga memungkinkan diterapkannya teknologi-teknologi keamanan seperti enkripsi, *hashing*, kode QR, dan tanda tangan digital (*digital signature*) [22].

Meskipun demikian, penggunaan sertifikat juga tetap memiliki risiko pemalsuan dan manipulasi data. Pada sertifikat digital, risiko tersebut semakin meningkat dengan beredarnya perangkat lunak yang dapat memodifikasi isi sertifikat secara mudah [3]. Oleh karena itu, sebuah sertifikat hanya berfungsi sebagai media penyimpanan informasi, sedangkan pada proses pengecekan keasliannya masih memerlukan dukungan sistem autentikasi.

2.2 Kriptografi

Kriptografi yang berarti tulisan tersembunyi, berasal dari bahasa Yunani yaitu "*kryptos*" yang berarti tersembunyi atau tidak terlihat, dan "*graphikos*" yang berarti tulisan [23]. Kriptografi adalah bidang dalam keamanan informasi yang bekerja dengan cara mengubah pesan asli menjadi bentuk acak melalui teknik pengkodean tertentu, sehingga hanya pihak berwenang yang bisa membaca dan memprosesnya [24].

Tujuan-tujuan dari kriptografi adalah sebagai berikut [25].

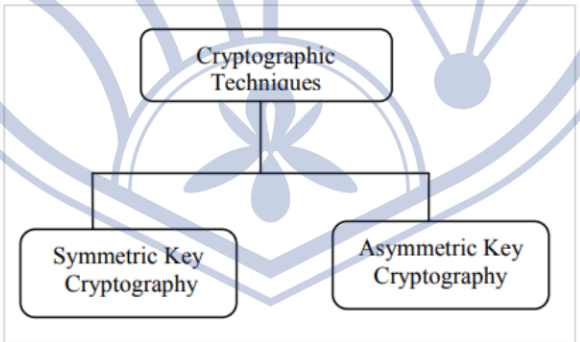
1. Kerahasiaan (*confidentiality*), yaitu memastikan informasi sensitif dan rahasia tetap terlindungi dari segala akses pihak tak berwenang

2. Integritas data (*data integrity*), yaitu menjamin keakuratan dan konsistensi informasi bahwa masih asli dan belum diubah oleh pihak tak berwenang
3. Autentikasi (*authentication*), yaitu mengidentifikasi kebenaran seperti pihak-pihak yang berkomunikasi dan sumber dari informasi
4. Nirpenyangkalan (*non-repudiation*), yaitu memastikan semua pihak yang terkait tidak dapat menyangkal dirinya telah mengirim atau menerima informasi.

Beberapa terminologi dasar dalam kriptografi [26].

1. *Plaintext*, yaitu pesan dalam bentuk aslinya. Contoh, Alice mengirimkan pesan "Halo" kepada Bob, maka "Halo" adalah *plaintext*
2. *Ciphertext*, yaitu data yang telah dienkripsi menjadi bentuk yang tidak dapat dipahami. Contoh, "Halo" yang dienkripsi menjadi *ciphertext* seperti "%L1*ta!K"
3. Enkripsi, yaitu sebuah proses dalam menerjemahkan *plaintext* menjadi *ciphertext* dengan bantuan algoritma enkripsi
4. Dekripsi, yaitu proses yang berlawanan dari enkripsi yakni menerjemahkan *ciphertext* menjadi *plaintext* dengan algoritma dekripsi
5. Kunci (*key*), yaitu angka, alfanumerik, atau simbol unik yang berperan penting dalam proses enkripsi, dekripsi, dan tingkat keamanan informasi.

Sistem kriptografi dapat dikategorikan menjadi dua tipe utama yaitu kriptografi simetris dan kriptografi asimetris. Sementara itu, komponen-komponen lain yang penting dalam kriptografi adalah *hash function* dan *digital signature* yang mengkombinasikan *hash function* dengan kriptografi asimetris dalam menjamin keaslian dan integritas pesan [27].

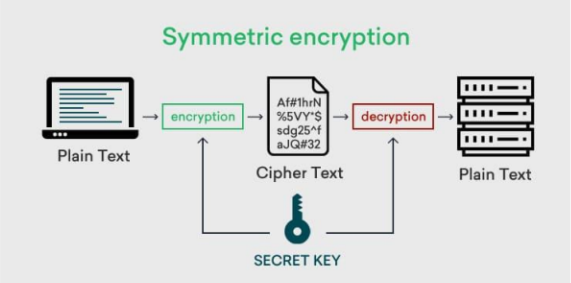


Gambar 2. 1 Kategori Kriptografi [28]

2.2.1 Kriptografi Simetris

Kriptografi simetris menggunakan sebuah *private key* untuk proses enkripsi oleh pengirim pesan dan proses dekripsi oleh penerima pesan. Kriptografi simetris efisien untuk

enkripsi data dalam ukuran besar, namun *private key* yang dibagikan harus tetap dirahasiakan dan dibagikan secara aman [29].



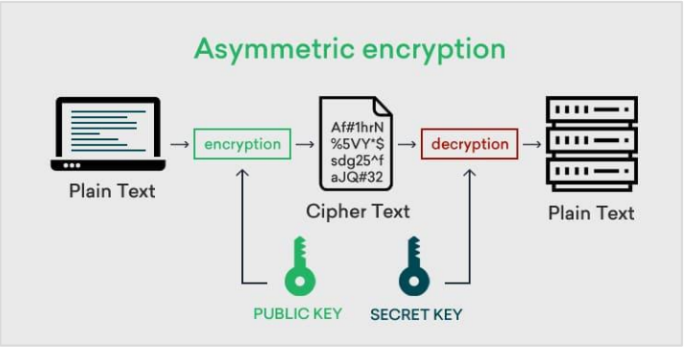
Gambar 2. 2 Skema Kriptografi Simetris [30]

Terdapat dua jenis algoritma kriptografi simetris, yaitu algoritma *stream cipher* dan *block cipher* [28][31]:

1. *Stream cipher* menggunakan *keystream* untuk mengkonversikan setiap *byte* dalam *plaintext* menjadi sebuah *ciphertext*. Sebuah kunci diberikan sebagai *input pseudorandom bit generator* untuk menghasilkan sebuah *keystream*. Kemudian akan dilakukan operasi XOR antara *plaintext* dan *ciphertext*. Untuk masing-masing *byte plaintext*, *key* berbeda digunakan dari *keystream* untuk menghasilkan *ciphertext*. Persamaan matematis *stream cipher* dapat direpresentasikan sebagai $C_i = P_i \oplus K_i$. Contoh algoritma-algoritma *stream cipher* adalah RC4, ChaCha20, dan Salsa20.
2. *Block cipher* mengenkripsi *plaintext* dalam satuan blok-blok yang ukurannya sudah ditentukan menjadi *ciphertext* menggunakan sebuah kunci. Ukuran blok harus merupakan perkalian dari 8, misalnya 64-bit atau 128-bit. *Padding* digunakan jika panjang tidak cocok dengan blok *plaintext*. *Block cipher* membutuhkan Initialization Vector (IV) untuk mengenkripsi blok-blok awal *plaintext*. *Ciphertext* yang diperoleh dari blok sebelumnya digunakan sebagai IV untuk blok-blok selanjutnya. Contoh algoritmanya adalah AES (Advanced Encryption Standard) dan DES (Data Encryption Algorithm).

2.2.2 Kriptografi Asimetris

Kriptografi asimetris yang juga dikenal dengan *public key encryption* menggunakan dua buah kunci yaitu *private key* dan *public key* untuk enkripsi dan dekripsi. *Public key* digunakan untuk mengenkripsi pesan dan *private key* digunakan untuk mendekripsi pesan. *Public key* adalah *key* yang dibagikan ke publik dan *private key* hanya diketahui oleh pengguna berwenang [32].



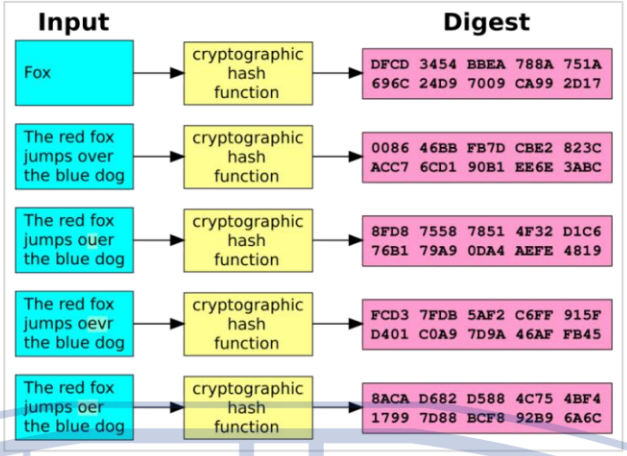
Gambar 2. 3 Skema Kriptografi Asimetris [30]

Kriptografi asimetris menghilangkan masalah distribusi *private key* serta peningkatan kemanan dalam mekanisme autentikasi, namun dengan biaya komputasi yang lebih tinggi dibandingkan kriptografi simetris [29].

Kriptografi asimetris banyak digunakan untuk pembentukan *digital signature*. Beberapa contoh algoritmanya adalah Rivest-Shamir-Adleman (RSA), Digital Signature Algorithm (DSA), metode Diffie-Hellman Key Exchange, dan Elliptic Curve Cryptography (ECC).

2.2.3 *Hashing*

Hashing merupakan proses mengubah data masukan berukuran variatif menjadi keluaran berukuran tetap (*fixed-size*) yang disebut sebagai sebuah *hash*, *hash code*, atau *message digest*. *Hashing* bersifat satu arah sehingga hasilnya tidak dapat dikembalikan ke bentuk awal. Dalam kriptografi, proses *hashing* dilakukan dengan mengaplikasikan fungsi *hash* (*hash function*) yang menerima masukan (pesan) dan mengembalikan *string* dengan ukuran *byte* yang tetap [33]. Beberapa contoh fungsi *hash* yaitu Message Digest 5 (MD5), Secure Hash Algorithm 1 (SHA-1), dan seri Secure Hashing Algorithm 2 (SHA-2) seperti SHA-256 dan SHA-512. Fungsi *hash* sendiri banyak diaplikasikan dalam penyimpanan kata sandi, *digital signature*, *hash table*, deduplikasi data, *blockchain*, dan mata uang kripto.



Gambar 2. 4 Visualisai Proses *Hashing* [34]

Sebuah fungsi *hash* yang ideal harus memiliki beberapa karakteristik berikut [33]:

1. *Determenistic output*, yaitu fungsi *hash* harus selalu menghasilkan keluaran yang sama untuk masukan yang sama
2. *Collision resistance*, yaitu secara teknis harus mustahil (sangat sulit) untuk menemukan dua masukan berbeda yang menghasilkan nilai *hash* yang sama
3. *Pre-image resistance*, yaitu merupakan sifat satu arah, yang berarti mustahil untuk mencari kembali data asli hanya berdasarkan nilai *hash*-nya
4. *Second pre-image resistance*, yang menjamin bahwa tidak mungkin menemukan masukan lain yang memiliki nilai *hash* yang sama dengan masukan yang sudah ditentukan
5. *Avalanche effect*, yaitu perubahan kecil pada input (bahkan satu bit sekalipun) harus menghasilkan perubahan nilai *hash* yang signifikan secara keseluruhan

2.2.4 *Digital Signature*

Digital signature merupakan mekanisme berbasis kriptografi asimetris untuk menjamin keaslian dokumen elektronik maupun pesan dalam pertukaran data digital. Hal ini dilakukan dengan teknik enkripsi untuk membuktikan bahwa dokumentasi tersebut masih asli dan belum dimodifikasi [35].

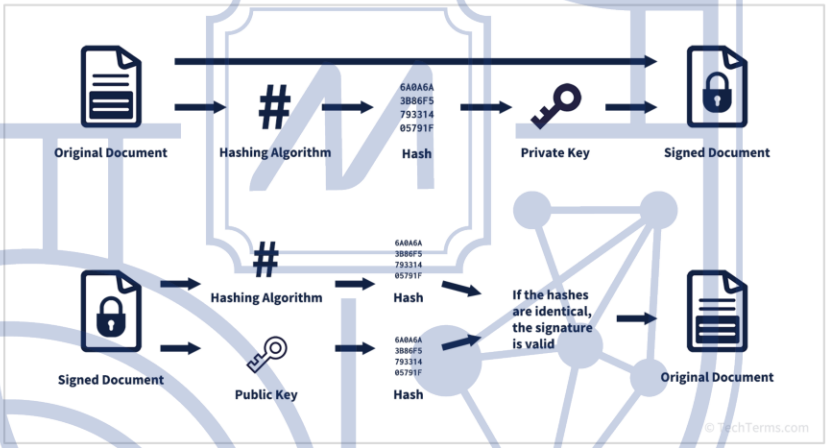
Digital signature atau tanda tangan digital merupakan teknik kriptografi yang berfungsi untuk memverifikasi keaslian dokumen, menjaga integritas data, serta menjamin identitas pengirim melalui penerapan algoritma enkripsi dan dekripsi. Secara lebih spesifik, *digital signature* merupakan mekanisme autentikasi yang dibuat dengan cara mengenkripsi *hash* pesan menggunakan kunci privat (*private key*) [36]. Menurut Jingkun Xu [37], *digital signature* (tanda tangan digital) dihasilkan melalui kombinasi penerapan fungsi *hash* dan

kriptografi asimetris untuk menandatangani pesan dan menyediakan proses autentikasi serta otorisasi pada kondisi tertentu.

Dalam sistem kriptografi asimetris, *private key* digunakan untuk membubuhkan tanda tangan pada pesan, sementara *public key* milik penandatangan digunakan oleh penerima untuk memverifikasi keabsahan tanda tangan tersebut [38].

Berbeda dengan tanda tangan fisik, *digital signature* jauh lebih aman karena terrsusun atas rangkaian biner (deretan angka 0 dan 1) yang bersifat unik untuk setiap pesan dan hampir mustahil untuk diubah atau dipalsukan [38], [39].

Digital signature merupakan elemen krusial dalam kriptografi asimetris dalam menjamin keaslian, integritas, dan nirsangkal (*non-repudiation*) pada pesan digital. Hal ini memungkinkan penerima dapat memverifikasi identitas pengirim, memastikan bahwa pesan tidak dimanipulasi selama proses transmisi, serta mencegah pengirim menyangkal pesannya di kemudian hari [38].



Gambar 2. 5 Skema *Digital Signature* [40]

Algoritma *digital signature* umumnya terdiri atas tiga proses [35], [39]:

1. Proses menghasilkan *key*, yaitu proses menghasilkan pasangan kunci yaitu *private key* K_{pr} dan *public key* K_{pb}
2. Proses menghasilkan *signature*, yaitu dengan mengolah *plaintext* pesan m yang kemudian di-*hash* $H(m)$ dan *private key* pengirim dengan sebuah fungsi enkripsi C.

   $$S_m = C\left(K_{pr}, (m)\right) \quad \ldots\ldots (1)$$

   Kemudian pengirim mengirimkankan pesan dan *signature* kepada penerima.

   $$m_{signed} = (S_m, m) \quad \ldots\ldots (2)$$
3. Proses memverifikasi *signature*, di mana menggunakan algoritma verifikasi *signature* untuk mengecek validitas *signature* yang diterima. Penerima melakukan *hash* terhadap *plaintext* pesan m,

$H(m)$ .....(3)

lalu melakukan dekripsi *signature* menggunakan fungsi dekripsi *D* dengan *public key* K_{pb}.

$D_{Sm} = D(K_{pb}, S_m)$ .....(4)

Jika *signature* benar, maka D_{Sm} dan $H(m)$ akan sama berdasarkan sifat dari kriptografi asimetris. Dengan demikian, pesan memenuhi syarat sebagai autentik dan asli.

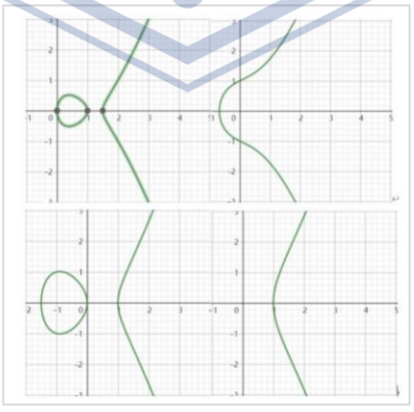
$$D_{Sm} = D(K_{pb}, S_m)$$
$$= D\left(K_{pb}, C\left(K_{pr}, (m)\right)\right)$$
$$= H(m) \quad .....(5)$$

Berbagai skema *digital signature* telah dikembangkan seperti RSA, tanda tangan ElGamal, DSA, dan ECDSA, dan EdDSA sebagai solusi keamanan pada transfer dana elektronik, pertukaran data, hingga distribusi perangkat lunak [37], [38].

2.2.5 Elliptic Curve Cryptography (ECC)

Elliptc Curve Cryptography (ECC) merupakan salah satu skema utama dalam kriptografi asimetris yang menggunakan kurva eliptik untuk proses enkripsi dan dekripsi. ECC ditemukan pada tahun 1985 oleh Neal Koblitz dan Miller [41]. ECC memiliki tingkat keamanan yang lebih tinggi dibandingkan skema kriptografi lainnya dengan ukuran *key* yang lebih kecil. Sebagai contoh, skema ECC dengan ukuran *key* 256-bit memiliki tingkat keamanan yang setara dengan skema RSA dengan ukuran *key* 3072-bit. Hal ini juga menjadikan biaya komputasi ECC lebih rendah [42].

Sebuah kurva eliptik adalah kurva afin yang tidak memiliki patahan dan bergenus 1. Grafik dari kurva eliptik berubah sesuai dengan koefisien yang dimasukkan, seperti bagaimana yang tertera pada gambar Gambar 2. 6 [42].



Gambar 2. 6 Bentuk-Bentuk Kurva Eliptik [42]

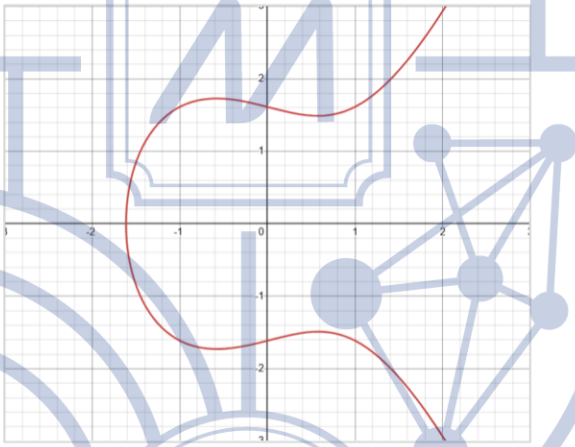
Dalam kurva eliptik, terdapat titik-titik yang membentuk sebuah grup. Operasi-operasi matematis (*point addition*) antar elemen-elemen pada grup tersebut memenuhi 4 sifat:

1. *Closure*, di mana hasil penjumlahan dua elemen masih berada di dalam grup tersebut
2. *Element identity*, yaitu elemen O disebut *point at infinity* di mana $P + O = P$
3. *Inverse element*, di mana untuk setiap titik P terdapat kebalikannya terhadap sumbu x yakni $-P$ di mana $P + (-P) = O$
4. *Associativity/commutative*, di mana $(P + Q) + R = P + (Q + R)$

Dengan demikian, grup tersebut termasuk ke dalam Grup Abelian [43].

Asumsikan sebuah bilangan prima besar p, dan $\mathbb{F}_p$ adalah *finite field over p* atau medan berhingga di atas p, yang berarti setiap operasi aritmatika selalu menghasilkan bilangan bulat dalam rentang 0 hingga p-1 [44]. Setelah diobservasi grafiknya, maka kurva eliptik E terbentuk pada $\mathbb{F}_p$, dan dapat diekspresikan ke bentuk persamaan (6),

$$y^2 = x^3 + ax + b \quad \text{.....(6)}$$



Gambar 2. 7 Bentuk Kurva Eliptik Persamaan $y^2 = x^3 + ax + b$

di mana a dan b merupakan parameter kurva angka-angka dalam $\mathbb{F}_p$, yang memenuhi kondisi pada persamaan (7).

$$4a^3 + 27b^2 \neq 0 \ (mod\ p) \quad \text{.....(7)}$$

Jika (x, y) memenuhi persamaan tersebut, maka pasangan (x, y) adalah titik pada kurva tersebut.

Sementara itu, *order* (n) dalam kurva eliptik adalah jumlah titik dalam batasan medan kurva tersebut [43].

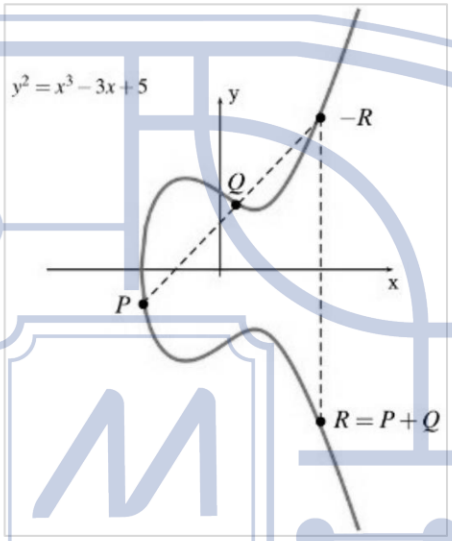
Sebuah *private key* (k) adalah angka acak, sementara itu *public key* adalah sebuah titik pada kurva. Sebuah *private key* jika dikalikan dengan titik generator G akan menghasilkan *public key* [41].

Konsep-konsep matematika umum pada kurva eliptik meliputi [35]:

1. Penjumlahan Titik (*Point Addition*)

   Misal pada kurva eliptik E terdapat dua titik $P_1(x_1, y_1)$ dan $P_2(x_2, y_2)$. Hasil penjumlahan kedua titik tersebut adalah $P_3(x_3, y_3)$ dari $E = (P_3 = P_1 + P_2)$. Sebuah garis menembus titik P dan Q, lalu akan bersilang dengan titik ketiga pada kurva eliptik, kemudian negasi terhadap sumbu x dari titik itu adalah hasil dari penjumlahan P dan Q.

   $$P + Q = R \quad \text{.....(8)}$$



Gambar 2. 8 Proses Penjumlahan Titik [45]

2. Perkalian Skalar (*Scalar Multiplication*)

   Berdasarkan penjumlahan titik, dapat dilakukan perkalian, dilambangkan menjadi $Q = kP$ pada kurva eliptik E di mana $k \in Z^+$ and $(P, Q) \in E^2$. Perkalian skalar pada dasarnya adalah deretan operasi penjumlahan titik.

   $$Q = kP = P + P + P \ldots + P \quad \text{.....(9)}$$

   Titik generator P digunakan untuk menghasilkan titik-titik lain lewat operasi perkalian skalar [43].

Berbeda dengan RSA yang hanya mengandalkan faktorisasi bilangan prima yang sangat besar [41], ECC mengandalkan tingkat kesulitan dalam memecahkan struktur aljabar kurva eliptik pada medan berhingga, serta permasalahan *elliptic curve discrete logarithm problem* (ECDLP) [46].

Asumsikan persamaan $Q = kP$, di mana Q dan P adalah titik pada kurva $E_p(a, b)$ dan $k < p$ di mana p adalah bilangan prima. Permasalahan dari ECDLP sebagai berikut:

a. Jika k dan P diberikan, akan sangat mudah untuk memperoleh nilai Q

b. Namun jika hanya diberikan P dan Q, maka hampir mustahil untuk memperoleh k

Beberapa protokol-protokol kriptografi ECC, antara lain [35]:

a. Elliptic-curve Diffie-Hellman (ECDH)
b. Elliptic Curve Integrated Encryption Scheme (ECIES)
c. Elliptic Curve ElGamal (ECEG)
d. Elliptic Curve Nyberg-Rueppel (ECNR)
e. Elliptic Curve Menezes-Qu-Vanstone (ECMQV)

Protokol-protokol ECC untuk *digital signature* yang paling banyak digunakan pada saat ini adalah ECDSA (Elliptic Curve Digital Signature Algorithm) dan EdDSA (Edwards-curve Digital Signature Algorithm) [47].

Algoritma-algortima ECC umumnya membutuhkan parameter-parameter utama seperti berikut [43]:

1. Bilangan prima p untuk menentukan ukuran medan berhingga
2. Koefisien a dan b dalam persamaan kurva eliptik (yang biasanya dihasilkan lewat *seed* dan fungsi *hash*)
3. Titik generator atau titik basis G untuk menghasilkan titik-titik berikutnya
4. Jumlah titik n
5. Kofaktor h
6. Bilangan acak d di dalam $\{1, \ldots, n-1\}$ sebagai *private key*
7. *Public key* yaitu titik $H = dG$

2.2.6 Edwards-curve Digital Signature Algorithm (EdDSA)

Edwards-curve Digital Signature Algorithm (EdDSA) merupakan algoritma tanda tangan digital yang termasuk dalam keluarga Elliptic Curve Cryptography (ECC). Algoritma ini pertama kali diperkenalkan oleh Daniel J. Bernstein dan rekan-rekannya pada tahun 2012 sebagai alternatif dari skema Elliptic Curve Digital Signature Algorithm (ECDSA) dengan mengimplementasikan kurva Edwards sebagai landasan matematisnya. Algoritma ini umumnya tersedia dalam dua ukuran tanda tangan digital, yaitu 32 *bytes* dan 57 *bytes* [48][49].

Berbeda dengan ECDSA yang bersifat probabilistik dan sangat bergantung pada kualitas Random Number Generator (RNG) untuk menghasilkan nilai acak pada setiap proses penandatanganan [49], EdDSA menerapkan mekanisme deterministik dengan menggunakan fungsi *hash* kriptografis untuk menghasilkan *nonce* dari pesan dan *private key*

[18]. Pendekatan ini bertujuan untuk meminimalisir risiko keamanan akibat kelemahan entropi pada perangkat serta memungkinkan implementasi *constant-time* yang lebih konsisten guna memitigasi serangan *side-channel analysis* (SCA) [50].

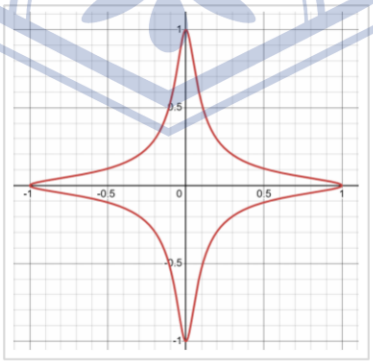
Penggunaan kurva Edwards memberikan karakteristik aritmetika yang berbeda dibandingkan kurva bentuk *Weierstrass* pada ECDSA, karena memiliki rumus penjumlahan titik yang bersifat *unified* atau *complete*. Sifat tersebut memungkinkan operasi titik dilakukan melalui langkah komputasi yang seragam tanpa memerlukan pengecekan kondisi khusus pada titik-titik pengecualian, sehingga memberikan efisiensi operasional yang lebih tinggi tanpa mengorbankan tingkat keamanan sistem [51]. Dalam perkembangannya, EdDSA mendukung instansi kurva seperti Ed25519 dan Ed448. Di antara keduanya, Ed25519 menjadi pilihan yang luas digunakan dalam berbagai protokol keamanan modern karena menawarkan performa yang optimal pada berbagai arsitektur perangkat keras [18].

2.2.7 Kurva Edwards

Kurva Edwards merupakan salah satu bentuk kurva eliptik yang diperkenalkan oleh Harold M. Edwards pada tahun 2007. Kurva ini dikembangkan sebagai alternatif representasi kurva eliptik yang menawarkan efisiensi komputasi lebih tinggi dibandingkan bentuk klasik seperti bentuk *Weierstrass form* yang umum digunakan dalam kriptografi kurva eliptik [52]. Secara matematis, kurva Edwards didefinisikan oleh persamaan (10), dengan ketentuan bahwa parameter d harus memenuhi kondisi $d \neq 0$ dan $d \neq 1$:

$$x^2 + y^2 = 1 + dx^2y^2 \quad \text{.....(10)}$$

Persamaan tersebut menggambarkan relasi antara koordinat x dan y dari titik-titik yang terletak pada kurva. Setiap pasangan nilai x dan y yang memenuhi persamaan tersebut merupakan titik pada kurva Edwards [48].



Gambar 2. 9 Bentuk *Edwards Curves* dengan Persamaan Matematika [48]

Pengembangan lebih lanjut dari kurva Edwards adalah Twisted Edwards Curves, yang didefinisikan melalui persamaan (11).

$$ax^2 + y^2 = 1 + dx^2y^2 \quad \text{.....(11)}$$

Dengan parameter $a, d \in k$, $d \neq 0,1$, dan $a \neq 0$. Apabila nilai $a = 1$, maka persamaan tersebut kembali menjadi bentuk kurva Edwards standar. Titik netral atau elemen identitas dalam struktur grup kurva ini direpresentasikan pada struktur grup kurva ini direpresentasikan oleh koordinat $(0,1)$.

Dalam operasi aritmatika kurva eliptik, titik-titik pada kurva dapat dioperasikan menggunakan aturan grup. Untuk dua titik $P(x_1, y_1)$ dan $Q(x_2, y_2)$ pada kurva Edwards, hasil penjumlahan titik $P + Q = (x_3, y_3)$ dapat dihitung menggunakan persamaan (12) dan (13).

$$x_3 = \frac{x_1y_2 + y_1x_2}{1 + dx_1y_1\ x_2y_2} \quad \text{.....(12)}$$

$$y_3 = \frac{y_1y_2 - ax_1x_2}{1 - dx_1x_2y_1y_2} \quad \text{.....(13)}$$

Rumus penjumlahan ini bersifat *unified*, yang berarti rumus yang sama tetap valid baik untuk menjumlahkan dua titik yang berbeda maupun untuk operasi penggandaan titik (*doubling*) di mana $P = Q$. Sifat ini memberikan keuntungan signifikan dalam implementasi kriptografi karena algoritma dapat menangani kasus-kasus khusus tanpa memerlukan kondisi percabangan (*if-else*) tambahan [53][54].

2.2.8 Ed25519

Ed25519 merupakan salah satu implementasi paling populer dari algoritma tanda tangan digital EdDSA. Algoritma ini dirancang untuk memberikan keseimbangan optimal antara kecepatan operasional yang tinggi dan kemampuan menjalankan implementasi *constant-time*, yang krusial untuk mencegah kebocoran informasi melalui pola waktu eksekusi [18].

Ed25519 merupakan jenis tanda tangan digital berbasis skema Schnorr yang menggunakan kurva Twisted Edwards yang dikembangkan oleh Daniel J. Bernstein dan rekan-rekannya [50]. Secara matematis, Ed25519 ekuivalen secara birasional dengan kurva Montgomery yang disebut Curve25519. Adapun parameter-parameter yang digunakan dalam Ed25519 seperti pada tabel 2.1 [55].

Tabel 2.1 Parameter-Parameter Utama dalam Ed25519

Parameter	Isi Parameter
p	2^{255}-19
b	256
Encoding of	255-bit *little-endian encoding of* $\{0,1,...,p-1\}$

$H(x)$	SHA-512(dom2(phflag, context)‖ x)
c	3
n	254
d	-121665/121666
a	-1
B	($X(P)$,$Y(P)$) of $D25519$ in [RFC7748] L151122213495354007725011514095885315114540126930418572060461132839498477622 02, 463168356949264781694283940034751631413079938662562256157830336031652518559 60))
L	$2^{252} + 27742317777372353535851937790883648493$
$PH(x)$	x

Dalam implementasinya, operasi aritmetika titik pada Ed25519 dilakukan menggunakan sistem koordinat Extended Twisted Edwards $(X, Y, T,$ dan $Z)$ untuk menghindari operasi *inversion field* yang lambat [54]. Pemetaan antara koordinat didefinisikan sebagai $x = X/Z$ dan $y = Y/Z$, sementara T merepresentasikan variabel bantu $xy = T/Z$. Terdapat dua algoritma fundamental dalam siklus operasi titik, yaitu *point doubling* (penggandaan titik) dan *point addition* (penjumlahan titik).

1. *Point Doubling* (Penggandaan Titik)

   Operasi penggandaan titik $P_3 = 2P_1$ dengan $P_1 = (X_1, Y_1, Z_1)$ menghasilkan $P_3 = (X_3, Y_3, Z_3)$ menggunakan representasi koordinat homogen. Algoritma perhitungannya adalah sebagai berikut:

$$A = X_1^2$$
$$B = Y_1^2$$
$$C = 2Z_1^2$$
$$D = -A$$
$$E = (X_1 + Y_1)^2 - A - B$$
$$G = D + B$$
$$F = G - C$$
$$H = D - B$$
$$X_3 = E \cdot F$$
$$Y_3 = G \cdot H$$
$$T_3 = E \cdot H$$
$$Z_3 = F \cdot G \quad \text{.....(14)}$$

2. *Point Addition* (Penjumlahan Titik)

   *O*perasi penjumlahan dua titik $P_1 = (X_1, Y_1, Z_1, T_1)$ dan $P_2 = (X_2, Y_2, Z_2, T_2)$ untuk menghasilkan $P_3 = P_1 + P_2$ dilakukan melalui serangkaian kombinasi operasi medan hingga (*finite field*) seperti berikut [50][55]:

$$A = (Y_1 - X_1) \cdot (Y_2 - X_2)$$
$$B = (Y_1 + X_1) \cdot (Y_2 + X_2)$$
$$C = 2D \cdot T_1 \cdot T_2$$
$$D = 2Z_1 \cdot Z_2$$
$$E = B - A$$
$$G = D - C$$
$$F = D + C$$
$$H = B + A$$
$$X_3 = E \cdot F$$
$$Y_3 = G \cdot H$$
$$T_3 = E \cdot H$$
$$Z_3 = F \cdot G \quad \text{.....(15)}$$

Dalam implementasi kriptografinya, Ed25519 menggunakan fungsi *hash* SHA-512 untuk memproses data yang terlibat dalam proses pembentukan tanda tangan digital. Algoritma ini menghasilkan pasangan kunci dengan panjang masing-masing 256 bit, sedangkan nilai *hash* yang digunakan memiliki panjang 512 bit. Kombinasi penggunaan kurva Edwards dan fungsi *hash* SHA-512 memungkinkan proses pembentukan tanda tangan digital dilakukan secara efisien [47].

Ed25519 terdiri dari tiga fase utama yang serupa dengan skema tanda tangan digital lainnya:

1. *Key Generation,* menghasilkan *public key* dari *private key* melalui perkalian titik skalar pada kurva.
2. *Signing,* menghasilkan pasangan tanda tangan digital dari *private key* dan pesan input menggunakan fungsi *hash* SHA-512.
3. *Verification,* memvalidasi tanda tangan menggunakan *public key* pengirim untuk memastikan integritas dan autentisitas pesan [56].

2.2.9 *Key Generation*

Proses pembangkitan kunci merupakan tahap fundamental dalam skema kriptografi, di mana pasangan *public key* dan *private* key diproduksi. *Private* key dihasilkan secara acak dan dijaga kerahasiaannya oleh penandatangan (*signer*), sementara kunci publik dikonstruksi menggunakan fungsi satu arah (*one-way function*) sehingga dapat digunakan oleh verifikator untuk mengautentikasi tanda tangan yang dihasilkan oleh *private key* tersebut [56].

Tahapan pembangkitan kunci pada EdDSA terdiri dari langkah-langkah sebagai berikut [48][50][56][57]:

1. Pemilihan *Private Key*

   Penanda tangan memilih *private key* SK_A berupa deretan 32-*byte* data acak. Dalam implementasi praktis, penggunaan fungsi seperti *os.urandom()* disarankan untuk menghasilkan *seed* dengan tingkat entropi yang tinggi.

2. Perhitungan *Hash Private Key*

   Nilai *hash* dari *private key* diperoleh dengan memasukkan SK_A ke dalam fungsi SHA-512, sehingga

   $$h = H(SK_A) \quad \text{.....(16)}$$

3. Pembagian Hasil *Hash*

   Hasil *hash* tersebut dibagi menjadi dua bagian:

   a. h_{left} (32 *bytes* pertama), digunakan sebagai basis pembuatan *public key* dan skalar tanda tangan.

   b. h_{right} (32 *bytes* sisanya), disimpan sebagai *prefix* rahasia untuk digunakan sebagai *nonce* pada saat proses penandatanganan berlangsung.

4. Konversi skalar dan *Bit-Clearing Watch Out*

   Bagian h_{left} diinterpretasikan sebagai bilangan bulat dalam format *little-endian*. Pada tahap ini, dilakukan proses *pruning* atau *bit-clearing* dengan memodifikasi bit-bit tertentu untuk membentuk skalar *s*. Proses ini memastikan bahwa skalar *s* memiliki panjang yang tepat, merupakan kelipatan dari *cofactor* kurva yaitu 8, dan berada dalam rentang yang aman. Proses ini dirumuskan sebagai berikut:

   $$s = 2^n + \sum_{i=c}^{n-1} 2^i h_i \quad \text{.....(17)}$$

   dengan keterangan:

   n = Menentukan panjang bit skalar (untuk Ed25519, n = 254).

c = Menentukan indeks awal *bit-clearing* untuk memastikan kelipatan *cofactor*.

h_i = Bit ke-i dari hasil *hash* h_{left}.

5. Pembentukan *Public Key*

   Public key A dihasilkan melalui operasi perkalian titik skalar antara skalar s dengan titik basis B pada kurva Edwards.

   $$A = [s]B \quad \text{.....(18)}$$

   Hasil dari operasi ini adalah sebuah titik pada kurva yang kemudian dikompresi menjadi format standar (biasanya berupa koordinat y dan satu bit tanda untuk x) dengan panjang total 256-*bit*.

6. Pengkodean *Public Key* (*enc(A)*)

   Titik $A = (x, y)$ yang dihasilkan dari operasi perkalian skalar dikonversi ke dalam format standar melalui proses pengkodean (*encoding*) menjadi $\underline{A} = enc(A)$. Proses ini mengkompresi koordinat titik menjadi 32-*byte* dengan memanfaatkan koordinat y dan satu bit tanda (*sign bit*) untuk merepresentasikan nilai x. Penggunaan format *little-endian* dalam pengkodean ini memastikan kompatibilitas antar platform dalam protokol komunikasi data.

2.2.10 *Signing*

Tanda tangan digital (*digital signature*) dihasilkan pada fase ini untuk mengautentikasi pesan atau *file* yang akan ditransfer. Proses pembangkitan tanda tangan melibatkan transformasi pesan ke dalam representasi unik, yang kemudian diintegrasikan dengan *private key* menggunakan fungsi matematika yang didefinisikan oleh parameter kurva. Tanda tangan yang dihasilkan membentuk bukti kriptografis terhadap integritas dan autentisitas pesan, serta memverifikasi identitas pengirim secara aman dalam lingkungan digital.

Dalam implementasi Ed25519, proses penandatanganan pesan M dilakukan dengan menghasilkan pasangan tanda tangan (R, S) melalui mekanisme deterministik. Berbeda dengan ECDSA yang sangat bergantung pada kualitas Random Number Generator (RNG), EdDSA menghasilkan *nonce* secara deterministik dari pesan dan *private key*, sehingga mengurangi risiko kebocoran kunci akibat *nonce* yang buruk atau berulang.

Tahapan penandatanganan pesan M dilakukan melalui langkah-langkah berikut [48][50][56][57]:

1. Pembuatan *nonce* (r), menghasilkan nilai *nonce* dengan melakukan *hash*ing terhadap *prefix* (yang berasal dari *private key*) dan pesan M:

$$h = H\big(dom(F,C)\big||prefix\big|M\big) \quad (19)$$

$$r = \sum_{0}^{2b-1} 2^i h_i (mod\ L) \quad (20)$$

Nilai *r* berfungsi sebagai komitmen acak dalam proses penandatanganan

2. Perhitungan Titik Komitmen (*R*), menghitung titik *R* pada kurva dengan mengalikan *nonce r* dengan titik basis *B*:

$$R = [r]B \quad (21)$$

Titik ini kemudian dikonversi menjadi representasi byte $(\underline{R} = enc(R))$ untuk disertakan dalam tanda tangan.

3. Perhitungan Tantangan (*Challenge k*), menghitung nilai *k* dengan melakukan *hash*ing titik komitmen yang telah dikodekan ($\underline{R}$), kunci public ($\underline{A}$), dan pesan *M*:

$$h = H(dom(F,C) \parallel \underline{R} \parallel \underline{A} \parallel M \quad (22)$$

$$k = \sum_{0}^{2b-1} 2^i h_i (mod\ L) \quad (23)$$

4. Perhitungan skalar tanda tangan (*S*), menghitung nilai *S* dengan mengintegrasikan nilai kunci privat *s*, nilai *challenge* (*k*), dan *nonce* (r) melalui persamaan:

$$S = r + k \times s (mod\ l) \quad (24)$$

5. Tanda tangan akhir dibentuk dengan menggabungkan (*concatenation*) titik *R* (32-*octet*) dan representasi *little-endian* dari *S* (32-*octet*).

$$Signature = \underline{R} \parallel \underline{S} \quad (25)$$

2.2.11 *Verification*

Tahap verifikasi dilakukan oleh penerima pesan untuk memastikan autentisitas tanda tangan serta integritas pesan *M*. Proses ini menggunakan kunci publik pengirim $\underline{A}$ untuk memvalidasi apakah tanda tangan tersebut dihasilkan oleh pemilik kunci privat yang sah [57].

Tahapan verifikasi tanda tangan pada EdDSA adalah sebagai berikut [48], [50], [56], [57]:

1. Rekonstruksi Tantangan (*Challenge*), yaitu verifikator menghitung nilai challenge *k* dengan melakukan *hash*ing pada tanda tangan yang telah dikodekan ($\underline{R}$), kunci publik pengirim($\underline{A}$), dan pesan *M* secara bersamaan:

$$h = H(dom(F,C) \parallel \underline{R} \parallel \underline{A} \parallel M \quad (26)$$

$$k = \sum_{0}^{2b-1} 2^i h_i (mod\ L) \quad (27)$$

2. Validasi matematika, yaitu verifikator melakukan pengecekan ekuivalensi antara dua sisi persamaan untuk membuktikan bahwa tanda tangan memenuhi sifat *scalar multiplication* pada kurva eliptik. Verifikator menghitung dua titik kurva, yaitu P_1 dan P_2:

   Perhitungan titik P_1: Menghitung kombinasi linear dari titik komitmen R dan kunci public $\underline{A}$:

   $$P_1 = R + [k]\underline{A} \quad (28)$$

   Perhitungan P_2: Menghitung titik menggunakan skalar tanda tangan S dan titik basis B:

   $$P_2 = [S]B \quad (29)$$

3. Pengambilan Keputusan, tanda tangan dinyatakan valid apabila kedua nilai tersebut sama, yaitu:

   $$[S]B = R + [k]\underline{A} \quad (30)$$

   Jika kedua sisi tidak sama, maka tanda tangan dinyatakan tidak valid, yang mengindikasikan bahwa pesan telah dimodifikasi atau kunci publik yang digunakan tidak sesuai.

2.2.12 Perbandingan EdDSA dengan Algoritma-Algoritma *Digital Signature* Lain

Elliptic Curve Cryptography (ECC) merupakan algoritma kriptografi asimetris yang dibangun berdasarkan struktur aljabar kurva eliptik. Dibandingkan dengan algoritma non-kurva eliptik, ECC mampu memberikan tingkat keamanan yang setara dengan ukuran kunci yang lebih kecil [58].

EdDSA adalah merupakan salah satu algoritma *digital signature* berbasis ECC yang cukup menonjol karena efisiensi operasionalnya yang tinggi tanpa mengurangi tingkat keamanan yang diperlukan [50]. Sefik Serengil pada penelitiannya memperoleh hasil perbandingan waktu komputasi (detik) EdDSA dengan algoritma-algoritma *digital signature* lain sebagai berikut [57]:

Tabel 2. 2 Performa Algoritma *Digital signature* untuk Tingkat Kemanan 128-bit

Algoritma	Bentuk	Kurva	Ukuran Kunci	*Hash*	*Key Generation*	*Signing*	*Verifying*
RSA	-	-	3072	sha256	9.3434d	0.0704d	0.0693d
DSA	-	-	3072	sha256	41.2033d	0.0065d	0.0126d
ECDSA	Weierstrass	secp256k1	256	sha256	0.0067d	0.0066d	0.0136d

ECDSA	Weierstrass	p256	256	sha256	0.0073d	0.0064d	0.0132d
EdDSA	Edwards	ed25519	253	sha256	0.0122d	0.0116d	0.0227d

Pada tingkat keamanan 128-bit, ECDSA (secp256k1) dan EdDSA (ed25519) menunjukkan kinerja yang lebih baik dibandingkan RSA dan DSA (3072-bit) dalam aspek *key generation*. Sementara itu, untuk *signing* dan *verifying*, RSA adalah yang terlamban di antara semuanya.

Tabel 2. 3 Performa Algoritma *Digital signature* untuk Tingkat Keamanan 192-bit

Algoritma	**Bentuk**	**Kurva**	**Ukuran Kunci**	***Hash***	***Key Generation***	***Signing***	***Verifying***
RSA	-	-	7680	sha384	43.8724d	0.9308d	0.9995d
DSA	-	-	7680	sha384	432.1513d	0.0412d	0.0833d
ECDSA	Weierstrass	p384	384	sha384	0.0174d	0.0171d	0.0352d
EdDSA	Edwards	numsp384tl	382	sha384	0.0356d	0.0201d	0.0616d

Untuk tingkat keamanan 192-bit, ECDSA dan EdDSA jauh mengungguli RSA dan DSA dalam semua aspek. Sementara RSA dan DSA memerlukan waktu puluhan hingga ratusan detik hanya untuk *key generation*, ECDSA dan EdDSA dapat menyelesaikannya dalam waktu kurang dari 40 milidetik.

Tabel 2. 4 Performa Algoritma *Digital signature* untuk Tingkat Keamanan 256-bit

Algoritma	**Bentuk**	**Kurva**	**Ukuran Kunci**	***Hash***	***Key Generation***	***Signing***	***Verifying***
RSA	-	-	15360	sha512	455d	7.0423d	7.6164d
DSA	-	-	15360	sha512	7172.103d	0.1937d	0.3868d
ECDSA	Weierstrass	p521	521	sha512	0.0372d	0.0429d	0.076d
EdDSA	Edwards	e521	519	sha512	0.0667d	0.032d	0.1357d

Ketika tingkat keamanan mencapai 256-bit, penggunaan RSA dan DSA (ukuran kunci 15360-bit) menjadi sangat tidak efisien karena proses pembuatan kuncinya memakan waktu yang sangat lama, yakni masing-masing 455 detik dan lebih dari 7.000 detik. Kecepatan *signing* dan *verifying* RSA dan DSA juga lebih lamban dibandingkan algoritma berbasis kurva eliptik yang menunjukkan skalabilitas yang jauh lebih baik dan tetap efisien pada tingkat keamanan yang lebih tinggi.

Hasil penelitian tersebut menunjukkan bahwa ECDSA dan EdDSA secara konsisten memiliki performa yang lebih unggul dibandingkan RSA dan DSA, terutama dalam aspek kecepatan dan efisiensi. Waktu *key generation* pada RSA dan DSA meningkat secara signifikan seiring bertambahnya ukuran kunci, sehingga kurang optimal untuk lingkungan yang membutuhkan proses *generation* dan *signing* yang cepat. Sebaliknya, ECDSA dan

EdDSA yang berbasis ECC mampu mempertahankan performa yang tinggi pada seluruh tingkat keamanan, dengan EdDSA lebih efisien dalam proses penandatanganan dan ECDSA lebih unggul dalam proses verifikasi.

Secara keseluruhan, EdDSA juga mengungguli ECDSA dalam segi performa, efisiensi, dan kemanan. Dalam penelitiannya, Guruprakash J. [18] menunjukan bahwa EdDSA memiliki ukuran kunci, biaya komputasi, waktu operasi aritmatika, waktu *key generation*, waktu *signing*, dan waktu *verifying* yang lebih sedikit. Selain itu, EdDSA yang berlandaskan kurva Edwards memiliki kelebihan lain seperti:

1. Pendekatan deterministik berbasis Schnorr yang menghilangkan ketergantungan pada bilangan acak (RNG) seperti pada ECDSA
2. *Collision-resistant* terhadap *hash* yang sama
3. Ketahanan *side-channel* yang lebih baik
4. Tidak terdapat *exception case* pada proses penjumlahan

Berdasarkan perbandingan-perbandingan yang ada, EdDSA dapat dianggap sebagai algoritma yang lebih efisien, aman, dan modern dibandingkan algoritma konvensional, serta layak dipertimbangkan sebagai alternatif utama dalam berbagai aplikasi, khususnya pada sistem keamanan sertifikat digital.

2.3 Kode QR

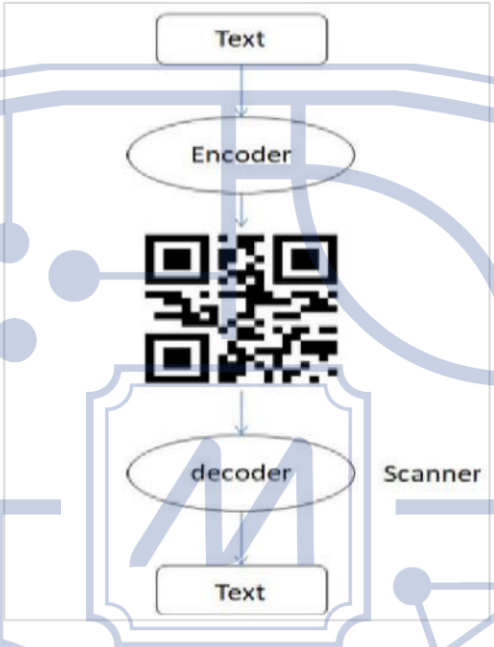
Kode QR merupakan sebuah jenis *barcode* yang menyimpan data dalam bentuk matriks persegi. Data yang disimpan direpresentasikan melalui kombinasi titik hitam (bernilai 1) dan area kosong berwarna putih (bernilai 0) yang membentuk pola tertentu. Titik-titik hitam dan putih tersebut disebut sebagai modul [59].

Gambar 2. 10 Contoh Kode QR

QR adalah singkatan dari "*quick response*" yang mencirikan kode dapat terbaca secara cepat. Data yang tersimpan dapat berupa teks, URL, atau informasi lainnya. Kode QR

diciptakan pada 1994 oleh perusahaan Jepang yang bernama Denso Wave, untuk melacak komponen kendaraan saat proses manufaktur sekaligus mengatasi keterbatasan kapasitas *barcode* tradisional [10].

Kode QR terdiri dari proses enkoder dan dekoder. Enkoder mengubah teks ke dalam bentuk kode QR. Saat dipindai menggunakan gawai, dekoder menginterpretasi pola titik-titik hitam dan putih lalu memunculkan/mengakses informasinya secara instan.

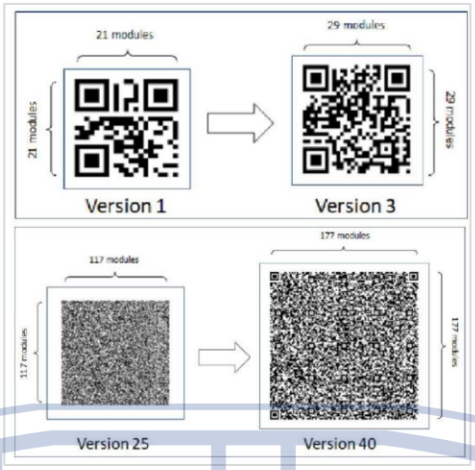


Gambar 2. 11 Alur Proses Kode QR [59]

Kode QR menyimpan data dengan menggunakan 4 standar [59]:

1. Numerik
2. Alfanumerik
3. *Byte* atau biner
4. Kanji atau aksara Jepang

Kode QR memerlukan semakin banyak modul apabila volume data yang perlu disimpan juga semakin banyak. Kode QR memiliki versi 1 hingga 40, masing-masing versi memiliki konfigurasi modul yang berbeda. Versi 1 memiliki konfigurasi modul 21 × 21, sedangkan versi 40 memiliki konfigurasi modul 177 × 177 [59].



Gambar 2. 12 Versi-Versi Kode QR [59]

Kode QR diklasifikasikan menjadi dua jenis, yaitu *static QR code* dan *dynamic QR code*. *Static QR code* bersifat permanen dan memiliki informasi yang tetap (*fixed*), sementara itu *dynamic QR code* memungkinkan pembaruan data tanpa perlu membuat kode QR yang baru.

Kode QR berbentuk dua dimensi (vertikal dan horizontal). Struktur dari kode QR terdiri atas dua komponen utama, yaitu *encoding region* dan *function pattern*.

Function pattern mencakup [59]:

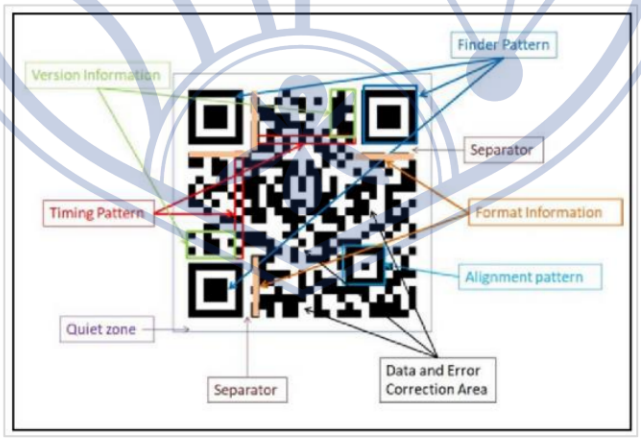
1. *Quiet zone*, area kosong di sekeliling kode QR yang berfungsi untuk membantu *scanner* dalam membedakan kode dari latar belakangnya
2. *Finder pattern*, persegi yang terletak pada tiga sudut utama kode QR. Komponen ini berfungsi sebagai pola pendeteksi posisi khusus untuk membantu *scanner* menyelaraskan, mengenali, dan mendeteksi keberadaan kode QR. Setiap pola merupakan persegi bersarang (*nested square*) dengan konfigurasi:
   a. Persegi Luar: Memiliki warna gelap (hitam) dengan dimensi 7×7 modul
   b. Persegi Tengah: Memiliki warna terang (putih) dengan dimensi 5×5 modul
   c. Persegi Pusat: Memiliki warna gelap (hitam) dengan dimensi 3×3 modul
3. *Separator*, area modul putih yang mengelilingi *finder pattern* untuk membantu identifikasi *encoding region* dan menjamin deteksi yang akurat oleh *scanner*
4. *Timing pattern*, modul terang dan gelap yang disusun selang-seling baik secara vertikal maupun horizontal, yang terletak di antara *separator*, yang berada di lapisan keenam
5. *Alignment pattern*, merupakan pola persegi kecil yang dirancang untuk menentukan posisi dan penyelarasan yang akurat selama proses pemindaian. Pola ini memainkan peran penting, terutama pada kode QR berukuran besar, untuk menjamin orientasi yang

tepat dan keterbacaan data. Setiap unit *alignment pattern* terdiri dari struktur persegi bersarang dengan konfigurasi sebagai berikut:

a. Persegi Luar: Terdiri dari 5 × 5 modul berwarna gelap (hitam) Persegi Dalam: Terdiri dari 3 × 3 modul berwarna terang (putih)
b. Pusat Pola: Terdiri dari sebuah modul tunggal berwarna gelap (hitam) di bagian tengah.

Encoding region mencakup [59]:

1. *Format information*, terletak setelah *separator* di dalam kode yang menyimpan informasi berupa *error correction level* dan *mask pattern* untuk menyederhanakan proses pemindaian kode
2. *Version information*, pola yang secara spesifik menentukan nomor versi suatu kode QR sekaligus menentukan ukuran dan kapasitas penyimpanan data. Meskipun kode QR menawarkan hingga 40 variasi versi yang berbeda, namun dalam implementasinya, penggunaan versi 1 sampai dengan versi 7 merupakan yang paling sering dijumpai.
3. *Data and error correction code*, bagian ini merupakan wilayah utama (*main region*) tempat penyimpanan informasi yang telah disandikan, seperti detail kontak, URL, teks, maupun data spesifik lainnya. Untuk menyimpan data terenkripsi tersebut, digunakan area seluas 6 × 3 modul yang terletak di atas *finder pattern* kiri bawah, serta area 3 × 6 modul di sebelah kiri *finder pattern* kanan atas. Selain menyimpan data informasi, wilayah ini juga memuat informasi terkait kode koreksi kesalahan (*error correction code*)



Gambar 2. 13 Bagian-Bagian Kode QR [59]

Kode QR banyak digunakan pada sistem autentikasi dokumen digital karena mampu menyimpan data dalam kapasitas yang lebih besar dibanding *barcode* satu dimensi serta dapat diakses dengan cepat menggunakan perangkat-perangkat seperti *smartphone*. Pada

sistem sertifikat digital konvensional, kode QR umumnya digunakan untuk menyimpan tautan menuju situs *web* autentikasi eksternal. Namun, pendekatan tersebut masih memiliki kelemahan karena validasi keaslian sertifikat bergantung pada situs web verifikasi eksternal. Oleh karena itu, pada penelitian ini kode QR digunakan untuk menyimpan hasil *digital signature* secara langsung agar proses autentikasi dapat dilakukan secara lebih mandiri dan aman.

2.4 Optical Character Recognition (OCR)

Optical Character Recognition (OCR) merupakan teknologi yang mengkombinasikan teknologi optik dan komputer untuk mengidentifikasi dan mengenali konten teks di dalam gambar sehingga dapat dikenali oleh mesin komputer. Tujuan utamanya adalah memberi kemudahan dalam mengkonversi informasi dokumen-dokumen secara otomatis menjadi teks digital yang dapat diedit dan diolah lebih lanjut [60].

Tahapan kerja OCR meliputi lima langkah utama [60]:

1. *Input*, yaitu membaca dan mengubah artikel atau dokumen yang dipindai menjadi data biner untuk dianalisis oleh program OCR.
2. *Image pre-processing*, yaitu tahap pembersihan gambar untuk menghilangkan cacat atau *noise*. Teknik yang digunakan meliputi koreksi penyelarasan (*offset correction*), pembersihan garis tepi, hingga penghapusan bintik-bintik pada gambar (removing speckle).
3. *Text detection*, yaitu proses mendeteksi keberadaan teks di dalam gambar. Sistem membedakan antara area terang sebagai latar belakang dan area gelap sebagai karakter atau kata.
4. *Text recognition*, yaitu bagian yang bertanggung jawab untuk pencocokan pola (*pattern matching*) dan ekstraksi fitur (*feature extraction*). *Pattern matching* membandingkan citra karakter (*glyphs*) dengan templat yang tersimpan, sedangkan ekstraksi fitur memecah karakter menjadi elemen seperti garis, lingkaran tertutup, dan arah garis untuk menemukan kecocokan terbaik.
5. *Output*, yaitu tahap akhir di mana sistem mengubah data yang diperoleh menjadi data tekstual yang dapat diidentifikasi oleh komputer.

Gambar 2. 14 Alur Proses OCR [60]

Performa sebuah sistem OCR diukur melalui beberapa indikator utama, seperti tingkat penolakan (*rejection rate*), tingkat kesalahan pengenalan (*false recognition rate*), serta kecepatan pengenalan. Terdapat beberapa faktor yang dapat menghambat akurasi identifikasi, antara lain [60]:

1. Kualitas gambar yang buruk akibat paparan cahaya berlebih (*exposure*), pantulan, atau tertutup sebagian (*partial occlusion*).
2. Variasi *font* tulisan yang tidak konsisten dan arah teks yang sembarang.
3. Latar belakang gambar yang kompleks dan berwarna-warni (*noise*).

Optical Character Recognition merupakan bidang luas dengan berbagai macam penerapan. Teknologi ini umum digunakan dalam pemrosesan faktur, sektor hukum, perbankan, serta layanan kesehatan. Selain itu, OCR juga banyak dimanfaatkan pada sistem CAPTCHA, repositori atau perpustakaan digital, pengenalan identitas, serta pengenalan tulisan tangan [60].

Pada penelitian ini, OCR digunakan untuk mengekstraksi informasi tekstual dari sertifikat digital sebelum dilakukan proses autentikasi menggunakan *digital signature*. Penggunaan OCR diperlukan karena sertifikat digital pada sistem ini disimpan dalam format citra sehingga informasi tekstual di dalamnya perlu dikonversi terlebih dahulu menjadi data yang dapat diproses secara komputasional. Teks hasil ekstraksi kemudian digunakan sebagai masukan dalam proses *hashing* dan verifikasi *digital signature* untuk memastikan integritas dan keaslian sertifikat.

2.5 Penerapan *Digital Signature* dalam Kode QR pada Aplikasi Autentikasi Sertifikat Digital

Autentikasi digital merupakan proses dalam memverifikasi keaslian data atau sertifikat digital serta memastikan bahwa data tersebut berasal dari sumber yang sah tanpa mengalami perubahan selama proses pengiriman maupun penyimpanan. Autentikasi digital digunakan pada berbagai dokumen penting seperti sertifikat akademik, ijazah, transkrip nilai, dan dokumen legal lainnya untuk mendukung verifikasi secara digital. Melalui penerapan teknologi kriptografi seperti *digital signature*, fungsi *hash*, dan enkripsi asimetris,

sistem ini dapat mempertahankan kerahasiaan (*confidentiality*), integritas (*integrity*), autentikasi (*authentication*), dan nirpenyangkalan (*non-repudiation*) [1], [5], [14].

Dalam bidang pendidikan, autentikasi digital diterapkan pada sertifikat elektronik (*e-certificate*) agar dapat diverifikasi oleh pihak ketiga secara mandiri tanpa harus melakukan konfirmasi secara langsung kepada penerbit. Proses verifikasi dilakukan melalui sistem digital yang terhubung dengan basis data sertifikat. Dalam perkembangannya, beberapa skema *digital signature* telah diusulkan untuk meningkatkan keamanan, salah satunya adalah penggunaan *digital signature* yang dikombinasikan dengan kode batang seperti kode QR. Pada skema ini, informasi autentikasi atau *digital signature* disisipkan ke dalam kode QR yang tertera pada sertifikat sehingga proses validasi dapat dilakukan secara instan melalui pemindaian menggunakan perangkat *scanner* atau *smartphone* [1], [15].

Dalam sistem autentikasi sertifikat digital, *administrator* memasukkan informasi sertifikat seperti nama pemegang, departemen, nomor ID, kelas gelar atau program atau tahun kelulusan ke dalam aplikasi untuk membentuk *digital signature* melalui enkripsi *message digest* sertifikat menggunakan *private key*. Sistem kemudian menghasilkan kode QR unik untuk setiap sertifikat dan menyimpan datanya di dalam basis data. Proses autentikasi dilakukan dengan membandingkan data hasil pemindaian kode QR terhadap informasi pada basis data, dan apabila *message digest* sertifikat identik dengan hasil dekripsi *digital signature*, maka dokumen dinyatakan valid [9], [14].

Beberapa penelitian terdahulu telah memberikan landasan bagi pengembangan sistem autentikasi ini. Penelitian yang dilakukan oleh Suhardi mengimplementasikan sistem autentikasi pada dokumen akademik mahasiswa dengan menggunakan algoritma Digital Signature Algorithm (DSA) dan fungsi *hash* SHA-1. Dalam penelitian tersebut, kode QR diimplementasikan sebagai media penyimpanan representasi kode unik dari *digital signature*, sehingga setiap dokumen memiliki identitas kriptografi yang berbeda dan sulit untuk diduplikasi secara ilegal. Hasil penelitian ini menunjukkan bahwa sistem yang mengintegrasikan kode QR mampu mempercepat proses validasi dokumen tanpa mengorbankan aspek keamanan data [1].

Selanjutnya, Yogi Ari Winanda memperluas kajian keamanan dokumen digital dengan menerapkan kombinasi algoritma RSA 2048-bit dan SHA-256. Fokus utama penelitian ini adalah menjamin aspek nirpenyangkalan pada dokumen *invoice* dan sertifikat. Penelitian tersebut menunjukkan bahwa meskipun penerapan RSA dilakukan dengan ukuran kunci yang besar untuk meningkatkan keamanan, namun penambahan data *digital signature*

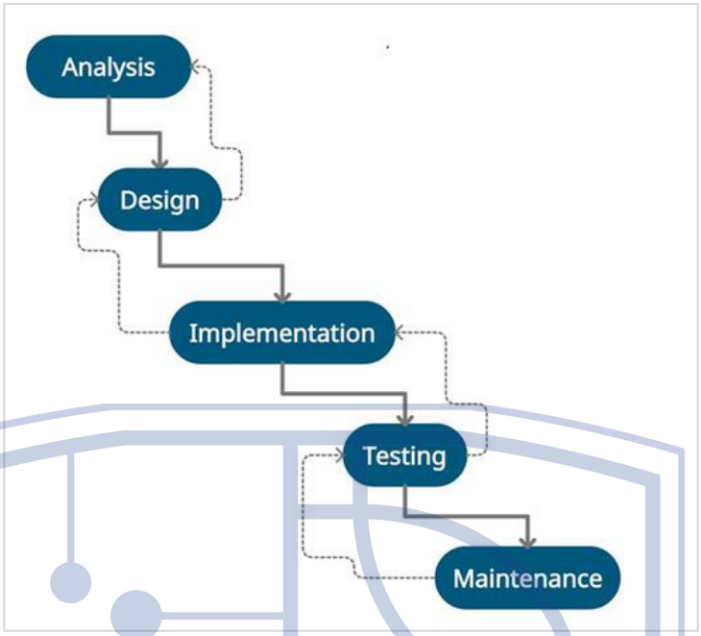
ke dalam kode QR hanya memberikan dampak yang sangat minimal terhadap ukuran *file* akhir. Hal ini dapat menjaga efisiensi penyimpanan dan transimisi data dokumen digital [14].

Penelitian lain oleh Theophilus Wellem yang mengeksplorasi penggunaan Elliptic Curve Digital Signature Algorithm (ECDSA) dengan kurva P-256 dan SHA-256 untuk melindungi dokumen akademik cetak dari pemalsuan. Keunggulan utama dari pendekatan ini adalah efisiensi penggunaan memori, karena ECDSA mampu memberikan tingkat keamanan yang setara dengan RSA 3072-bit namun hanya memerlukan panjang kunci 256-bit. Hal ini menjadi sangat krusial dalam konteks penyimpanan pada kode QR yang memiliki kapasitas karakter terbatas, sehingga memungkinkan penyimpanan data *digital signature* yang kompleks tanpa mengurangi kecepatan pemindaian dan sistem [15].

Secara keseluruhan, penelitian-penelitian tersebut membuktikan bahwa integrasi *digital signature* ke dalam kode QR merupakan metode yang efektif untuk menjamin integritas dan autentikasi sertifikat digital secara otomatis dan efisien. Terlebih lagi, sistem-sistem tersebut memiliki kemampuan untuk mencegah manipulasi dokumen serta pemalsuan tanda tangan melalui mekanisme kriptografi yang jauh lebih aman dibandingkan metode konvensional. Dengan demikian, penerapan sistem sejenis dapat menjadi alternatif dalam meningkatkan keamanan dan efisiensi verifikasi dokumen akademik di era digital.

2.6 Metode Waterfall

Metode Waterfall merupakan model pengembangan perangkat lunak klasik yang bersifat linier dan sekuensial, di mana setiap fase dalam Software Development Life Cycle (SDLC) diselesaikan secara berurutan sebelum proses beralih ke fase berikutnya. Metode ini pertama kali diperkenalkan untuk membawa struktur yang disiplin dalam rekayasa perangkat lunak, memastikan bahwa setiap tahapan memiliki dokumentasi yang lengkap dan tujuan yang jelas. Dalam praktiknya, metode ini sangat bergantung pada definisi kebutuhan di awal proyek, sehingga cocok untuk proyek dengan lingkup yang stabil dan dapat diprediksi [61], [62].



Gambar 2. 15 Model Metode Waterfall [62]

Tahapan-tahapan dalam Metode Waterfall dapat diuraikan sebagai berikut [61], [62], [63]:

1. Analisis Kebutuhan

   Tahap awal yang berfungsi untuk mengumpulkan seluruh kebutuhan sistem secara menyeluruh. Dokumentasi yang dihasilkan pada tahap ini menjadi acuan utama untuk bagi seluruh proses pengembangan selanjutnya.

2. Perancangan Sistem

   Berdasarkan spesifikasi kebutuhan yang telah ditetapkan, pengembang merancang arsitektur sistem, struktur data, dan antarmuka pengguna. Tahap ini berfokus pada desain basis data dan algoritma yang digunakan.

3. Implementasi

   Proses penerjemahan hasil perancangan sistem ke dalam bentuk kode program menggunakan bahasa pemrograman yang telah ditentukan.

4. *Testing*

   Setelah tahap pengkodean selesai, sistem diuji secara menyeluruh untuk memastikan bahwa aplikasi berjalan sesuai dengan kebutuhan yang telah ditetapkan dan bebas dari kesalahan teknis (*bugs).*

5. *Maintenance*

   Tahap akhir di mana setelah sistem digunakan, proses pemeliharaan dilakukan untuk memperbaiki kesalahan yang ditemukan selama penggunaan serta melakukan penyesuaian terhadap perubahan kebutuhan atau lingkungan sistem.