



PANDUAN PENGGUNAAN ALGORITMA KRIPTOGRAFI INDONESIA VERSI 1.0

Deputi Bidang Strategi dan Kebijakan Keamanan Siber dan Sandi
Badan Siber dan Sandi Negara

2025

PANDUAN PENGGUNAAN ALGORITMA KRIPTOGRAFI INDONESIA VERSI 1.0



Direktorat Kebijakan Teknologi Keamanan Siber dan Sandi

Badan Siber dan Sandi Negara (BSSN)

Jl. Cijayanti No. 70, Kec. Babakan Madang, Kabupaten Bogor

Website : <https://www.bssn.go.id>

Email : humas@bssn.go.id

TIM PERUMUS PANDUAN

Pengarah : Tjahjo Khurniawan (BSSN)

Perumus :

1. Soetedjo Joewono (BSSN)
2. Sari Agustini Hafman (BSSN)
3. Mohamad Ikro. (BSSN)
4. Muhammad Arief, MSEE, M.Sc. (BRIN)
5. Okky Robiana Sulaeman, S.Kom, M.Kom (KEMKOMDIGI)
6. Eko Rizanoordibyo (OJK)
7. Dr. Bety Hayat Susanti, S.Si, M.E. (POLTEK SSN)
8. Dr. Sugi Guritman (IPB)
9. Dr. Eng. Khoirul Anwar, S.T., M.Eng. (TELKOM UNIVERSITY)
10. Uha Isnaini, S.Si., M.Sc., Ph.D (UGM)
11. Dr. Pratama Dahlian Persadha, S.Sos, M.M. (CISSReC)
12. Christian Riesaputra (THALES INDONESIA)
13. Angga Airlangga, S.IP, M.PP (IBM INDONESIA)
14. Teguh Wahyono (BSSN)
15. Sutoro, S.ST.TP, M.T. (AFSI)
16. Ratih Mumpuni Arti (BSSN)
17. Novita Angraini (BSSN)
18. Mohamad Rahmadi (BSSN)
19. Oky Jati Permana. (BSSN)
20. Yhufi Swastantri Gustiviana. (BSSN)
21. Siti Hikmatul Khasanah (BSSN)
22. Dian Yuda Paripurna (BSSN)
23. Andreas Lugas (BSSN)

PENGANTAR

Puji syukur kita panjatkan ke hadirat Tuhan Yang Maha Esa karena atas rahmat-Nya, *Panduan Penggunaan Algoritma Kriptografi Indonesia Versi 1.0* ini dapat diterbitkan sebagai salah satu upaya penting untuk memperkuat kedaulatan dan ketahanan siber nasional di tengah percepatan transformasi digital bangsa.

Keamanan informasi merupakan pilar yang harus menjadi prioritas dalam penyelenggaraan sistem elektronik. Meningkatnya intensitas dan kompleksitas ancaman di ruang siber menuntut tersedianya mekanisme perlindungan yang mandiri dan adaptif. Oleh karena itu, penggunaan Algoritma Kriptografi Indonesia (AKI) menjadi bagian yang tidak terpisahkan dari strategi nasional dalam menjaga kerahasiaan, integritas, dan ketersediaan informasi.

Panduan ini menjadi rujukan praktis dalam pemilihan dan penerapan beragam algoritma kriptografi yang menjadi bagian dari AKI, termasuk informasi mengenai batas akhir penggunaan algoritma dengan panjang kunci tertentu sebagai bentuk antisipasi terhadap perkembangan komputasi kuantum. Dengan demikian, dokumen ini diharapkan dapat memudahkan penyelenggara sistem elektronik dalam menerapkan pengamanan kriptografi secara tepat sesuai kategori dan kebutuhan operasional sistem elektronik.

Keberhasilan penguatan keamanan kriptografi nasional memerlukan kolaborasi erat antara pemerintah, industri, akademisi, dan masyarakat. Untuk itu, kami mengajak seluruh pemangku kepentingan untuk menjadikan panduan ini sebagai rujukan dalam penerapan pengamanan informasi yang selaras dengan kepentingan nasional.



Akhir kata, kami menyampaikan apresiasi kepada seluruh Komite AKI dan Sekretariat AKI yang telah terlibat dalam penyusunan panduan ini. Semoga dokumen ini memberikan manfaat nyata dalam mewujudkan ruang siber Indonesia yang tangguh, tepercaya, dan berdaulat, serta mendukung tercapainya visi Indonesia sebagai bangsa digital yang maju dan aman.

Drs. Nugroho Sulistyo Budi, M.M., M.Han.

Kepala Badan Siber dan Sandi Negara

Desember 2025



PENGANTAR PENGARAH KOMITE AKI



Puji syukur kita panjatkan ke hadirat Tuhan Yang Maha Esa karena atas rahmat dan karunia-Nya, *Panduan Penggunaan Algoritma Kriptografi Indonesia (AKI) Versi 1.0* ini dapat disusun dan diterbitkan. Panduan ini merupakan wujud komitmen Badan Siber dan Sandi Negara (BSSN) dalam memperkuat keamanan informasi nasional melalui penggunaan algoritma kriptografi yang aman, terstandar, dan sesuai kebutuhan PSE di Indonesia.

Dokumen ini memberikan panduan praktis bagi para pemangku kepentingan, termasuk PSE, pengembang, dan auditor, dalam menerapkan Algoritma Kriptografi Indonesia (AKI) secara tepat dan bertanggung jawab. Di dalamnya mencakup rekomendasi parameter dan praktik implementasi untuk berbagai jenis algoritma kriptografi seperti *block cipher*, *stream cipher*, fungsi *hash*, *deterministic random bit generator*, serta skema enkripsi dan tanda tangan digital asimetrik. Panduan ini juga melengkapi standar dan pedoman teknis yang telah ada, guna mendukung konsistensi dan efektivitas penerapan Algoritma AKI.

Kami berharap panduan ini dapat menjadi rujukan nasional dalam penerapan kriptografi yang kuat, andal, dan sesuai standar keamanan, sekaligus memperkuat ketahanan keamanan informasi dan kedaulatan digital Indonesia menuju ekosistem keamanan siber yang tangguh dan berkelanjutan.

Tjahjo Khurniawan
Deputi Bidang Strategi dan Kebijakan Keamanan Siber dan Sandi
Desember 2025

DAFTAR ISI

TIM PERUMUS PANDUAN	ii
PENGANTAR.....	iii
PENGANTAR PENGARAH KOMITE AKI	v
DAFTAR ISI	vi
DAFTAR TABEL	viii
DAFTAR GAMBAR	ix
I. PENDAHULUAN.....	1
A. Gambaran Umum	1
B. Ruang Lingkup.....	1
C. Tujuan	2
D. Target Implementasi.....	2
E. Algoritma Kriptografi Indonesia (AKI).....	3
II. ISTILAH, DAN SINGKATAN.....	6
A. Istilah	6
B. Singkatan	7
III. SANDI BLOK (<i>BLOCK CIPHER</i>).....	8
A. Algoritma.....	8
B. Penggunaan.....	9
1. Konfidensialitas	10
a. <i>Electronic Code Book (ECB)</i>	10
b. <i>Cipher Block Chaining (CBC)</i>	13
c. <i>Cipher Feedback (CFB)</i>	15
d. <i>Output Feedback (OFB)</i>	19
e. Counter (CTR).....	21
f. <i>Counter advanced cryptographic prolongation of key material (CTR-ACPKM)</i>	26
2. Enkripsi Terotentikasi (<i>Authenticated Encryption</i>).....	28
a. <i>Galois/Counter Mode (GCM)</i>	29
b. <i>Counter Mode with Cipher Block Chaining Message Authentication Code (CCM)</i>	33
IV. SANDI ALIR (<i>STREAM CIPHER</i>)	36
A. Algoritma.....	36
B. Penggunaan.....	37
V. FUNGSI HASH (<i>HASH FUNCTION</i>).....	41
A. Algoritma.....	41
B. Penggunaan.....	42
VI. PEMBANGKIT BIT ACAK DETERMINISTIK (<i>DETERMINISTIC RANDOM BIT GENERATORS</i>)	46
A. Pembangkit Bit Acak Deterministik berbasis Fungsi Hash (<i>Deterministic Random Bit Generators Based On Hash Function</i>)	48
1. Algoritma	49
2. Penggunaan	49
B. Pembangkit Bit Acak Deterministik Kode Autentikasi Pesan Hash (<i>Hash Message Authentication Code Deterministic Random Bit Generators</i>).....	51
1. Algoritma	51

2. Penggunaan dan Pertimbangan Teknis	52
C. Pembangkit Bit Acak Deterministik Berbasis Sandi Blok (<i>Deterministic Random Bit Generators Based On Block Cipher</i>)	53
1. Algoritma	54
2. Penggunaan	55
VII. ALGORITMA ASIMETRIK	57
A. Primitif Asimetrik	57
1. Problem faktorisasi integer	58
a. Algoritma	58
b. Penggunaan	59
2. Problem logaritma diskrit	64
a. Algoritma	64
b. Penggunaan	64
3. Problem logaritma diskrit kurva eliptis	66
a. Algoritma	66
b. Penggunaan	67
B. Skema Enkripsi Asimetrik	72
1. Algoritma	72
a. RSA-OAEP	72
b. RSA-KEM	73
2. Penggunaan	73
a. RSA-OAEP	75
b. RSA-KEM	76
C. Skema Tanda Tangan Asimetrik	77
1. Algoritma	77
2. Penggunaan	80
DAFTAR PUSTAKA	85



DAFTAR TABEL

Tabel 1.	Daftar Algoritma AKI.	3
Tabel 2.	Parameter teknis Algoritma <i>Stream Cipher</i>	36
Tabel 3.	Fungsi Hash dan Standar Rujukannya.	41
Tabel 4.	Properti Fungsi Hash.	43
Tabel 5.	Parameter dan Tingkat Keamanan SHA-3.	45
Tabel 6.	Daftar Algoritma Hash_DRBG dalam Algoritma AKI.	49
Tabel 7.	Daftar Algoritma HMAC-DRBG dalam Algoritma AKI.	52
Tabel 8.	Daftar Algoritma CTR_DRBG dalam Algoritma AKI.	54
Tabel 9.	Panjang minimum dan maksimum auxiliary primes p_1, p_2, q_1 , dan q_2	60
Tabel 10.	Jenis Serangan dan Mekanisme Weak Key pada RSA	62
Tabel 11.	Parameter pembangkitan pasangan kunci sesuai NIST SP 56a Rev 3 ...	65
Tabel 12.	Algoritma Primitif Asimetrik berbasis ECDLP.	66
Tabel 13.	Panjang minimum kunci simetris dan fungsi hash yang setara atau lebih tinggi dari parameter domain Brainpool.	69
Tabel 14.	Parameter algoritma Curve.	69
Tabel 15.	Parameter Edwards25519.	71
Tabel 16.	Parameter Edwards448.	71
Tabel 17.	Panjang modulus yang direkomendasikan dan tingkat kekuatan keamanannya.	74



DAFTAR GAMBAR

Gambar 1.	Enkripsi dan dekripsi dengan mode operasi ECB.....	12
Gambar 2.	Enkripsi dan dekripsi dengan mode operasi CBC	14
Gambar 3.	Enkripsi dan dekripsi dengan mode operasi CFB.....	17
Gambar 4.	Enkripsi dan dekripsi dengan mode operasi OFB.....	20
Gambar 5.	Enkripsi dan dekripsi dengan mode operasi CTR.....	23
Gambar 6.	Mode CTR-ACPKM	27
Gambar 7.	Fungsi authenticated encryption mode operasi GCM.....	32
Gambar 8.	Fungsi authenticated decryption mode operasi GCM.....	32
Gambar 9.	Skema Umum Enkripsi Algoritma Stream Cipher.	39
Gambar 10.	Mekanisme DRBG.....	48
Gambar 11.	Skema tanda tangan digital	77



I. PENDAHULUAN

A. Gambaran Umum

Dokumen ini disusun sebagai panduan praktis untuk mendukung penggunaan Algoritma Kriptografi Indonesia (AKI) secara aman, sebagaimana direkomendasikan oleh Badan Siber dan Sandi Negara (BSSN) melalui Komite AKI. Panduan ini mencakup pemilihan parameter yang tepat untuk berbagai jenis algoritma kriptografi, seperti sandi blok (*block cipher*), sandi alir (*stream cipher*), pembangkit bit acak deterministik (*deterministic random bit generator*), fungsi *hash*, skema enkripsi asimetrik, dan skema tanda tangan digital asimetrik, serta ulasan singkat mengenai praktik implementasi yang baik dalam penerapannya di sistem elektronik (SE).

Panduan ini tidak dimaksudkan untuk menggantikan standar, pedoman, atau rekomendasi teknis yang telah ditetapkan sebelumnya, melainkan berfungsi sebagai pelengkap yang merujuk pada berbagai referensi terkait. Tujuannya adalah menjadi panduan tambahan yang komprehensif bagi para pemangku kepentingan dalam mengadopsi algoritma AKI secara tepat dan bertanggung jawab.

Perlu dicatat bahwa spesifikasi teknis rinci dari setiap algoritma tidak dicantumkan dalam dokumen ini, melainkan telah tersedia dalam dokumen referensi yang disebutkan.

B. Ruang Lingkup

Ruang lingkup penerapan panduan ini difokuskan pada upaya pengamanan dan perlindungan informasi elektronik dalam penyelenggaraan sistem elektronik (PSE) sebagaimana terdapat dalam Peraturan BSSN Nomor 8 Tahun 2020, khususnya dalam konteks perlindungan terhadap konfidensialitas, integritas, dan autentisitas informasi yang dikelola oleh instansi pemerintah maupun entitas lainnya yang menyelenggarakan layanan elektronik.

C. Tujuan

Dokumen ini bertujuan untuk menjadi referensi panduan bagi pemangku kepentingan dalam menerapkan algoritma AKI secara tepat dan aman. Sasaran utamanya mencakup PSE, developer, dan auditor agar memahami prinsip dasar dan praktik terbaik dalam pemanfaatan algoritma AKI sesuai rekomendasi BSSN. Dengan panduan ini, diharapkan implementasi algoritma AKI berlangsung secara konsisten, efektif, dan selaras dengan ketentuan yang berlaku demi memperkuat keamanan informasi dalam ekosistem SE nasional.

D. Target Implementasi

Dokumen ini ditujukan kepada individu maupun organisasi yang menerapkan algoritma AKI dalam rangka melindungi informasi digital (informasi digital yang dimaksud dapat berada dalam berbagai bentuk termasuk, namun tidak terbatas pada teks, berkas, gambar, audio, dan video). Panduan ini diharapkan dapat menjadi acuan yang jelas dan aplikatif bagi berbagai pihak yang terlibat dalam pengelolaan keamanan informasi.

Target pembaca dari dokumen ini mencakup, namun tidak terbatas pada:

1. PSE, yang bertanggung jawab atas pengelolaan, pemrosesan, dan penyimpanan informasi digital dalam berbagai sektor layanan publik maupun privat;
2. Developer, yang mengimplementasikan algoritma AKI baik secara langsung (dari awal) maupun melalui integrasi dalam pustaka kriptografi yang sudah ada dalam sistem atau aplikasi;
3. Auditor, yang menjalankan fungsi penilaian dan evaluasi terhadap aplikasi, produk, atau layanan yang disediakan oleh pihak ketiga untuk memastikan kepatuhan terhadap standar dan praktik keamanan kriptografi yang berlaku.

Dengan cakupan pembaca tersebut, dokumen ini diharapkan dapat menjadi referensi yang bermanfaat dalam mendukung penerapan algoritma kriptografi secara tepat dan bertanggung jawab.

E. Algoritma Kriptografi Indonesia (AKI)

Dalam algoritma kriptografi yang direkomendasikan oleh BSSN, hampir seluruh primitif kriptografis (*cryptographic primitive*) memanfaatkan kunci, kecuali fungsi *hash* dan *deterministic random bit generator* (DRBG) yang tidak memerlukan kunci untuk operasinya. Algoritma simetrik, seperti *block cipher* dan *stream cipher*, menggunakan satu kunci rahasia yang sama untuk proses enkripsi dan dekripsi. Sebaliknya, algoritma asimetrik, seperti skema enkripsi asimetrik dan skema tanda tangan digital asimetrik, menggunakan pasangan kunci yang terdiri atas kunci privat dan kunci publik. Kunci rahasia maupun kunci privat harus dijaga kerahasiaannya, karena keamanan dari *cryptographic primitive* tersebut bertumpu pada perlindungan kunci, bukan pada kerahasiaan algoritmanya.

Adapun daftar algoritma kriptografi yang direkomendasikan oleh BSSN dapat dilihat pada Tabel 1, sesuai dengan Keputusan Kepala BSSN.

Tabel 1. Daftar Algoritma AKI.

No	Nama Algoritma Kriptografi	Kategori SE		
		Rendah	Tinggi	Strategis
Sandi Blok (<i>Block Cipher</i>)				
1.	AES-128	✓	x	x
2.	AES-192	✓	✓	x
3.	AES-256	✓	✓	✓
4.	Camellia-128	✓	x	x
5.	Camellia-192	✓	✓	x
6.	Camellia-256	✓	✓	✓
Sandi Alir (<i>Stream Cipher</i>)				
1.	HC-128	✓	✓	✓
2.	Salsa 20/20	✓	✓	✓
3.	Sosemanuk	✓	✓	✓
4.	ChaCha20	✓	✓	✓
Fungsi Hash (<i>Hash function</i>)				
1.	SHA-384	✓	✓	✓

No	Nama Algoritma Kriptografi	Kategori SE		
		Rendah	Tinggi	Strategis
2.	WHIRLPOOL	✓	✓	✓
3.	SHA-512/256	✓	✓	✓
4.	SHA3-256	✓	✓	✓
5.	SHA3-384	✓	✓	✓
6.	SHA3-512	✓	✓	✓
7.	SHAKE256	✓	✓	✓
8.	Blake2	✓	✓	✓
Pembangkit Bit Acak Deterministik berbasis Fungsi Hash (Deterministic Random Bit Generators Based On Hash Function)				
1.	SHA1-DRBG	✓	x	x
2.	SHA224-DRBG	✓	✓	x
3.	SHA512/224-DRBG	✓	✓	x
4.	SHA512/256-DRBG	✓	✓	✓
5.	SHA256-DRBG	✓	✓	✓
6.	SHA384-DRBG	✓	✓	✓
7.	SHA512-DRBG	✓	✓	✓
8.	SHA3-224-DRBG	✓	✓	x
9.	SHA3-256 DRBG	✓	✓	✓
10.	SHA3-384 DRBG	✓	✓	✓
11.	SHA3-512 DRBG	✓	✓	✓
Pembangkit Bit Acak Deterministik Kode Autentikasi Pesan Hash (Hash Message Authentication Code Deterministic Random Bit Generators)				
1.	HMAC-SHA1-DRBG	✓	x	x
2.	HMAC-SHA224-DRBG	✓	✓	x
3.	HMAC-SHA512/224-DRBG	✓	✓	x
4.	HMAC-SHA512/256-DRBG	✓	✓	✓
5.	HMAC-SHA256-DRBG	✓	✓	✓
6.	HMAC-SHA384-DRBG	✓	✓	✓
7.	HMAC-SHA512-DRBG	✓	✓	✓

No	Nama Algoritma Kriptografi	Kategori SE		
		Rendah	Tinggi	Strategis
8.	HMAC-SHA3-224-DRBG	✓	✓	x
9.	HMAC-SHA3-256-DRBG	✓	✓	✓
10.	HMAC-SHA3-384-DRBG	✓	✓	✓
11.	HMAC-SHA3-512-DRBG	✓	✓	✓
Pembangkit Bit Acak Deterministik Berbasis Sandi Blok <i>(Deterministic Random Bit Generators Based On Block Cipher)</i>				
1.	AES128-CTR-DRBG	✓	x	x
2.	AES192-CTR-DRBG	✓	✓	x
3.	AES256-CTR-DRBG	✓	✓	✓
Algoritma Asimetrik Problem faktorisasi integer <i>(Integer factorization problem)</i>				
1.	minimum RSA-3072	✓	✓	✓
Algoritma Asimetrik Problem logaritma diskrit <i>(Discrete Logarithm Problem)</i>				
1.	minimum DH-3072	✓	✓	✓
Algoritma Asimetrik Problem logaritma diskrit kurva eliptis/ECDLP <i>(Elliptic Curve Discrete Logarithm Problem based Algorithm)</i>				
1.	P-521	✓	✓	✓
2.	Brainpool 256r1	✓	✓	✓
3.	Brainpool 384r1	✓	✓	✓
4.	Brainpool 512r1	✓	✓	✓
5.	Curve25519	✓	✓	✓
6.	Curve448	✓	✓	✓
7.	Edward25519	✓	✓	✓
8.	Edward448	✓	✓	✓
Skema enkripsi asimetrik <i>(Asymmetric encryption scheme)</i>				
1.	RSA-OAEP	✓	✓	✓
2.	RSA-KEM	✓	✓	✓
Skema tanda tangan digital asimetrik <i>(Asymmetric digital signature scheme)</i>				

No	Nama Algoritma Kriptografi	Kategori SE		
		Rendah	Tinggi	Strategis
1.	RSASSA-PKCS1-v1.5	✓	✓	✓
2.	RSSASSA-PSS	✓	✓	✓
3.	ECDSA	✓	✓	✓
4.	EdDSA	✓	✓	✓

II. ISTILAH, DAN SINGKATAN

A. Istilah

- Nonce* : Nilai yang berubah seiring waktu dan memiliki kemungkinan yang sangat kecil untuk terulang, misalnya nilai acak yang dibangkitkan baru setiap kali digunakan, *timestamp*, *sequence number*, atau kombinasi dari nilai-nilai tersebut [1].
- Personalization string* : Sebuah *string* bit opsional yang dikombinasikan dengan input entropi rahasia dan (mungkin) sebuah *nonce* untuk menghasilkan sebuah *seed* [1].
- Entropy input* : Sebuah *string* bit input yang menyediakan tingkat minimum ketidakpastian yang telah dinilai untuk mekanisme DRBG [1].
- Internal state* : Kumpulan informasi yang disimpan mengenai sebuah instansiasi DRBG. Informasi ini dapat mencakup baik informasi yang bersifat rahasia maupun tidak rahasia [1].
- Cryptographic Primitives* : Algoritma kriptografi tingkat rendah yang digunakan sebagai komponen dasar dalam membangun algoritma kriptografi tingkat lebih tinggi.
- Initialization Vector* : 1) Sebuah vektor yang digunakan untuk menentukan titik awal suatu proses kriptografi [2].

2) Sebuah blok data yang pada beberapa mode operasi diperlukan sebagai input awal tambahan [3].

Siklus (*cycle*) : Jumlah siklus *clock* [CLK] yang dibutuhkan untuk satu kali eksekusi algoritma hingga menghasilkan *output*. Ukuran tersebut bergantung pada strategi implementasi arsitektural dan dapat diperoleh dengan mudah melalui perhitungan *paper and pencil* [4].

Bilangan yang terbukti prima (*provable primes*) : suatu bilangan bulat yang dibentuk sebagai bilangan prima atau dibuktikan sebagai bilangan prima dengan menggunakan algoritma pembuktian keprimaan (*primality-proving algorithm*) [5].

Bilangan yang kemungkinan besar prima (*probable primes*) : Suatu bilangan bulat yang diyakini sebagai bilangan prima berdasarkan uji keprimaan probabilistik, dengan kemungkinan yang sangat kecil hingga dapat diabaikan bahwa bilangan tersebut sebenarnya merupakan bilangan komposit [6].

Destroy (penghancuran) : Tindakan yang dilakukan terhadap kunci atau data rahasia. Setelah dimusnahkan, tidak ada informasi tentang nilai kunci atau data rahasia yang dapat dipulihkan. Proses ini dikenal juga dengan sebutan *zeroization* [7].

B. Singkatan

AE : *Authenticated Encryption*
AES : *Advanced Encryption Standard*
DRBG : *Deterministic Random Bit Generator*
FIPS : *Federal Information Processing Standards*
IETF RFC : *Internet Engineering Task Force Request for Comments*
ISO/IEC : *International Organization for Standardization/International Electrotechnical Commission*
MAC : *Message Authentication Code*

NIST SP	: <i>National Institute of Standards and Technology Special Publications</i>
PKCS	: <i>Public Key Cryptography Standards</i>
SNI	: Standar Nasional Indonesia
IV	: <i>Initialization Vector</i>
LCM	<i>Least Common Multiple</i>

III. SANDI BLOK (*BLOCK CIPHER*)

A. Algoritma

Terdapat dua buah algoritma *block cipher* yang direkomendasikan sebagai Algoritma AKI yaitu AES dan Camellia dengan ukuran kunci dari masing-masing algoritma tersebut adalah 128 bit, 192 bit dan 256 bit. SE rendah menggunakan algoritma dengan panjang kunci minimal 128 bit, SE tinggi menggunakan algoritma dengan panjang kunci minimal 192 bit, dan SE strategis menggunakan algoritma dengan panjang kunci 256 bit.

AES dalam dokumen ini mengacu pada standar berikut:

1. ISO/IEC 18033-3:2010 *Information technology – Security techniques – Encryption algorithms, Part 3: Block ciphers* [8];
2. SNI ISO/IEC 18033-3:2010 Teknologi informasi – Teknik keamanan – Algoritma enkripsi – Bagian 3: Penyandian blok (ISO/IEC 18033-3:2010 dan ISO/IEC 18033-3:2010 Amd.1:2021, IDT, Eng) [9];
3. NIST FIPS PUB 197 - *Advanced Encryption Standard* [10].

Camellia dalam dokumen ini mengacu pada standar berikut:

1. ISO/IEC 18033-3:2010 *Information technology – Security techniques – Encryption algorithms, Part 3: Block ciphers* [8];
2. SNI ISO/IEC 18033-3:2010 Teknologi informasi – Teknik keamanan – Algoritma enkripsi – Bagian 3: Penyandian blok (ISO/IEC 18033-3:2010 dan ISO/IEC 18033-3:2010 Amd.1:2021, IDT, Eng) [9].

Penggunaan algoritma *block cipher* didukung dengan adanya mode operasi. Mode operasi *block cipher* dapat digunakan untuk mendukung konfidensialitas dan *Authenticated Encryption* (AE), dengan mengacu pada standar berikut:

1. NIST 800-38A *Recommendation for Block Cipher Modes of Operation: Methods and Techniques* [11];
2. ISO/IEC 11016:2017 *Information technology - Security techniques - Modes of operation for an n-bit block cipher* [12];
3. SNI ISO/IEC 11016:2017 Teknologi informasi - Teknik keamanan - Mode operasi untuk penyandian blok *n*-bit (ISO/IEC 10116:2017 dan ISO/IEC 10116:2017/Amd1:2021, IDT, Eng) [13];
4. NIST 800-38C *Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality* [14];
5. NIST 800-38D *Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC* [15].

Karena adanya perkembangan teknologi, khususnya komputasi kuantum dengan superposisi, maka algoritma *block cipher* dengan panjang kunci 128 bit dan 192 bit berstatus akseptabel sampai tahun 2030. Setelah tahun 2030, hanya algoritma *block cipher* dengan panjang kunci 256 bit yang masih direkomendasikan penggunaannya.

B. Penggunaan

Bagian ini menguraikan beberapa pedoman penggunaan mode operasi pada *block cipher*. *Block cipher* merupakan algoritma kriptografi yang dirancang untuk memproses data dalam blok-blok berukuran tetap, misalnya 128 bit, 192 bit, atau 256 bit. Dalam praktiknya, data yang akan dienkripsi seringkali memiliki ukuran yang jauh melebihi panjang blok tersebut, misalnya data berukuran 514 bit, sedangkan *block cipher* berukuran 128 bit. Oleh karena itu, mode operasi diperlukan untuk mengatur bagaimana setiap blok diproses dan dihubungkan, sehingga *block cipher* dapat digunakan secara efektif untuk data dengan panjang bervariasi.

Secara umum, terdapat dua jenis skema penggunaan *block cipher*, yaitu: (a) skema enkripsi yang hanya menjamin konfidensialitas dan (b) skema AE yang

memberikan perlindungan tambahan terhadap integritas dan autentisitas data, yang dijelaskan sebagai berikut:

1. **Konfidensialitas**

Konfidensialitas dalam kriptografi dirancang untuk menjaga konfidensialitas data, tanpa menyediakan jaminan atas integritas atau autentisitas data. Oleh karena itu, penggunaannya tidak direkomendasikan dalam skenario komunikasi yang menuntut keamanan tinggi antara dua entitas atau lebih, kecuali jika mode ini dipadukan dengan skema lain, misalnya *Message Authentication Code* (MAC) yang kuat.

Terdapat enam mode operasi *block cipher* untuk konfidensialitas yang direkomendasikan dalam panduan ini dengan mengacu pada tiga standar (NIST SP 800-38A [11], ISO/IEC 11016:2017 [12], dan SNI ISO/IEC 11016:2017 [13]) yaitu *Electronic Code Book* (ECB), *Cipher Block Chaining* (CBC), *Cipher Feedback* (CFB), *Output Feedback* (OFB), *Counter* (CTR), dan *Counter Advanced Cryptographic Prolongation of Key Material* (CTR-ACPKM). Penjelasan dari masing-masing mode operasi adalah sebagai berikut:

a. ***Electronic Code Book* (ECB)**

Mode operasi ECB adalah mode operasi yang paling sederhana dalam algoritma *block cipher*. Karakteristik utama mode operasi ini adalah kemampuan untuk mengenkripsi dan mendekripsi setiap blok data secara independen tanpa bergantung pada blok lain. Meskipun mode operasi ECB memberikan keunggulan dalam hal pemrosesan secara paralel dan akses acak terhadap blok, namun mode operasi ini memiliki sejumlah kelemahan kriptografis.

Kelemahan utama ECB terletak pada sifat deterministik, yaitu blok *plaintext* yang sama akan selalu menghasilkan blok *ciphertext* yang sama ketika dienkripsi dengan kunci yang sama. Sifat tersebut membuat ECB rentan terhadap *dictionary attack*, yaitu penyerang

dapat menyusun himpunan pasangan *plaintext-ciphertext* untuk menebak isi pesan. Dalam komunikasi digital yang sering kali mengandung pengulangan atau pola berulang, ECB menjadi tidak efektif untuk menjamin konfidensialitas. Oleh karena itu, penggunaan ECB tidak disarankan untuk pesan dengan panjang melebihi satu blok, kecuali dalam kondisi khusus seperti ketika tidak terdapat pengulangan pada input atau ketika diperlukan akses langsung dan acak terhadap setiap blok data.

NIST SP 800-57 – Part 1 Rev. 4 [16] menyatakan bahwa mode operasi ECB tidak direkomendasikan untuk penggunaan enkripsi secara umum. Namun untuk aplikasi tertentu, mode operasi ECB masih dapat digunakan, seperti pada NIST SP 800-73-5 part 1 [17], NIST SP 800-73-5 part 2 [18], NIST SP 800-73-5 part 3 [19]. Dalam NIST SP 800-73-5 [17] ECB merupakan salah satu parameter yang pemanggilannya ditentukan oleh *PIV algorithm identifier*. *PIV algorithm identifier* adalah pengidentifikasi berukuran 1 byte yang berfungsi untuk menentukan algoritma kriptografi dan ukuran kunci [17]. Lebih spesifik lagi, ECB digunakan dalam proses pembangkitan IV yang selanjutnya dibutuhkan untuk mengenkripsi *PIV Data* dengan menggunakan mode operasi CBC [18]. ECB juga dimungkinkan untuk digunakan dalam proses autentikasi *PIV Card Application Administrator* [19].

Penggunaan mode operasi ECB terbatas pada aplikasi-aplikasi di atas karena dua hal. Pertama, isi *plaintext* tidak bersifat rahasia (bukan untuk tujuan konfidensialitas). Kedua, panjang *plaintext* tidak lebih dari satu blok (mode operasi pada umumnya dirancang untuk *plaintext* dengan panjang yang bervariasi lebih dari satu blok).

Menurut laporan internal NIST dalam IR 8459 *Report on the Block Cipher Modes of Operation in the NIST SP 800-38 Series* [3], mode operasi ECB dapat tetap memberikan jaminan konfidensialitas dengan catatan *plaintext* yang akan dienkripsi bersifat acak secara

merata (*uniformly random*). Contoh *plaintext* dengan karakteristik tersebut adalah kunci kriptografi yang perlu dienkripsi menggunakan kunci lainnya.

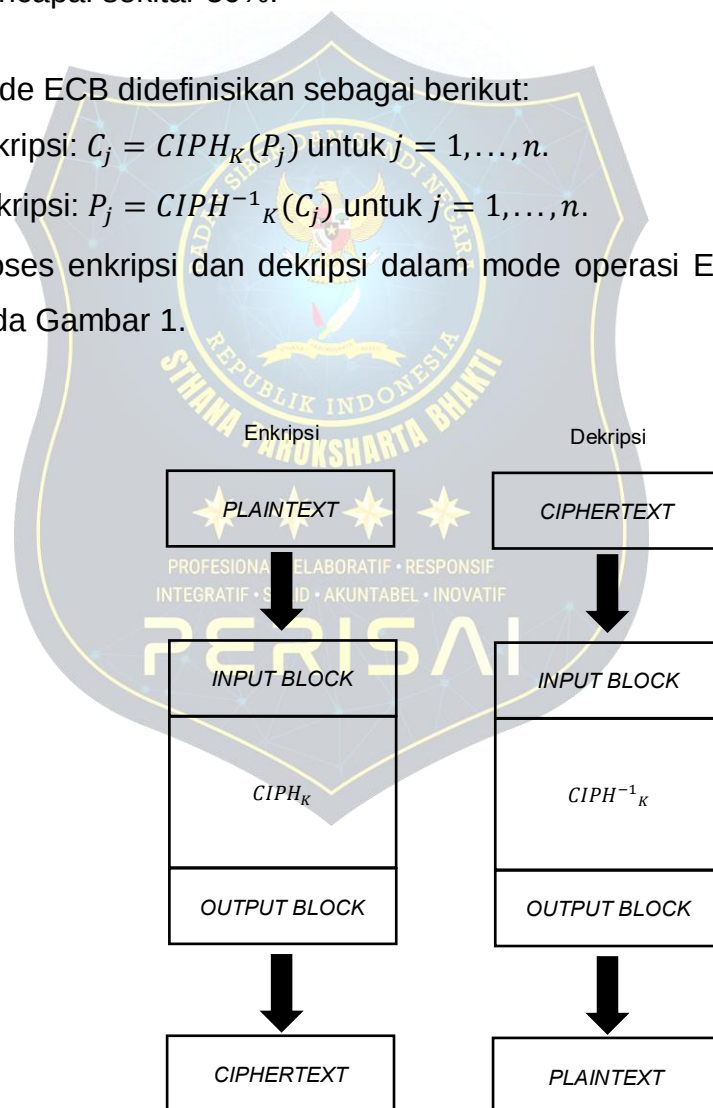
Mode operasi ECB memiliki karakteristik kesalahan lokal (*local error*), yaitu apabila terjadi kesalahan bit pada satu blok *ciphertext*, maka kesalahan tersebut hanya akan memengaruhi hasil dekripsi blok itu saja. Meskipun kesalahan tidak menyebar ke blok lainnya, probabilitas kesalahan per bit dalam blok *plaintext* hasil dekripsi mencapai sekitar 50%.

Mode ECB didefinisikan sebagai berikut:

Enkripsi: $C_j = CIPH_K(P_j)$ untuk $j = 1, \dots, n$.

Dekripsi: $P_j = CIPH^{-1}_K(C_j)$ untuk $j = 1, \dots, n$.

Proses enkripsi dan dekripsi dalam mode operasi ECB ditunjukkan pada Gambar 1.



Gambar 1. Enkripsi dan dekripsi dengan mode operasi ECB digambar ulang dari [11].

Pada proses enkripsi, data *plaintext* terlebih dahulu dibagi menjadi blok-blok dengan panjang tetap yang sesuai dengan ukuran *block cipher* yang digunakan. Setiap blok kemudian diproses melalui fungsi enkripsi yaitu algoritma *block cipher* yang bekerja menggunakan kunci rahasia K yang dinotasikan dengan $CIPH_K$. Hasil proses ini adalah blok *ciphertext* yang bersesuaian. Kemudian, pada proses dekripsi, blok *ciphertext* yang telah diterima akan diproses satu per satu melalui fungsi dekripsi menggunakan kunci rahasia K yang sama yang dinotasikan dengan $CIPH^{-1}_K$. Proses ini mengembalikan setiap blok ke bentuk *plaintext* semula.

b. Cipher Block Chaining (CBC)

Mode operasi CBC merupakan salah satu mode operasi yang digunakan untuk menjamin konfidensialitas data. Pada mode operasi ini, proses enkripsi dilakukan dengan cara menggabungkan setiap blok *plaintext* dengan blok *ciphertext* sebelumnya menggunakan operasi XOR sebelum dienkripsi dengan algoritma *block cipher*. Blok pertama dikombinasikan dengan *Initialization Vector* (IV) yang bersifat acak dan unik untuk setiap sesi enkripsi. Struktur penggabungan antar blok ini menciptakan efek rantai (*chaining block*), yang berfungsi untuk menyamarkan pola berulang dalam *plaintext*, sehingga meningkatkan konfidensialitas data.

Mode CBC didefinisikan sebagai berikut:

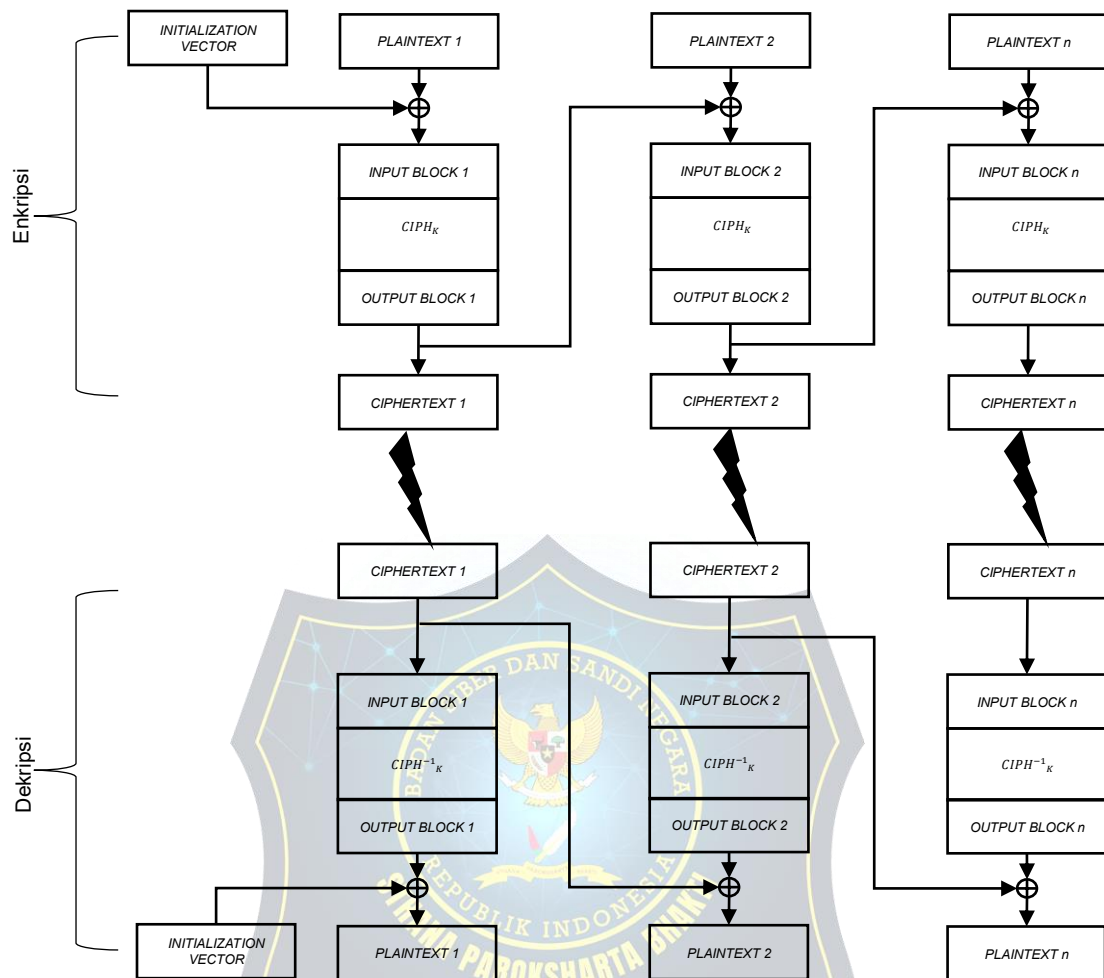
Enkripsi: $C_1 = CIPH_K(P_1 \oplus IV)$;

$$C_j = CIPH_K(P_j \oplus C_{j-1}) \quad \text{untuk } j = 2, \dots, n.$$

Dekripsi: $P_1 = CIPH^{-1}_K(C_1) \oplus IV$;

$$P_j = CIPH^{-1}_K(C_j) \oplus C_{j-1} \quad \text{untuk } j = 2, \dots, n.$$

Gambar 2 menunjukkan alur enkripsi dekripsi dengan mode operasi CBC.



Gambar 2. Enkripsi dan dekripsi dengan mode operasi CBC digambar ulang dari [11].

Pada saat memproses blok pertama, mode operasi CBC menggunakan IV yang dikombinasikan dengan blok *plaintext* pertama menggunakan operasi XOR. Meskipun IV tidak perlu dirahasiakan, nilainya harus tidak dapat diprediksi untuk mencegah kebocoran informasi kriptografis. Hasil operasi ini kemudian dienkripsi menggunakan algoritma *block cipher* dengan kunci rahasia K ($CIPH_K$) untuk menghasilkan blok *ciphertext* pertama.

Untuk blok-blok berikutnya, setiap blok *plaintext* dikombinasikan terlebih dahulu dengan *ciphertext* dari blok sebelumnya menggunakan XOR sebelum dienkripsi. Proses ini berlanjut secara berurutan hingga seluruh blok *plaintext* diproses menjadi *ciphertext*.

Pada proses dekripsi, alur proses berlangsung secara terbalik. Blok *ciphertext* pertama terlebih dahulu didekripsi menggunakan fungsi dekripsi dengan kunci yang sama ($CIPH^{-1}_K$) kemudian hasilnya di-*XOR* dengan IV untuk mendapatkan blok *plaintext* pertama. Blok *ciphertext* kedua didekripsi, lalu hasilnya di-*XOR* dengan *ciphertext* pertama untuk memperoleh blok *plaintext* kedua. Proses ini dilanjutkan hingga semua blok berhasil didekripsi.

Karakteristik utama mode operasi CBC mencakup ketahanan terhadap manipulasi urutan blok *ciphertext*. Karena setiap blok *ciphertext* bergantung pada blok *plaintext* sebelumnya, perubahan urutan *ciphertext* tidak akan menghasilkan urutan *plaintext* yang valid. Selain itu, penggunaan IV yang berbeda untuk setiap enkripsi akan memastikan bahwa *plaintext* yang identik tidak menghasilkan *ciphertext* yang sama, bahkan jika kunci yang digunakan tetap.

CBC memungkinkan enkripsi paralel untuk beberapa blok jika ukuran unit pemrosesan (m) lebih besar dari satu. Sementara itu, proses dekripsi memiliki sifat *random-access*, artinya setiap blok *ciphertext* dapat didekripsi secara langsung tanpa harus mendekripsi seluruh urutan blok-blok sebelumnya. Secara teoretis, CBC dinyatakan aman secara komputasi jika *block cipher* yang digunakan dapat dimodelkan sebagai permutasi semu-acak dan IV dipilih secara acak.

Penggunaan IV secara aman menjadi elemen krusial dalam menjaga keamanan mode CBC. Direkomendasikan dua metode untuk menghasilkan IV yang aman, yaitu memilih IV secara acak dari himpunan string n bit menggunakan sumber keacakan yang tidak dapat diprediksi [4] atau menetapkan blok *plaintext* pertama sebagai *identifier* unik dan mengatur IV sebagai string n bit bernilai nol.

c. **Cipher Feedback (CFB)**

Mode CFB adalah salah satu mode enkripsi untuk menjamin konfidensialitas data, yang bekerja dengan cara memasukkan

kembali segmen-segmen *ciphertext* secara berurutan (*feedback*) ke dalam blok input untuk menghasilkan blok *output*. Blok *output* tersebut kemudian dikombinasikan dengan *plaintext* melalui operasi XOR untuk menghasilkan *ciphertext*. Proses dekripsi dilakukan dengan cara yang serupa, yaitu *ciphertext* dikombinasikan dengan *output cipher* untuk memulihkan *plaintext*.

Mode ini memerlukan sebuah IV sebagai blok input awal. Meskipun IV tidak wajib dirahasiakan, nilainya harus acak atau tidak dapat diprediksi. Prosedur pembangkitan IV yang memenuhi persyaratan ini dijelaskan lebih lanjut dalam *Appendix C NIST SP 800-38A* [11]. Selain IV, CFB juga memerlukan sebuah parameter integer, dinotasikan s , dengan ketentuan $1 \leq s \leq b$, dengan b adalah panjang blok *cipher* (dalam bit). Dalam spesifikasi mode CFB, setiap segmen *plaintext* P dan *ciphertext* C memiliki panjang s bit. Nilai s ini sering kali diikutsertakan dalam penamaan mode CFB, misalnya seperti mode CFB 1 bit, mode CFB 8 bit, mode CFB 64 bit, atau mode CFB 128 bit, yang mencerminkan panjang segmen data yang diproses pada setiap iterasi.

Mode CFB didefinisikan sebagai berikut:

Enkripsi : $I_1 = IV$;

$$I_j = LSB_{b-s}(I_{j-1}) \parallel C_{j-1}^{\#} \quad \text{untuk } j = 2, \dots, n;$$

$$O_j = CIPH_k(I_j) \quad \text{untuk } j = 1, 2, \dots, n;$$

$$C_j^{\#} = P_j^{\#} \oplus MSB_s(O_j) \quad \text{untuk } j = 2, \dots, n.$$

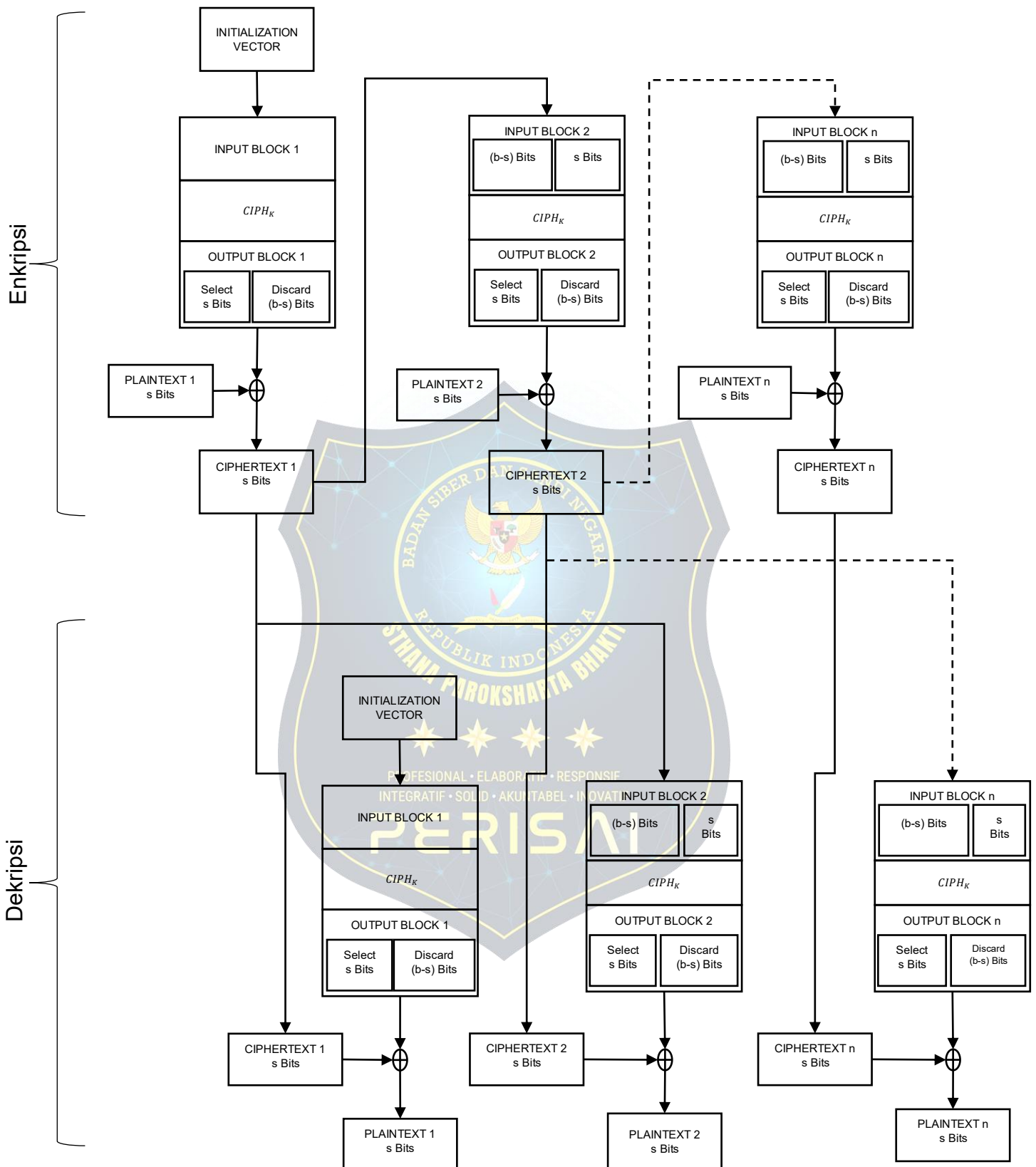
Dekripsi : $I_1 = IV$;

$$I_j = LSB_{b-s}(I_{j-1}) \parallel C_{j-1}^{\#} \quad \text{untuk } j = 2, \dots, n;$$

$$O_j = CIPH_k(I_j) \quad \text{untuk } j = 1, 2, \dots, n;$$

$$P_j^{\#} = C_j^{\#} \oplus MSB_s(O_j) \quad \text{untuk } j = 2, \dots, n.$$

Proses enkripsi dan dekripsi *block cipher* dengan mode CFB ditunjukkan pada Gambar 3.



Gambar 3. Enkripsi dan dekripsi dengan mode operasi CFB
digambar ulang dari [11].

Pada mode CFB, proses enkripsi dimulai dengan penggunaan IV sebagai blok input pertama. Segmen *ciphertext* pertama diperoleh dengan melakukan operasi XOR antara segmen *plaintext* pertama dan bit paling signifikan (*most significant bit*) dari blok *output* tersebut. Sisa bit dari blok *output* dibuang. Untuk membentuk blok input kedua, bit paling tidak signifikan (*least-significant bit*) dari IV digabungkan secara berurutan dengan bit dari segmen *ciphertext* pertama.

Prosedur ini dilanjutkan untuk setiap blok input berikutnya hingga setiap segmen *plaintext* berhasil dienkripsi menjadi segmen *ciphertext*. Secara umum, tiap-tiap blok input selanjutnya akan diproses melalui fungsi enkripsi untuk menghasilkan blok *output*. s bit paling signifikan (*most-significant bit*) dari setiap blok *output* kemudian di-XOR dengan segmen *plaintext* yang bersesuaian guna membentuk segmen *ciphertext*. Segmen *ciphertext* yang dihasilkan, kecuali segmen terakhir, digunakan kembali sebagai bagian dari blok input berikutnya, sesuai dengan mekanisme umpan balik yang telah dijelaskan.

Prosedur dekripsi dimulai dengan menjadikan IV sebagai blok input pertama. Pembentukan setiap blok input berikutnya mengikuti prosedur yang sama seperti pada enkripsi CFB, yaitu dengan menggabungkan $b - s$ *least-significant bit* dari blok input sebelumnya dengan s *most-significant bit* dari segmen *ciphertext* sebelumnya.

Setiap blok input kemudian diproses menggunakan fungsi *cipher* maju (*forward cipher function*) untuk menghasilkan blok *output*. Selanjutnya, s *most-significant bit* dari blok *output* tersebut di-XOR dengan segmen *ciphertext* yang bersesuaian untuk memperoleh kembali segmen *plaintext*.

Sama halnya dengan enkripsi dalam mode CBC, proses enkripsi dalam CFB juga bersifat serial, karena setiap blok input (kecuali yang pertama) bergantung pada hasil dari fungsi enkripsi pada langkah

sebelumnya. Dengan demikian, operasi enkripsi tidak dapat dieksekusi secara paralel. Namun, pada dekripsi CFB memungkinkan untuk dilakukan secara paralel, dengan syarat seluruh blok input telah berhasil dibentuk terlebih dahulu berdasarkan IV dan *ciphertext* yang tersedia.

d. **Output Feedback (OFB)**

Mode OFB merupakan salah satu mode operasi untuk menjaga konfidensialitas data. Mode ini menggunakan iterasi dari *forward cipher function* terhadap sebuah IV untuk menghasilkan serangkaian blok *output* yang kemudian dioperasikan dengan XOR terhadap *plaintext* guna membentuk *ciphertext*, dan sebaliknya untuk proses dekripsi.

Mode OFB mengharuskan IV berupa *nonce*, yaitu IV harus unik untuk setiap eksekusi mode dengan kunci yang sama. Prosedur untuk menghasilkan IV yang memenuhi sifat tersebut dijelaskan dalam *Appendix C NIST SP 800-38A [11]*.

Mode OFB didefinisikan sebagai berikut:

Enkripsi : $I_1 = IV;$

$$I_j = O_{j-1} \quad \text{untuk } j = 2, \dots, n;$$

$$O_j = CIPH_K(I_j) \quad \text{untuk } j = 1, 2, \dots, n;$$

$$C_j = P_j \oplus O_j \quad \text{untuk } j = 1, 2, \dots, n - 1;$$

$$C_n^* = P_n^* \oplus MSB_u(O_n).$$

Dekripsi : $I_1 = IV;$

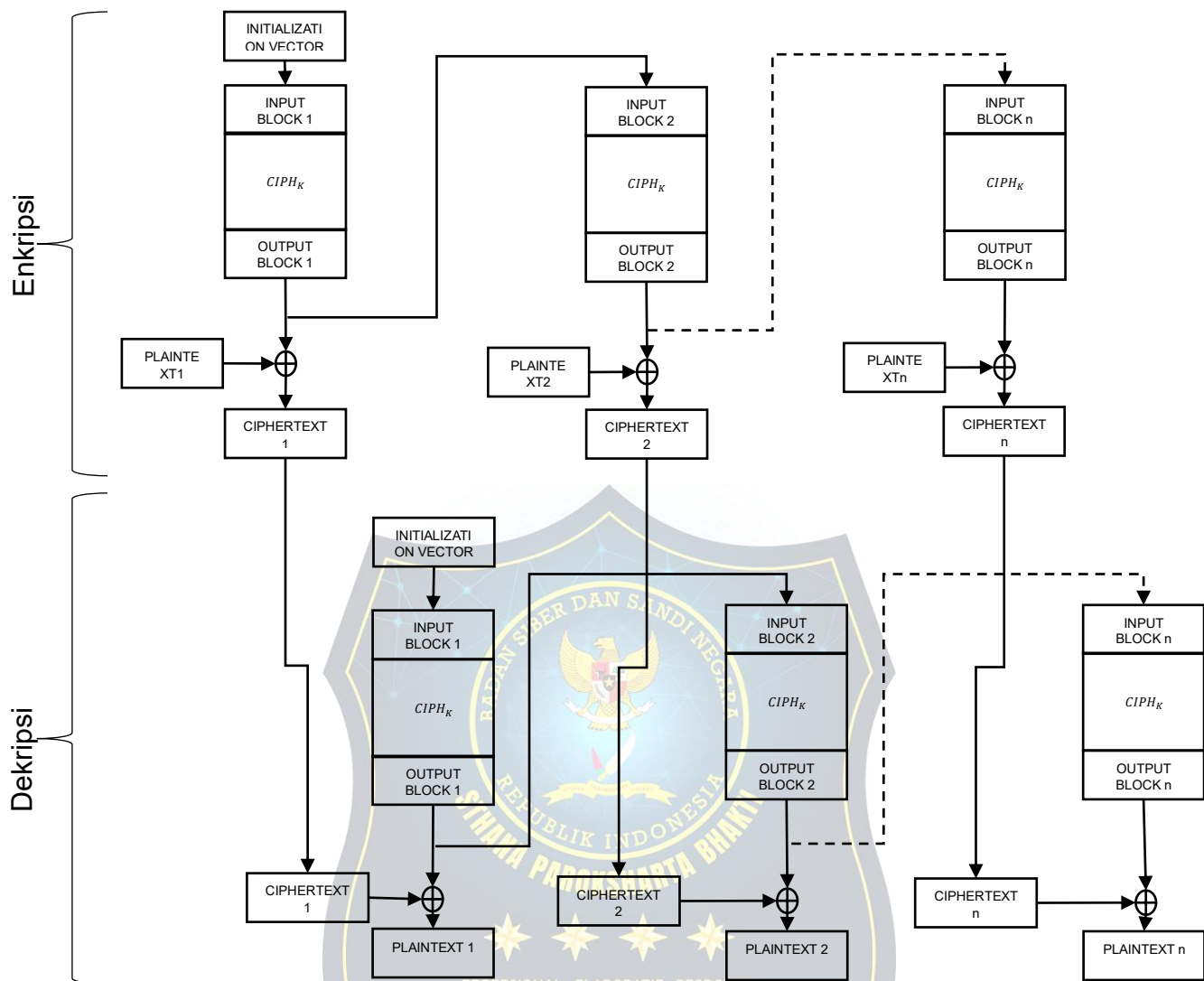
$$I_j = O_{j-1} \quad \text{untuk } j = 2, \dots, n;$$

$$O_j = CIPH_K(I_j) \quad \text{untuk } j = 1, 2, \dots, n;$$

$$P_j = C_j \oplus O_j \quad \text{untuk } j = 1, 2, \dots, n - 1;$$

$$P_n^* = C_n^* \oplus MSB_u(O_n).$$

Proses enkripsi dan dekripsi *block cipher* dengan mode OFB ditunjukkan pada Gambar 4.



Gambar 4. Enkripsi dan dekripsi dengan mode operasi OFB digambar ulang dari [11].

Proses enkripsi dengan mode OFB, IV diproses melalui *forward cipher function* untuk menghasilkan blok *output* pertama. Blok *output* ini kemudian dioperasikan dengan XOR terhadap blok *plaintext* pertama guna menghasilkan blok *ciphertext* pertama.

Selanjutnya, *forward cipher function* diterapkan kembali pada blok *output* pertama untuk menghasilkan blok *output* kedua. Blok *output* kedua ini di-XOR dengan blok *plaintext* kedua, menghasilkan *ciphertext* kedua. Proses ini berlanjut secara iteratif, yaitu setiap blok *output* berikutnya diperoleh dengan menerapkan fungsi enkripsi pada

blok *output* sebelumnya, dan hasilnya kemudian di-XOR dengan *plaintext* yang bersesuaian untuk membentuk *ciphertext*.

Dengan demikian, urutan blok *output* terbentuk dari hasil iteratif fungsi enkripsi terhadap blok-blok sebelumnya, tanpa melibatkan *plaintext* atau *ciphertext* dalam jalur umpan balik. Operasi XOR terhadap blok *plaintext* menjamin konfidensialitas, selama IV bersifat unik dan fungsi enkripsi tetap konsisten dengan kunci yang sama.

Untuk blok terakhir yang merupakan blok parsial berukuran u bit, hanya u *most-significant bit* dari blok *output* terakhir yang digunakan dalam operasi XOR. Sisa $b - u$ bit dari blok tersebut tidak digunakan.

Dalam proses dekripsi, urutan blok *output* (O_j) dihasilkan melalui fungsi $CIPH_K$ yang diterapkan pada blok *output* sebelumnya, identik dengan prosedur enkripsi. Blok-blok tersebut kemudian digunakan dalam operasi XOR dengan segmen *ciphertext* (C_j) untuk memulihkan *plaintext* yang bersesuaian.

Mode OFB menawarkan keuntungan dalam hal stabilitas struktur fungsi $CIPH_K$ terhadap kesalahan pada *ciphertext*; karena tidak ada umpan balik dari *ciphertext* ke dalam fungsi $CIPH_K$, kesalahan bit pada *ciphertext* hanya berdampak lokal pada *plaintext*. Selain itu, sifat simetris dari proses enkripsi dan dekripsi membuat implementasi OFB efisien untuk perangkat lunak dan perangkat keras dengan sumber daya terbatas.

e. Counter (CTR)

Mode operasi CTR merupakan salah satu mode operasi *block cipher* yang dirancang untuk menjamin konfidensialitas data. Mode operasi ini memanfaatkan *forward cipher function* yang diterapkan terhadap serangkaian blok input yang disebut *counter* untuk menghasilkan blok *output* (*keystream*). Setiap blok *output* tersebut kemudian di-XOR dengan blok *plaintext* yang bersesuaian untuk menghasilkan blok

ciphertext. Proses dekripsi dilakukan dengan cara yang identik, yaitu dengan meng- *XOR* blok *output* yang sama terhadap blok *ciphertext* untuk menghasilkan *plaintext*.

Mode CTR memerlukan *counter* yang digunakan dalam proses enkripsi maupun dekripsi. Agar keamanan terjamin, setiap blok *counter* dalam urutan tersebut harus unik, baik dalam satu pesan maupun di seluruh pesan yang dienkripsi menggunakan kunci yang sama. Dengan kata lain, tidak boleh ada dua blok *counter* yang identik, terlepas dari kapan dan dalam pesan tersebut digunakan. *Counter* dapat direpresentasikan sebagai fungsi inkremental (*incrementing*), yaitu suatu fungsi yang menambahkan satu nilai terhadap *counter* saat ini secara modulus terhadap suatu integer.

Misalnya, jika digunakan *block cipher* dengan panjang blok $b = 128$ bit, maka *counter* juga akan terdiri atas 128 bit. Fungsi *incrementing* dapat dibatasi hanya pada m bit terendah, dengan $m \leq b$. Dalam hal ini, *counter* akan bertambah dengan kelipatan satu modulus 2^m .

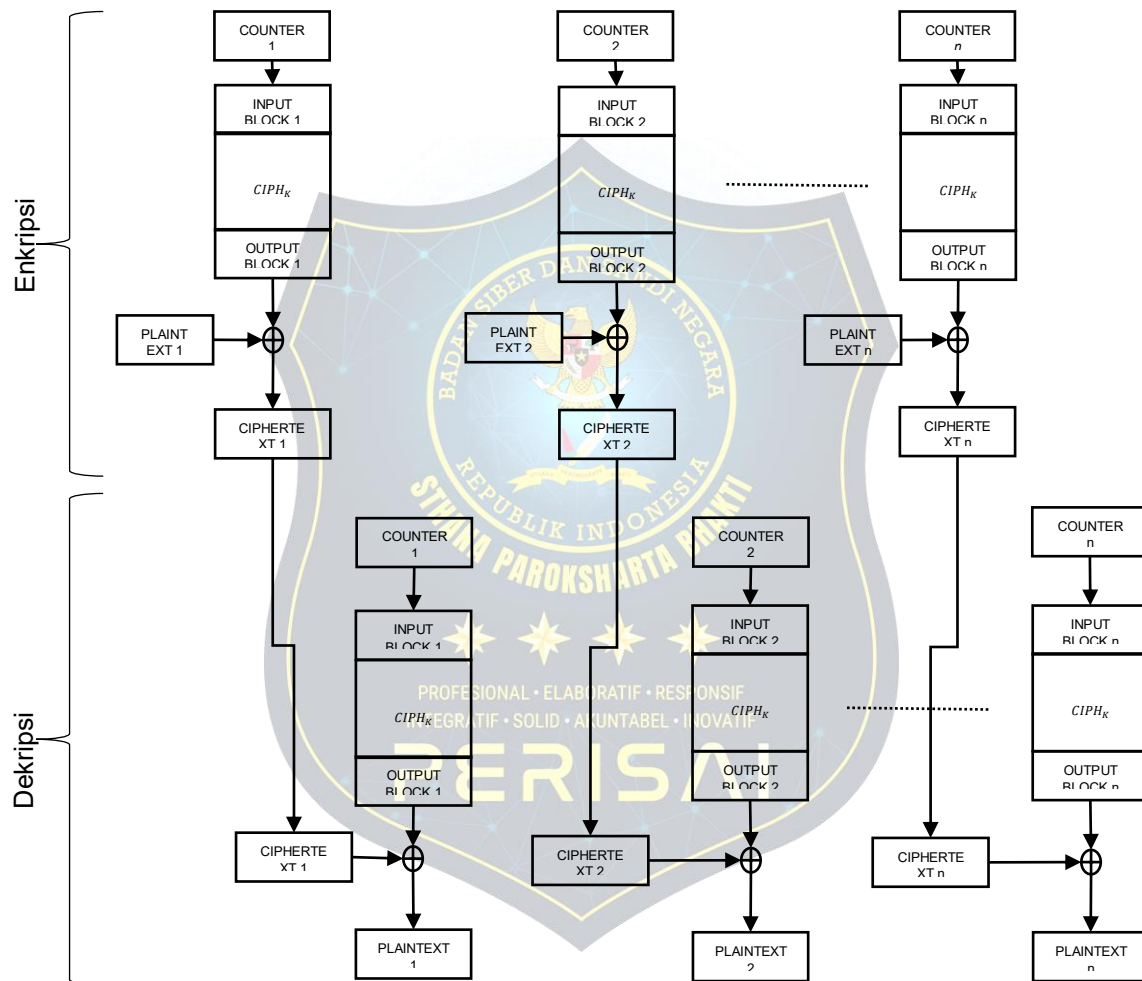
Ketika nilai *counter* mencapai batas maksimum $2^m - 1$, *counter* akan berulang (*wrap around*) ke nol. Oleh karena itu, jumlah blok pesan yang dapat dienkripsi menggunakan satu nilai *counter* awal tertentu di bawah satu kunci harus tidak melebihi 2^m . Jika jumlahnya melebihi batas ini, maka nilai *counter* akan digunakan ulang, dan hal ini dapat membuka potensi serangan untuk memulihkan *plaintext*.

Dalam NIST SP 800-38A [11], blok-blok *counter* untuk suatu pesan dinotasikan dengan $T_1, T_2, \dots, T_n$. Metode untuk menghasilkan *counter* dibahas dalam Lampiran B NIST SP 800-38A [11]. Dengan diberikan counter $T_1, T_2, \dots, T_n$, mode operasi CTR didefinisikan sebagai berikut:

$$\begin{aligned} \text{Enkripsi : } O_j &= CIPH_K(T_j) && \text{untuk } j = 1, 2, \dots, n; \\ C_j &= P_j \oplus O_j && \text{untuk } j = 1, 2, \dots, n - 1; \\ C_n^* &= P_n^* \oplus MSB_u(O_n) \end{aligned}$$

$$\begin{aligned} \text{Dekripsi : } O_j &= CIPH_K(T_j) && \text{untuk } j = 1, 2, \dots, n; \\ P_j &= C_j \oplus O_j && \text{untuk } j = 1, 2, \dots, n-1; \\ P_n^* &= C_n^* \oplus MSB_u(O_n) \end{aligned}$$

Proses enkripsi dan dekripsi *block cipher* dengan mode CTR ditunjukkan pada Gambar 5.



Gambar 5. Enkripsi dan dekripsi dengan mode operasi CTR digambar ulang dari [11].

Dalam proses enkripsi CTR, *forward cipher function* dijalankan untuk setiap blok *counter*, dan blok *output* yang dihasilkan kemudian menggunakan operasi XOR dengan blok-blok *plaintext* yang bersesuaian untuk menghasilkan blok-blok *ciphertext*. Untuk blok terakhir, yang mungkin berupa blok parsial sepanjang u bit, hanya

u *most significant bit* dari blok *output* terakhir yang digunakan dalam operasi XOR, sedangkan $b - u$ bit sisanya dari blok *output* tersebut diabaikan.

Dalam proses dekripsi CTR, *forward cipher function* juga dijalankan untuk setiap blok *counter* dan blok *output* yang dihasilkan menggunakan operasi XOR dengan blok-blok *ciphertext* yang bersesuaian untuk menghasilkan blok-blok *plaintext*. Untuk blok terakhir, yang mungkin berupa blok parsial sepanjang u bit, hanya u *most significant bit* dari blok *output* terakhir yang digunakan dalam operasi XOR, sedangkan $b - u$ bit sisanya dari blok *output* tersebut diabaikan.

Baik dalam enkripsi maupun dekripsi pada mode CTR, seluruh *forward cipher function* dapat dijalankan secara paralel, karena setiap blok *counter* bersifat independen satu sama lain. Dengan demikian, setiap blok *plaintext* yang bersesuaian dengan suatu blok *ciphertext* tertentu dapat dipulihkan secara mandiri tanpa bergantung pada blok lainnya, selama blok *counter* yang sesuai dapat ditentukan. Selain itu, *forward cipher function* juga dapat diterapkan pada blok-blok *counter* sebelum data *plaintext* atau *ciphertext* tersedia, sehingga memungkinkan proses *precomputation* untuk meningkatkan efisiensi sistem.

Menurut NIST SP 800-38A [11], terdapat dua pendekatan untuk menetapkan nilai awal *counter*, yaitu:

1) Pendekatan *Counter Global*

Dalam pendekatan ini, seluruh pesan *plaintext* dianggap sebagai satu kesatuan. Nilai *counter* akan diinkrementasi secara terus menerus dari nilai awal untuk semua pesan atau *file*. Pendekatan ini cocok digunakan untuk sistem yang menjamin tidak adanya pengulangan *counter* antar *file*.

2) Pendekatan *Nonce* + *Counter*

Counter 128 bit dibagi menjadi dua bagian yang masing-masing berukuran $b/2$ bit:

- *Nonce* (N_i) berubah setiap kali pesan baru dienkripsi, menjamin keunikan antar pesan.
- *Counter* yang diinkrementasi untuk setiap blok dalam pesan

Formulasi *counter* untuk blok ke- j dan *file* ke- i adalah:

$$T_{i,j} = N_i || (j \bmod 2^m) \text{ dengan } i = 1, 2, 3$$

Untuk menjamin keamanan, *Nonce* (N_i) harus unik untuk setiap pesan *plaintext* yang dienkripsi menggunakan kunci yang sama. Penggunaan kembali *nonce* dapat menyebabkan *keystream reuse* yang berpotensi membuka celah terhadap serangan yang memanfaatkan XOR dari dua *ciphertext* berbeda.

Dalam mode operasi CTR, IV tidak perlu dijaga kerahasiaannya. Oleh karena itu, IV dapat dikirimkan dalam bentuk *plaintext* bersama *ciphertext* atau dapat dikonstruksi ulang oleh penerima. Menurut NIST SP 800-38A [11], terdapat dua metode untuk menghasilkan IV secara acak dan aman, yaitu:

1) Metode Enkripsi *Nonce*

IV dihasilkan dengan cara mengenkripsi sebuah *nonce* menggunakan *block cipher* yang sama dan dengan kunci yang sama. Untuk menjamin keunikan IV, *nonce* dapat diinkrementasi setiap proses enkripsi baru dilakukan.

2) Penggunaan *Counter* atau Nomor Pesan

Jika mode yang digunakan memerlukan IV sebagai *nonce*, maka IV dapat dibentuk dari nomor pesan, nomor sesi atau *counter* yang menjamin keunikan.

Keunggulan utama dari mode CTR adalah kemampuannya untuk diproses secara paralel. Tidak terdapat ketergantungan antar blok selama proses enkripsi maupun dekripsi. Hal ini memungkinkan:

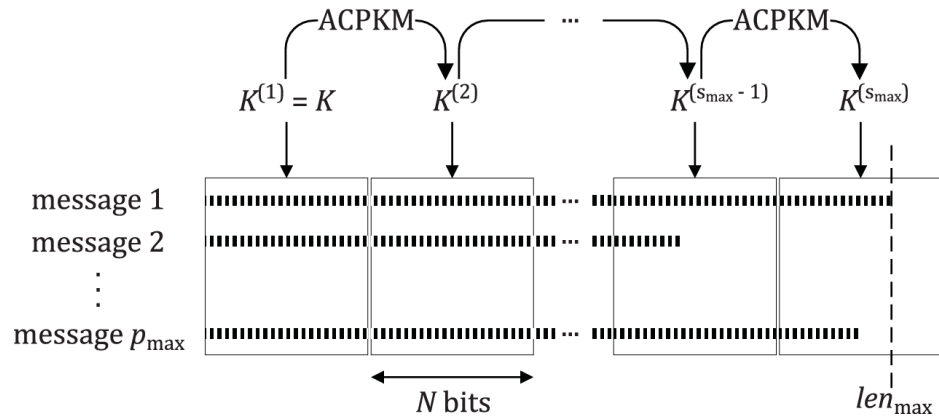
- Pelaksanaan fungsi *cipher* terhadap blok *counter* secara simultan.

- Pemrosesan *ciphertext* dan *plaintext* secara independen antar blok.
- Penerapan fungsi *cipher* terhadap *counter* dapat dilakukan bahkan sebelum data *plaintext* atau *ciphertext* tersedia, menjadikannya ideal untuk implementasi dalam sistem berperforma tinggi dan bersifat *streaming*.

f. Counter advanced cryptographic prolongation of key material (CTR-ACPKM).

Mode operasi CTR-ACPKM merupakan pengembangan dari mode CTR yang dirancang untuk memperpanjang *key lifetime* dalam sistem kriptografi simetris. Mode operasi ini mengimplementasikan mekanisme transformasi terhadap *section key* secara berkala selama proses enkripsi satu pesan. Setiap pesan diproses dengan menggunakan *section key* pertama yang sama sebagai titik awal, dan setiap *section key* diperbarui setelah satu bagian sepanjang N bit telah selesai diproses [13]. Tujuan utama dari mode operasi ini adalah meningkatkan keamanan terhadap serangan yang memanfaatkan penggunaan kunci jangka panjang, khususnya dalam skenario enkripsi data berukuran besar atau komunikasi berkelanjutan.

Proses enkripsi tetap menggunakan pendekatan mode operasi CTR, yaitu menerapkan fungsi enkripsi terhadap blok-blok *counter* untuk menghasilkan blok *output* yang kemudian di-XOR dengan blok *plaintext*. Namun, berbeda dengan CTR konvensional, CTR-ACPKM menyisipkan mekanisme pembaruan kunci setelah setiap *section* berukuran tetap (N bit) selesai diproses. Transformasi kunci ini bertujuan untuk mengurangi risiko serangan akibat penggunaan kunci yang terlalu panjang atau berulang. Ide utama mode operasi CTR-ACPKM terdapat pada Gambar 6.



Gambar 6. Mode CTR-ACPKM

Digambar ulang dari [13].

Dengan,

K : kunci awal (*intial key*)

$p_{\max}$: Jumlah maksimum pesan yang dapat dienkripsi di bawah satu kunci awal K

$len_{\max}$: Panjang maksimum pesan (dalam bit)

$s_{\max}$: $\lceil len_{\max}/N \rceil$

Selama pemrosesan pesan *plaintext* P dengan panjang len (dalam satuan bit) menggunakan mode enkripsi CTR-ACPKM, pesan tersebut dibagi menjadi $s = \lceil len / N \rceil$ sections, yang dinotasikan sebagai $P^1, P^2 \dots P^s$ dengan panjang setiap bagian P^z sebesar N bit untuk $1 \leq z \leq s - 1$, sedangkan bagian terakhir P^s dapat memiliki panjang $\leq N$ bit. Bagian pertama dari setiap pesan diproses menggunakan *section key* pertama ($K^{(1)}$) yang nilainya sama dengan kunci awal K . Untuk memproses bagian ke- $(z + 1)$ dari setiap pesan, *section key* berikutnya $K^{(z+1)}$ dihitung menggunakan transformasi ACPKM. Transformasi ACPKM menerima input berupa kunci K^z dengan panjang k bit, dan menghasilkan *output* berupa kunci $K^{(z+1)}$ dengan panjang yang sama, yaitu k bit, yang dihitung sebagai berikut:

$$K^{(z+1)} = ACPKM(K^{(z)}) = k \sim (eK^{(z)}(D_1) | \dots | eK^{(z)}(D_l)),$$

dengan $J = \lceil k/n \rceil$, and $D_1, D_2, \dots, D_J$ adalah *string* n bit yang dihitung sebagai berikut:

$$D_1 | D_2 | \dots | D_J = (J \cdot n) \sim D,$$

D adalah konstanta sepanjang 1024 bit yang didefinisikan sebagai berikut (D dinyatakan dalam notasi heksadesimal, dengan ketentuan setiap dua digit heksadesimal merepresentasikan satu *byte*, dan setiap 8 *byte* dipisahkan oleh satu spasi untuk meningkatkan visibilitas):

8081828384858687 88898a8b8c8d8e8f 9091929394959697 98999a9b9c9d9e9f
a0a1a2a3a4a5a6a7 a8a9aaabacadaeaf b0b1b2b3b4b5b6b7 b8b9babbbcbdbef
c0c1c2c3c4c5c6c7 c8c9cacbcccdcecf d0d1d2d3d4d5d6d7 d8d9daddbcddeedf
e0e1e2e3e4e5e6e7 e8e9eaebeceedeef f0f1f2f3f4f5f6f7 f8f9fafbfcfdfeff

Mode operasi CTR-ACPKM sering digunakan pada aplikasi yang memerlukan keamanan jangka panjang atau pemrosesan data terus-menerus, seperti komunikasi militer, VPN permanen, dan sistem *backup* otomatis. Beberapa keunggulan CTR-ACPKM, yaitu:

- Perindungan terhadap serangan kunci ulang karena setiap bagian pesan dienkripsi menggunakan kunci berbeda, sehingga mengurangi risiko korelasi antar blok.
- Perpanjangan *key lifetime* untuk menghindari penggunaan satu kunci tunggal untuk seluruh data, yang sangat penting pada sistem dengan trafik besar.
- Mitigasi serangan analisis statistik dan diferensial dengan meminimalkan pola berulang akibat kunci yang tetap.
- CTR-ACPKM mempertahankan seluruh keunggulan mode CTR, seperti: paralelisme penuh, efisiensi tinggi, dan skalabilitas.

2. Enkripsi Terautentikasi (*Authenticated Encryption*)

Mode AE merupakan skema kriptografi yang dirancang untuk secara simultan menyediakan tiga properti keamanan utama, yaitu

konfidensialitas, autentikasi dan integritas. Terdapat dua mode AE berbasis *block cipher* yang direkomendasikan, yaitu *Galois/Counter Mode* (GCM), dan *Counter Mode with Cipher Block Chaining Message Authentication Code* (CCM). Penjelasan mengenai mode GCM dan CCM adalah sebagai berikut.

a. *Galois/Counter Mode* (GCM)

GCM merupakan mode operasi yang dapat memberikan jaminan konfidensialitas dan autentikasi data. Operasi GCM bergantung pada pemilihan algoritma *block cipher* yang digunakan dengan ukuran blok minimal 128 bit, misalnya algoritma AES. Kunci GCM identik dengan kunci *block cipher* yang digunakan.

Untuk setiap kunci tertentu, *block cipher* yang mendasari mode ini terdiri atas dua fungsi yang merupakan invers satu sama lain. Pemilihan *block cipher* juga mencakup penetapan salah satu dari dua fungsi tersebut sebagai *forward cipher function*, sebagaimana ditentukan dalam spesifikasi algoritma *block cipher*. GCM tidak menggunakan fungsi *cipher invers* (dekripsi).

Forward cipher function adalah permutasi terhadap *string* bit berukuran tetap (disebut blok), dengan panjang blok disebut sebagai ukuran blok. Fungsi ini bergantung pada kunci K dan dilambangkan sebagai $CIPH_K$.

Block cipher yang digunakan dalam GCM harus memenuhi kriteria sebagai berikut:

- 1) Merupakan algoritma *block cipher* yang telah disetujui dan tercantum dalam Algoritma AKI;
- 2) Memiliki ukuran blok sebesar 128 bit;
- 3) Memiliki ukuran kunci minimal 128 bit.

Kunci harus dibangkitkan secara acak seragam, atau sedekat mungkin dengan distribusi seragam, artinya setiap kemungkinan kunci memiliki peluang yang hampir sama untuk dibangkitkan.

Dengan demikian, kunci yang digunakan akan bersifat *fresh*, yaitu sangat mungkin berbeda dari kunci-kunci yang pernah digunakan sebelumnya. Kunci harus dirahasiakan dan digunakan secara eksklusif untuk operasi GCM dengan *block cipher* yang dipilih.

Dua fungsi yang berlaku pada mode operasi GCM meliputi *authenticated encryption* dan *authentication decryption*. Fungsi *authenticated encryption* dan *authentication decryption* didefinisikan sebagai berikut dan dirincikan lebih detail dalam NIST SP 800-38D [15].

Fungsi *authenticated encryption* sesuai NIST SP 800-38D [15].

Algorithm 4: $GCM-AE_K(IV, P, A)$

Prerequisites:

approved block cipher *CIPH* with a 128-bit block size;
key *K*;
definitions of supported input-output lengths;
supported tag length *t* associated with the key.

Input:

initialization vector *IV* (whose length is supported);
plaintext *P* (whose length is supported);
additional authenticated data *A* (whose length is supported).

Output:

ciphertext *C*;
authentication tag *T*.

Steps:

1. Let $H = CIPH_K(0^{128})$.
2. Define a block, J_0 , as follows:
If $len(IV) = 96$, then let $J_0 = IV \parallel 0^{31} \parallel 1$.
If $len(IV) \neq 96$,
then let $s = 128 \lceil len(IV)/128 \rceil - len(IV)$,
and let $J_0 = GHASH_H(IV \parallel 0^{s+64} \parallel [len(IV)]_{64})$.
3. Let $C = GCTR_K(inc_{32}(J_0), P)$.
4. Let $u = 128 \cdot \lceil len(C)/128 \rceil - len(C)$
and let $v = 128 \cdot \lceil len(A)/128 \rceil - len(A)$.
5. Define a block, S , as follows:
 $S = GHASH_H(A \parallel 0^v \parallel C \parallel 0^u \parallel [len(A)]_{64} \parallel [len(C)]_{64})$.
6. Let $T = MSB_t(GCTR_K(J_0, S))$.
7. Return (C, T) .

Fungsi *authenticated decryption* sesuai NIST SP 800-38D [15].

Algorithm 5: $\text{GCM-AD}_K(IV, C, A, T)$

Prerequisites:

approved block cipher $CIPH$ with a 128-bit block size;
key K ;
definitions of supported input-output lengths;
supported tag length t associated with the key.

Input:

initialization vector IV ;
ciphertext C ;

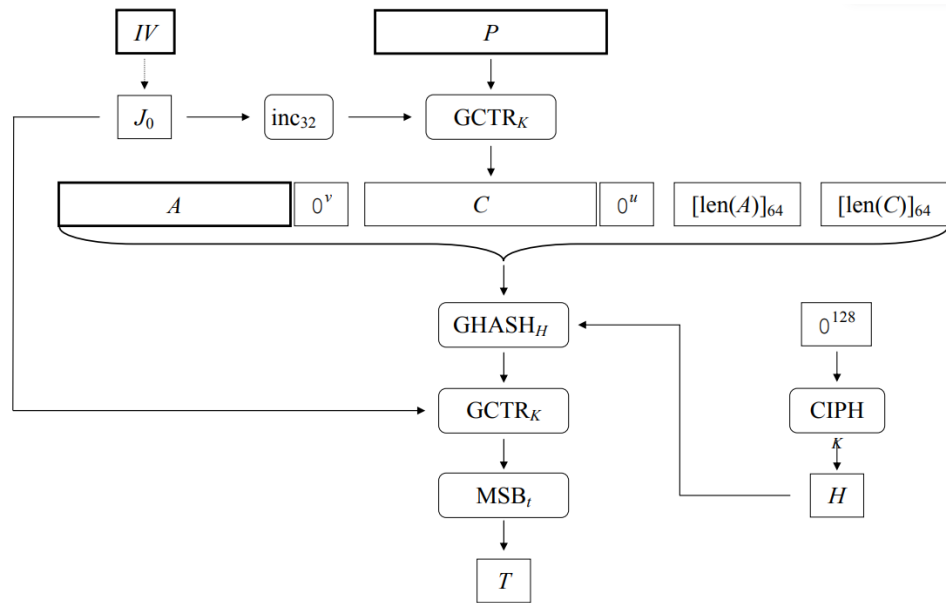
Output:

plaintext P or indication of inauthenticity FAIL.

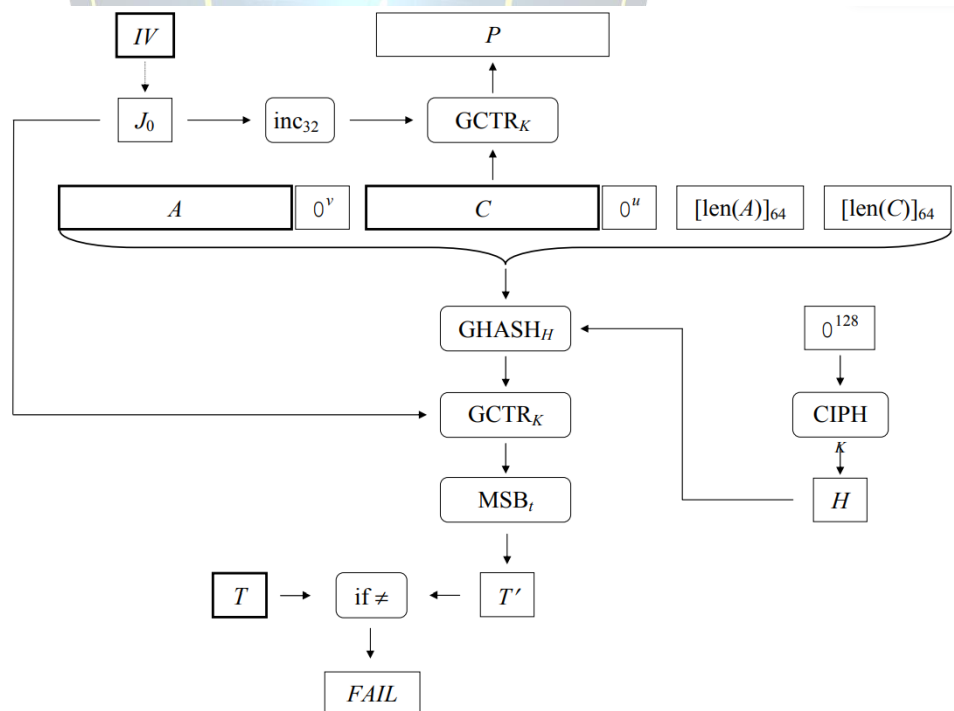
Steps:

1. If the bit lengths of IV , A or C are not supported, or if $\text{len}(T) \neq t$, then return FAIL.
2. Let $H = CIPH_K(0^{128})$.
3. Define a block, J_0 , as follows:
If $\text{len}(IV) = 96$, then let $J_0 = IV \parallel 0^{31} \parallel 1$.
If $\text{len}(IV) \neq 96$,
then let $s = 128 \lceil \text{len}(IV)/128 \rceil - \text{len}(IV)$,
and let $J_0 = GHASH_H(IV \parallel 0^{s+64} \parallel [\text{len}(IV)]_{64})$.
4. Let $P = GCTR_K(\text{inc}_{32}(J_0), C)$.
5. Let $u = 128 \cdot \lceil \text{len}(C)/128 \rceil - \text{len}(C)$
and let $v = 128 \cdot \lceil \text{len}(A)/128 \rceil - \text{len}(A)$.
6. Define a block, S , as follows:
 $S = GHASH_H(A \parallel 0^v \parallel C \parallel 0^u \parallel [\text{len}(A)]_{64} \parallel [\text{len}(C)]_{64})$.
7. Let $T' = \text{MSB}_t(GCTR_K(J_0, S))$.
8. If $T = T'$, then return P ; else return FAIL.

Fungsi *authenticated encryption* dan *authenticated decryption* mode operasi GCM ditunjukkan pada Gambar 7 dan Gambar 8.



Gambar 7. Fungsi *authenticated encryption* mode operasi GCM digambar ulang dari [15].



Gambar 8. Fungsi *authenticated decryption* mode operasi GCM digambar ulang dari [15].

GCM memberikan jaminan terhadap konfidensialitas data melalui penggunaan varian dari mode operasi *counter* untuk proses enkripsi.

Selain itu, GCM juga memberikan jaminan autentikasi terhadap data rahasia (hingga sekitar 64 gigabyte per satu kali pemanggilan) dengan memanfaatkan fungsi *hash* universal yang didefinisikan pada *binary Galois Field*, sebagaimana dijelaskan dalam NIST SP 800-38D [15].

GCM juga mampu menyediakan jaminan autentikasi untuk data tambahan yang tidak dienkripsi, dengan panjang data yang secara praktis tidak terbatas dalam satu kali pemrosesan. Apabila input GCM dibatasi hanya pada data yang tidak akan dienkripsi, maka spesialisasi dari GCM ini disebut sebagai GMAC, yang berfungsi murni sebagai mode autentikasi atas data input. Dalam NIST SP 800-38D [15], seluruh pernyataan mengenai GCM juga berlaku untuk GMAC.

GCM memberikan jaminan autentikasi yang lebih kuat dibandingkan metode non-kriptografis seperti *checksum* atau kode deteksi kesalahan. Secara khusus, GCM mampu mendeteksi: modifikasi data yang tidak disengaja, serta modifikasi yang disengaja dan tidak sah.

Salah satu hal penting dalam penggunaan GCM adalah bahwa pelanggaran terhadap persyaratan keunikan *string* inisialisasi dapat secara signifikan melemahkan jaminan keamanan. Oleh karena itu, mode operasi ini tidak boleh digunakan tanpa memastikan bahwa persyaratan keunikan tersebut benar-benar terpenuhi. Beberapa pertimbangan praktis lebih lanjut dibahas dalam NIST SP 800-38D [15].

b. Counter Mode with Cipher Block Chaining Message Authentication Code (CCM)

CCM merupakan kombinasi dari mode *counter* untuk menjamin konfidensialitas dan *Cipher Block Chaining Message Authentication Code* (CBC-MAC) untuk menjamin integritas dan autentikasi data. CCM didefinisikan dalam NIST SP 800-38C [14].

CCM didasarkan pada algoritma *block cipher*, dengan ukuran blok sebesar 128 bit, seperti algoritma AES. Seperti halnya mode operasi lainnya, kunci untuk *block cipher* harus ditetapkan terlebih dahulu di antara pihak-pihak yang akan bertukar data. Oleh karena itu, CCM harus diimplementasikan dalam struktur manajemen kunci yang dirancang dengan baik.

Input ke CCM mencakup tiga elemen, yaitu:

- 1) Data yang akan diautentikasi sekaligus dienkripsi, disebut *payload*;
- 2) Data terkait (*associated data*), misalnya *header*, yang akan diautentikasi namun tidak dienkripsi;
- 3) Nilai unik, yang ditetapkan untuk *payload* dan *associated data*.

CCM terdiri atas dua proses yang saling berkaitan:

- 1) Pembangkitan dan enkripsi (*generation-encryption*)

Dalam proses ini, *Cipher Block Chaining* (CBC) diterapkan pada *payload*, *associated data*, dan *nonce* untuk menghasilkan sebuah MAC; kemudian, enkripsi mode CTR diterapkan pada MAC dan *payload* untuk mengubahnya menjadi bentuk *ciphertext*. Dengan demikian, proses *generation-encryption* pada CCM akan menambahkan ukuran *payload* sebesar ukuran MAC.

- 2) Dekripsi dan verifikasi (*decryption-verification*)

Pada proses ini, dekripsi mode *counter* diterapkan pada *ciphertext* untuk mendapatkan kembali MAC dan *payload* yang bersesuaian. Kemudian, CBC diterapkan pada *payload*, data terkait, dan *nonce* yang diterima untuk memverifikasi kode MAC. Verifikasi yang berhasil dapat memberikan jaminan bahwa *payload* dan data terkait berasal dari sumber sah yang memiliki akses ke kunci.

Dua proses yang berlaku pada mode operasi CCM meliputi *generation-encryption* dan *decryption-verification*. Fungsi *generation-encryption* dan *decryption-verification* didefinisikan sebagai berikut dan dirincikan lebih detail dalam NIST SP 800-38C [14].

Proses *generation-encryption* berdasarkan NIST SP 800-38C [14].

Prerequisites:

block cipher algorithm;
key K ;
counter generation function;
formatting function;
MAC length $Tlen$.

Input:

valid nonce N ;
valid payload P of length $Plen$ bits;
valid associated data A ;

Output:

ciphertext C .

Steps:

1. Apply the formatting function to (N, A, P) to produce the blocks $B_0, B_1, \dots, B_r$.
2. Set $Y_0 = CIPH_K(B_0)$.
3. For $i = 1$ to r , do $Y_i = CIPH_K(B_i \oplus Y_{i-1})$.
4. Set $T = MSB_{Tlen}(Y_r)$.
5. Apply the counter generation function to generate the counter blocks $Ctr_0, Ctr_1, \dots, Ctr_m$, where $m = \lceil Plen/128 \rceil$.
6. For $j = 0$ to m , do $S_j = CIPH_K(Ctr_j)$.
7. Set $S = S_1 || S_2 || \dots || S_m$.
8. Return $C = (P \oplus MSB_{Plen}(S)) || (T \oplus MSB_{Tlen}(S_0))$.

Proses *decryption-verification* berdasarkan NIST SP 800-38C [14].

Prerequisites:

block cipher algorithm;
key K ;
counter generation function;
formatting function;
valid MAC length $Tlen$.

Input:

valid nonce N ;
valid associated data A ;
purported ciphertext C of length $Clen$ bits;

Output:

either the payload P or INVALID.

Steps:

1. If $Clen \leq Tlen$, then return INVALID.

2. Apply the counter generation function to generate the counter blocks $Ctr_0, Ctr_1, \dots, Ctr_m$, where $m = \lceil (Clen - Tlen)/128 \rceil$.
3. For $j = 0$ to m , do $S_j = CIPH_K(Ctr_j)$.
4. Set $S = S_1 || S_2 || \dots || S_m$.
5. Set $P = MSB_{Clen-Tlen}(C) \oplus MSB_{Clen-Tlen}(S)$.
6. Set $T = LSB_{Tlen}(C) \oplus MSB_{Tlen}(S_0)$.
7. If N , A , or P is not valid, then return INVALID, else apply the formatting function to (N, A, P) to produce the blocks $B_0, B_1, \dots, B_r$.
8. Set $Y_0 = CIPH_K(B_0)$.
9. For $i = 1$ to r , do $Y_i = CIPH_K(B_i \oplus Y_{i-1})$.
10. If $T \neq MSB_{Tlen}(Y_r)$, then return INVALID, else return P .

IV. SANDI ALIR (STREAM CIPHER)

A. Algoritma

Terdapat empat algoritma kriptografi *stream cipher* yang direkomendasikan dalam algoritma AKI. Keempat algoritma tersebut masih dikategorikan sebagai akseptabel untuk digunakan hingga setidaknya tahun 2030. Rincian setiap algoritma beserta karakteristik utamanya disajikan pada Tabel 2.

Tabel 2. Parameter teknis Algoritma *Stream Cipher*.

No	Nama Algoritma	Panjang Kunci (bit)	Panjang IV (bit)	Bit per cycle (siklus)	Max keystream (bit)	Jenis Algoritma	Standar
1	HC-128	128	128	32	2^{37}	<i>synchronous stream cipher</i>	Kompetisi eSTREAM
2	Salsa 20/20	128 atau 256	64	512	2^{73}	<i>synchronous stream cipher</i>	Kompetisi eSTREAM
3	Sosemanuk	128	128	128	2^{71}	<i>synchronous stream cipher</i>	Kompetisi eSTREAM
4	ChaCha20	256	96	512	2^{73}	<i>synchronous stream cipher</i>	IETF RFC 8439

HC-128 menghasilkan *keystream* sebesar 32 bit pada setiap siklus. Dengan satu pasangan kunci dan IV, algoritma ini dapat menjalankan hingga 2^{32} siklus, sehingga total *keystream* maksimum yang dapat dihasilkan mencapai $2^{32} \times 32 = 2^{37}$ bit [20].

Salsa20/20 menghasilkan *keystream* sebesar 512 bit per siklus. Dengan menggunakan 64 bit *block counter*, algoritma ini memungkinkan hingga 2^{64} siklus untuk setiap pasangan kunci dan IV, menghasilkan total *keystream* sebesar $2^{64} \times 512 = 2^{73}$ bit [21].

Sosemanuk memproduksi *keystream* sebesar 128 bit pada setiap siklus. Meskipun spesifikasi resmi tidak mencantumkan batas eksplisit jumlah siklus, asumsi umum dalam desain algoritma *stream cipher* menyatakan bahwa batas maksimum adalah sebesar 2^{64} siklus per pasangan kunci dan IV. Dengan demikian, total *keystream* yang dapat dihasilkan mencapai $2^{64} \times 128 = 2^{71}$ bit [22].

ChaCha20, seperti Salsa20/20, menghasilkan *keystream* sepanjang 512 bit untuk setiap siklus operasi. Dengan menggunakan *block counter* 64 bit, algoritma ini mampu melakukan hingga 2^{64} siklus dalam satu sesi enkripsi. Oleh karena itu, jumlah maksimum *keystream* yang dapat dihasilkan adalah $2^{64} \times 512$ bit, yaitu sebesar 2^{73} bit [21].

B. Penggunaan

Berdasarkan SNI ISO/IEC 18033-4:2011 [23], algoritma *stream cipher* adalah mekanisme enkripsi yang memanfaatkan *keystream* untuk mengenkripsi *plaintext*, baik secara bit per bit maupun dalam unit blok berukuran kecil. Terdapat dua jenis utama *stream cipher*:

1. *Synchronous stream cipher*, yaitu algoritma yang menghasilkan *keystream* berdasarkan kunci rahasia dan IV, tanpa bergantung pada *ciphertext* yang dihasilkan sebelumnya.
2. *Self-synchronizing stream cipher*, yang membangkitkan *keystream* tidak hanya dari kunci rahasia dan IV, tetapi juga menggunakan sebagian *ciphertext* sebelumnya sebagai input.

Secara umum, algoritma *stream cipher* terdiri atas lima komponen utama:

1. Fungsi *key setup*

Bertugas untuk menginisialisasi status internal algoritma menggunakan kunci rahasia sebelum proses pembangkitan *keystream* dimulai.

2. Fungsi *IV setup*

Menyesuaikan status internal berdasarkan *IV* atau *nonce* untuk memastikan bahwa setiap sesi enkripsi tetap unik, meskipun menggunakan kunci yang sama.

3. Fungsi *keystream generator*

Menghasilkan *keystream* sebagai deretan bit yang digunakan dalam proses enkripsi atau dekripsi. Komponen ini beroperasi dengan mempertahankan status internal dan menerima input berupa kunci, *IV*, serta, dalam kasus tertentu, *ciphertext* sebelumnya.

4. Fungsi *output*

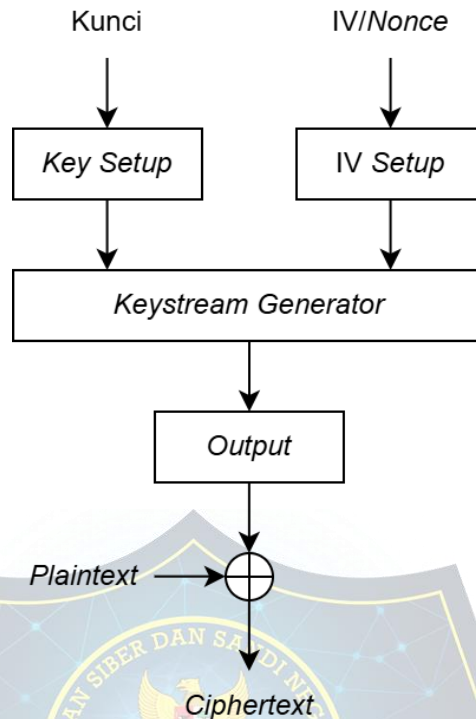
Mengubah status internal terkini menjadi *keystream* yang siap digunakan dalam proses enkripsi atau dekripsi.

5. Fungsi enkripsi dan dekripsi.

Melakukan operasi XOR antara *keystream* dan *plaintext* untuk menghasilkan *ciphertext* pada proses enkripsi, atau antara *keystream* dan *ciphertext* untuk memperoleh kembali *plaintext* pada proses dekripsi.

Agar keutuhan dan keamanan proses enkripsi tetap terjaga, setiap implementasi algoritma *stream cipher* harus mengikuti urutan struktur komponen di atas. Ilustrasi umum proses enkripsi *stream cipher* disajikan pada Gambar 9.





Gambar 9. Skema Umum Enkripsi Algoritma *Stream Cipher*.

Dalam penerapan algoritma *stream cipher*, terdapat sejumlah aspek teknis dan keamanan yang wajib diperhatikan untuk memastikan efektivitas dan ketahanan sistem terhadap serangan kriptografi, yaitu:

1. Pembatasan *keystream* per pasangan kunci dan IV.

Untuk menjaga keamanan algoritma *stream cipher* yang tercantum dalam daftar Algoritma AKI, diperlukan mekanisme kontrol yang membatasi jumlah *keystream* yang dihasilkan dari setiap pasangan kunci dan IV agar tidak melebihi batas maksimum sebagaimana ditetapkan dalam Tabel 2. Pelanggaran terhadap batas ini dapat membuka celah keamanan serius, termasuk potensi serangan berbasis analisis *keystream* yang berulang untuk merekonstruksi *plaintext*.

2. Penggunaan IV atau *nonce* yang unik.

IV atau *nonce* wajib bersifat unik dan tidak boleh digunakan ulang dalam proses enkripsi yang menggunakan kunci yang sama. Penggunaan ulang IV atau *nonce* dapat menyebabkan kebocoran *keystream* dan melemahkan kekuatan kriptografi, sehingga prinsip ini harus diterapkan secara konsisten dalam setiap implementasi.

3. Perlindungan terhadap integritas dan autentikasi data.

Sifat *malleable* pada algoritma *stream cipher* memungkinkan perubahan bit pada *ciphertext* menghasilkan perubahan terprediksi pada hasil dekripsi. Akibatnya, algoritma ini tidak menjamin integritas maupun autentikasi data secara bawaan. Oleh karena itu, pada SE yang memerlukan jaminan keamanan menyeluruh, penggunaan *stream cipher* harus dikombinasikan dengan skema MAC, seperti HMAC-SHA256. Pendekatan yang lebih ideal adalah penggunaan dalam kerangka *Authenticated Encryption with Associated Data* (AEAD).

Khusus untuk ChaCha20, kombinasi dengan Poly1305 dalam skema ChaCha20-Poly1305, sebagaimana didefinisikan dalam IETF RFC 8439, direkomendasikan sebagai solusi terpadu yang menjamin kerahasiaan, integritas, dan autentikasi data.

4. Kebijakan masa berlaku algoritma.

Keempat algoritma *stream cipher* yang direkomendasikan saat ini dikategorikan sebagai akseptabel hingga tahun 2030. Setelah periode tersebut, penggunaannya tidak lagi disarankan. Oleh karena itu, disarankan untuk mulai merencanakan strategi transisi menuju algoritma yang lebih tahan terhadap ancaman dari komputasi kuantum.

5. Acuan spesifikasi teknis.

Setiap algoritma memiliki dokumentasi teknis resmi yang dapat dijadikan referensi implementasi. HC-128, Salsa20/20, dan Sosemanuk merupakan finalis dalam *eSTREAM Project* oleh ECRYPT, dengan spesifikasi terperinci yang tersedia dalam laporan resmi proyek tersebut [20] [21] [22].

Dokumentasi lengkap mengenai penggunaan ChaCha20, termasuk kombinasi teknisnya dengan Poly1305, tercantum dalam IETF RFC 8439 [24]. Standar ini menjelaskan secara rinci aspek implementasi, panjang *nonce*, batas maksimum blok yang diizinkan, serta parameter kunci. Selain itu, [24] juga membahas penerapan ChaCha20-Poly1305 dalam protokol kriptografi modern seperti TLS 1.3 dan WireGuard, dengan fokus pada keamanan praktis dan efisiensi operasional.

V. FUNGSI *HASH* (*HASH FUNCTION*)

A. Algoritma

Terdapat delapan fungsi *hash* yang direkomendasikan sebagai Algoritma AKI dan dapat digunakan untuk seluruh kategori SE. Kedelapan algoritma tersebut masih berstatus akseptabel hingga tahun 2030. Daftar algoritma beserta standar rujukannya ditunjukkan pada Tabel 3.

Tabel 3. Fungsi *Hash* dan Standar Rujukannya.

No	Fungsi <i>Hash</i>	Standar
1.	SHA-384	• FIPS 180-4 [25] • ISO/IEC 10118-3 [26]
2.	SHA-512/256	• FIPS 180-4 [25] • ISO/IEC 10118-3 [26]
3.	WHIRPOOL	• FIPS PUB 202 [27] • ISO/IEC 10118-3 [26]
4.	SHA3-256	• FIPS PUB 202 [27] • ISO/IEC 10118-3 [26]
5.	SHA3-384	• FIPS PUB 202 [27] • ISO/IEC 10118-3 [26]
6.	SHA3-512	• FIPS PUB 202 [27] • ISO/IEC 10118-3 [26]
7.	SHAKE256	FIPS PUB 202 [27]
8.	Blake2	RFC 7693 [28]

Meskipun tidak termasuk dalam daftar fungsi *hash* terbaru, SHA-256 dan SHA-512 dari famili SHA-2 masih banyak digunakan. Kedua algoritma ini telah lama menjadi standar dalam berbagai aplikasi keamanan, termasuk integritas data, tanda tangan digital, dan *blockchain*. Namun, penggunaannya mulai dipertimbangkan kembali karena keterbatasan desain.

SHA-2 menggunakan struktur Merkle–Damgård, sama seperti SHA-1 yang telah terbukti rentan terhadap kolisi. Hingga September 2025, belum ditemukan kolisi praktis pada SHA-256 maupun SHA-512. Meski demikian,

desain ini kurang fleksibel dibandingkan algoritma baru seperti SHA-3, yang berbasis Keccak dan menggunakan konstruksi *sponge* untuk meningkatkan keamanan. Salah satu kerentanan utama SHA-256 dan SHA-512 adalah serangan *length extension*. Dalam serangan ini, seorang penyerang yang mengetahui nilai *hash* dan panjang pesan asli dapat menghitung nilai *hash* dari pesan yang diperpanjang tanpa mengetahui isi pesan asli. Hal ini menjadi masalah serius ketika fungsi *hash* digunakan sebagai MAC dengan konstruksi seperti $\text{Hash}(\text{secret} || \text{message})$, karena penyerang dapat menambahkan data baru dan menghasilkan nilai *hash* yang valid tanpa mengetahui kunci rahasia. Untuk mengatasi kerentanan ini, disarankan menggunakan *Hashed Message Authentication Code* (HMAC), yang menggabungkan fungsi *hash* dengan kunci rahasia secara aman. Alternatif lain adalah beralih ke SHA-3, yang menggunakan konstruksi *sponge* dan tidak rentan terhadap serangan *length extension*, sehingga lebih aman untuk aplikasi modern.

Secara umum, proses pada fungsi *hash* terdiri atas dua tahap utama: pra-pemrosesan dan perhitungan nilai *hash*. Pra-pemrosesan meliputi penambahan *padding*, pemecahan pesan menjadi blok-blok m bit, serta penetapan nilai inisialisasi. Selanjutnya, perhitungan nilai *hash* dilakukan dengan membentuk *message schedule*, lalu menggunakan fungsi *hash*, konstanta, dan operasi *word* untuk menghasilkan nilai *hash* secara iteratif hingga diperoleh nilai *hash* akhir.

B. Penggunaan

Perbedaan utama antar fungsi *hash* terletak pada kekuatan keamanan, ukuran blok, ukuran *word*, serta panjang nilai *hash* yang dihasilkan. Tingkat keamanan masing-masing fungsi *hash*, khususnya ketika digunakan bersama algoritma kriptografi lain seperti tanda tangan digital atau HMAC, telah dibahas dalam dokumen NIST SP 800-57 dan SP 800-107 [29]. Ringkasan properti dasar dari fungsi *hash* yang direkomendasikan dapat dilihat pada Tabel 4.

Tabel 4. Properti Fungsi *Hash*.

Fungsi <i>Hash</i>	Ukuran pesan (bit)	Ukuran blok (bit)	Ukuran <i>word</i> (bit)	Ukuran nilai <i>hash</i>
SHA-384	$< 2^{128}$	1024	64	384
SHA-512/256	$< 2^{128}$	1024	64	256
WHIRPOOL	$< 2^{256}$	512	64	512
SHA3-256	$< 2^{64}$	512 (rate 1088)	64	256
SHA3-384	$< 2^{64}$	832 (rate 768)	64	384
SHA3-512	$< 2^{64}$	1024 (rate 576)	64	512
SHAKE256	$< 2^{64}$ (extendable)	576 (rate 512)	64	Variable (min 256)
Blake2	$< 2^{64}$	1024	64	Variable (up to 512)

Tabel 4 menyajikan properti dasar beberapa fungsi *hash* yang direkomendasikan, meliputi ukuran pesan, ukuran blok, ukuran *word*, dan panjang nilai *hash*. Keempat properti ini sangat penting karena memengaruhi keamanan, efisiensi, dan fleksibilitas masing-masing algoritma.

Pertama, ukuran pesan menunjukkan panjang maksimum data yang dapat diolah oleh fungsi *hash*. SHA-384 dan SHA-512/256, misalnya, mampu memproses pesan hingga kurang dari 2^{128} bit, sementara WHIRLPOOL mendukung hingga kurang dari 2^{256} bit. SHA-3 dan Blake2 mendukung pesan hingga kurang dari 2^{64} bit, dengan SHAKE256 dan Blake2 memiliki kemampuan *output* yang dapat diperluas, sehingga lebih fleksibel dalam penggunaannya. Batas ini bersifat teoretis dan jarang tercapai dalam praktik, namun tetap menjadi pertimbangan dalam desain sistem kriptografi yang aman.

Kedua, ukuran blok merujuk pada jumlah bit yang diproses fungsi *hash* dalam satu iterasi. SHA-384 dan SHA-512/256 menggunakan blok 1024 bit, sedangkan WHIRLPOOL menggunakan 512 bit. SHA-3 memiliki ukuran blok

berbeda berdasarkan variannya, misalnya SHA3-256 menggunakan blok 1152 bit dengan *rate* 1088, dan SHAKE256 menggunakan 576 bit (*rate* 512). Ukuran blok ini memengaruhi kecepatan dan efisiensi pemrosesan, serta bagaimana data dipecah untuk iterasi.

Ketiga, ukuran *word* adalah unit data internal yang digunakan fungsi *hash* untuk operasi *bitwise* dan matematis. Semua fungsi *hash* dalam Tabel 4 menggunakan *word* 64 bit, yang memungkinkan pemrosesan cepat pada arsitektur komputer modern dan mendukung operasi fungsi *hash* yang kompleks dan aman.

Keempat, ukuran nilai *hash* menunjukkan panjang *output* yang dihasilkan fungsi *hash*. SHA-384 menghasilkan 384 bit, SHA-512/256 menghasilkan 256 bit meskipun berbasis SHA-512, dan WHIRLPOOL menghasilkan 512 bit. SHA3-256, SHA3-384, dan SHA3-512 menghasilkan nilai *hash* sesuai dengan namanya, sedangkan SHAKE256 dan Blake2 mendukung ukuran nilai *hash* variabel, mulai dari minimum 256 bit hingga maksimum yang ditentukan. Panjang nilai *hash* ini menentukan tingkat keamanan fungsi *hash*, terutama terhadap serangan *collision* dan *preimage*, dengan nilai *hash* lebih panjang menurut teori lebih tahan terhadap upaya pembobolan.

Secara keseluruhan, properti-properti ini memperlihatkan perbedaan karakteristik antara fungsi *hash*. SHA-2 menawarkan keamanan yang terbukti dan nilai *hash* tetap, namun memiliki keterbatasan desain seperti rentan terhadap serangan *length extension*. SHA-3 dengan struktur *sponge* memberikan fleksibilitas *output* dan keamanan tambahan. WHIRLPOOL dan Blake2 menekankan kombinasi antara keamanan dan efisiensi, dengan Blake2 menekankan pada kecepatan dan kemampuan menghasilkan nilai *hash* variabel. Pemahaman terhadap properti ini penting untuk memilih fungsi *hash* yang tepat sesuai kebutuhan keamanan dan performa aplikasi.

Dalam Tabel 4 juga dicantumkan tingkat keamanan fungsi *hash* SHA-3 terhadap serangan yang paling dikenal. Istilah "*collision security*" mengacu pada kompleksitas untuk menemukan *collision*, yaitu kondisi ketika dua pesan

berbeda menghasilkan nilai *hash* yang sama. Jika *collision security* bernilai t , ini berarti serangan untuk menemukan *collision* memerlukan kira-kira 2^t langkah. Sementara itu, istilah “*preimage security*” memiliki arti yang serupa, tetapi mencakup serangan untuk menemukan *preimage* (pesan asli dari fungsi *hash* yang diketahui) maupun *second preimage* (pesan kedua yang berbeda tetapi menghasilkan nilai *hash* sama dengan pesan pertama).

Tabel 5. Parameter dan Tingkat Keamanan SHA-3.

Fungsi Hash	Ukuran State (b)	Rate (r)	Capacity (c)	Collision Security	Preimage Security
SHA3-256	1600	1088	512	128	256
SHA3-384	1600	832	768	192	384
SHA3-512	1600	576	1024	256	512
SHAKE256	1600	1088	512	$\min\{d/2, 256\}$	$\min\{d, 256\}$

Tabel 5 menunjukkan parameter dan tingkat keamanan beberapa varian SHA-3. Semua fungsi *hash* menggunakan *state* 1600 bit, dibagi menjadi *rate* (r) untuk jumlah bit input per iterasi, dan *capacity* (c) yang menentukan keamanan.

- SHA3-256: nilai *hash* 256 bit, *collision security* 128 bit, *preimage security* 256 bit.
- SHA3-384: nilai *hash* 384 bit, *collision security* 192 bit, *preimage security* 384 bit.
- SHA3-512: nilai *hash* 512 bit, *collision security* 256 bit, *preimage security* 512 bit.
- SHAKE256: *output* variabel d , *collision security* minimum $d/2$ atau 256 bit, *preimage security* minimum d atau 256 bit.

Dalam algoritma SHA-3, *rate* dan *capacity* adalah parameter utama yang menentukan efisiensi dan keamanan. *Rate* yang lebih tinggi memungkinkan pemrosesan lebih banyak bit per iterasi, sehingga proses perhitungan nilai *hash* menjadi lebih cepat, sedangkan *capacity* yang lebih besar meningkatkan ketahanan terhadap serangan *preimage* dan *collision*. Dengan kombinasi ini, SHA-3 mampu menyeimbangkan kecepatan dan keamanan. Selain itu,

struktur *sponge* membuat SHA-3 tahan terhadap serangan *length extension*, dan varian SHAKE memberikan fleksibilitas *output* untuk kebutuhan yang berbeda.

VI. PEMBANGKIT BIT ACAK DETERMINISTIK (*DETERMINISTIC RANDOM BIT GENERATORS*)

Deterministic random bit generators (DRBG) merupakan mekanisme yang menghasilkan rangkaian bit yang tampak acak namun bersifat deterministik. Bit-bit *output* ini digunakan dalam berbagai primitif kriptografi, baik secara langsung maupun setelah dikonversi ke bentuk lain seperti *integer*.

Disebut deterministik karena DRBG akan selalu menghasilkan *output* identik apabila diberikan *input* awal (*seed*) yang sama. Proses pembangkitan bit didasarkan pada algoritma tertentu yang menggunakan nilai *seed* sebagai sumber keacakan awal [1]. Agar bit *output* tidak dapat diprediksi, *seed* harus diperoleh dari sumber dengan entropi tinggi dan dijaga kerahasiaannya. Selama algoritma DRBG aman dan *seed* terlindungi, *output* bit tetap tidak mudah ditebak dalam batas kekuatan keamanan yang berlaku.

Operasi DRBG dimulai dengan proses instansiasi, yaitu pembentukan *internal state* awal menggunakan *seed* dan parameter terkait. Untuk menjaga tingkat keamanan, DRBG dapat melakukan *reseed*, yakni pembaruan *seed* dengan entropi baru secara berkala. Jangka waktu penggunaan setiap *seed* dibatasi oleh *seed period*, yang menentukan kapan pembaruan harus dilakukan. Selain *seed*, DRBG juga dapat menerima input tambahan seperti *timestamp* atau *nonce*. Walaupun tidak bersifat rahasia, input ini membantu memperkaya keacakan internal. Setelah diinstansiasi, DRBG menghasilkan bit-bit *pseudorandom*, bukan acak sejati, karena sifat deterministik dari algoritmanya.

Menurut NIST SP 800-90A Rev. 1 [1], mekanisme DRBG mengelola *internal state* sebagaimana diilustrasikan pada Gambar 10 melalui lima fungsi utama berikut:

1. Fungsi instansiasi (*instantiate function*)

Fungsi ini bertugas membentuk *internal state* awal dengan menggabungkan input entropi, *nonce*, dan *personalization string* menjadi sebuah *seed*. Hasilnya menjadi dasar bagi operasi DRBG berikutnya.

2. Fungsi pembangkitan (*generate function*)

Fungsi ini membangkitkan bit-bit *pseudorandom* sesuai permintaan berdasarkan *internal state* saat ini. Jika tersedia, *additional input* dapat ditambahkan untuk memperkaya proses pembangkitan. Setelahnya, *internal state* diperbarui untuk menjaga keamanan *output* berikutnya.

3. Fungsi *reseed* (*reseed function*)

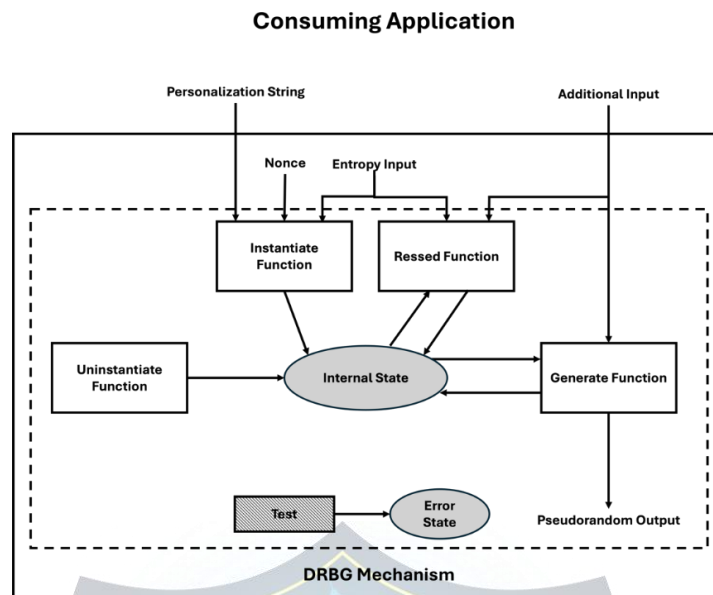
Fungsi ini memperbarui *internal state* dengan menggabungkan entropi baru, *internal state* sebelumnya, serta *additional input*. Proses ini menghasilkan *seed* baru yang memperkuat ketahanan terhadap prediksi *output*.

4. Fungsi *uninstantiate* (*uninstantiate function*)

Fungsi ini berfungsi untuk menghapus seluruh *internal state* DRBG melalui proses *zeroization*, sehingga DRBG tidak lagi dapat digunakan hingga dilakukan instansiasi ulang.

5. Fungsi pengujian kesehatan (*health test function*)

Fungsi ini memastikan DRBG beroperasi secara normal. Pengujian dilakukan untuk mendeteksi kondisi kegagalan atau anomali yang dapat memengaruhi keamanan *output*.



Gambar 10. Mekanisme DRBG digambar ulang dari [1]

Pengguna disarankan merujuk pada NIST SP 800-90A Rev. 1 [1] atau ISO/IEC 18031 [30] sebagai referensi komprehensif dalam penerapan DRBG secara aman dan sesuai standar internasional. Dalam NIST SP 800-90A Rev. 1 [1], mekanisme DRBG diklasifikasikan berdasarkan primitif kriptografi yang digunakan, meliputi Hash_DRBG (berbasis fungsi *hash*), HMAC_DRBG (berbasis fungsi HMAC), dan CTR_DRBG (berbasis *block cipher*). Pemilihan jenis DRBG yang tepat perlu mempertimbangkan kekuatan keamanan dari primitif yang digunakan serta ketersediaan dan dukungannya dalam lingkungan sistem tempat DRBG akan diimplementasikan.

A. Pembangkit Bit Acak Deterministik berbasis Fungsi *Hash* (*Deterministic Random Bit Generators Based On Hash Function*)

Mekanisme DRBG berbasis fungsi *hash*, selanjutnya disebut *Hash-DRBG*, dirancang agar kompatibel dengan fungsi *hash* yang telah disetujui (*approved hash functions*) dan tercantum dalam Algoritma AKI. Seluruh proses utama dalam *Hash-DRBG* yaitu *instantiation*, *reseeding*, dan *generate* bergantung pada fungsi *hash* yang digunakan. Selama satu siklus instansiasi, *Hash-DRBG* harus menggunakan fungsi *hash* yang sama secara konsisten.

1. Algoritma

Sebanyak sebelas algoritma *Hash-DRBG* direkomendasikan sebagai bagian dari Algoritma AKI. Algoritma ini tercantum dalam dua standar internasional, yaitu:

- NIST SP 800-90A Revisi 1 *Recommendation for Random Number Generation Using Deterministic Random Bit Generators* [1].
- ISO/IEC 18031:2025 *Information technology — Security techniques — Random bit generation* [30].
- SNI ISO/IEC 18031:2025 *Teknologi informasi — Teknik keamanan — Pembangkitan bit random* [31].

Seluruh algoritma *Hash-DRBG* tersebut tetap dinyatakan akseptabel hingga tahun 2030. Daftar lengkap beserta sifatnya dapat ditemukan pada Tabel 6.

Tabel 6. Daftar Algoritma *Hash_DRBG* dalam Algoritma AKI.

No	Hash DRBG	Kekuatan Keamanan	Output Block Length (outlen)	Seed Length (seedlen)	Min Entropy Input Length
1	SHA1-DRBG	128	160	440	128
2	SHA224-DRBG	192	224	440	192
3	SHA512/224-DRBG	192	224	440	192
4	SHA512/256-DRBG	192	256	440	192
5	SHA256-DRBG	≥ 256	256	440	256
6	SHA384-DRBG	≥ 256	384	888	256
7	SHA512-DRBG	≥ 256	512	888	256
8	SHA3-224-DRBG	192	224	440	192
9	SHA3-256-DRBG	≥ 256	256	440	256
10	SHA3-384-DRBG	≥ 256	384	888	256
11	SHA3-512-DRBG	≥ 256	512	888	256

2. Penggunaan

Dalam penerapan algoritma *Hash-DRBG*, pengguna perlu memperhatikan aspek teknis dan keamanan berikut:

- a. Fungsi *hash* yang digunakan harus memiliki kekuatan keamanan minimal setara dengan kebutuhan keamanan aplikasi yang menggunakan *output* DRBG.
- b. Mekanisme *Hash-DRBG* dapat digunakan pada berbagai tingkat kebutuhan keamanan, selama entropi awal (*seed*) memadai dan fungsi *hash* yang digunakan sesuai dengan tingkat keamanan yang diinginkan.

Perlu diperhatikan bahwa algoritma SHA-224, SHA-512/224, dan SHA-512/256 tidak memberikan keuntungan efisiensi dibandingkan algoritma dasar yang menjadi basisnya, seperti SHA-256 atau SHA-512. Hal yang sama berlaku untuk SHA-384, yang berbasis pada SHA-512. Oleh karena itu, pertimbangan efisiensi, kompatibilitas, dan kebutuhan keamanan harus digunakan secara bersamaan dalam pemilihan fungsi *hash*.

Menurut NIST SP 800-131A Rev. 3, yang mengatur transisi penggunaan algoritma kriptografi dengan mempertimbangkan ancaman komputasi kuantum, penggunaan fungsi *hash* SHA-1 dan fungsi *hash* 224 bit (termasuk SHA-224, SHA-512/224, dan SHA3-224) telah berstatus *deprecated* karena tidak lagi memenuhi standar keamanan yang direkomendasikan. Penggunaan fungsi *hash* tersebut hanya diperbolehkan hingga 31 Desember 2030, dan dinyatakan *disallowed* setelah tanggal tersebut [32].

Sebaliknya, penggunaan *Hash-DRBG* dengan fungsi *hash* SHA-2 dan SHA-3 yang lebih kuat, seperti SHA-384, SHA-512/256, SHA3-256, SHA3-384, dan SHA3-512, tetap dinyatakan akseptabel sesuai dengan daftar Algoritma AKI. Meskipun SHA-256 dan SHA-512 tidak tercantum secara eksplisit dalam daftar tersebut, keduanya masih dapat digunakan dalam mekanisme *Hash-DRBG* karena keamanan keseluruhan DRBG tidak hanya bergantung pada fungsi *hash*, tetapi juga pada parameter konfigurasi, entropi input, dan praktik implementasi yang benar.

B. Pembangkit Bit Acak Deterministik Kode Autentikasi Pesan *Hash* (*Hash Message Authentication Code Deterministic Random Bit Generators*)

Mekanisme DRBG yang menggunakan HMAC, selanjutnya disebut HMAC-DRBG, dibangun berdasarkan *approved hash functions* dan tercantum dalam Algoritma AKI. Dalam mekanisme ini, fungsi HMAC digunakan di dalam prosedur HMAC-DRBG_Update sebagai fungsi derivasi selama proses instansiasi dan *reseeding*. Fungsi *hash* yang dipilih harus digunakan secara konsisten sepanjang satu siklus instansiasi HMAC-DRBG agar menjamin konsistensi *output* dan keamanan sistem.

Mekanisme HMAC-DRBG terdiri atas tiga tahap utama, yaitu instansiasi, *generate*, dan *reseed*. Pada setiap tahap tersebut, fungsi HMAC-DRBG_Update berperan dalam memperbarui *internal state*, baik ketika entropi baru atau *additional input* diberikan, maupun setelah proses pembangkitan bit *pseudorandom* dilakukan.

1. Algoritma

Terdapat 11 (sebelas) buah algoritma HMAC-DRBG yang direkomendasikan sebagai bagian Algoritma AKI. Algoritma HMAC-DRBG dinyatakan dalam standar internasional berikut:

- a. NIST SP 800-90A Revisi 1 *Recommendation for Random Number Generation Using Deterministic Random Bit Generators* [1].
- b. ISO/IEC 18031:2025 *Information technology — Security techniques — Random bit generation* [30].
- c. SNI ISO/IEC 18031:2025 *Teknologi informasi — Teknik keamanan — Pembangkitan bit random* [31].

Algoritma HMAC-DRBG tersebut masih berstatus akseptabel sampai tahun 2030. Kesebelas algoritma HMAC-DRBG beserta propertinya dapat dilihat dalam Tabel 7.

Tabel 7. Daftar Algoritma HMAC-DRBG dalam Algoritma AKI.

No	HMAC_DRBG	Kekuatan Keamanan	Output Block Length (outlen)	Seed Length (seedlen)	Min Entropy Input Length
1	HMAC-SHA1-DRBG	128	224	440	128
2	HMAC-SHA224_DRBG	192	224	440	192
3	HMAC-SHA256-DRBG	≥ 256	256	440	≥ 256
4	HMAC-SHA384-DRBG	≥ 256	385	888	≥ 256
5	HMAC-SHA512-DRBG	≥ 256	512	888	≥ 256
6	HMAC-SHA-512/224-DRBG	192	224	440	192
7	HMAC-SHA512/256-DRBG	≥ 256	256	440	≥ 256
8	HMAC--SHA3-224-DRBG	192	224	440	192
9	HMAC-SHA3-256-DRBG	≥ 256	256	440	≥ 256
10	HMAC-SHA3-384-DRBG	≥ 256	384	888	≥ 256
11	HMAC-SHA3-512-DRBG	≥ 256	512	888	≥ 256

2. Penggunaan dan Pertimbangan Teknis

Dalam penerapan algoritma HMAC-DRBG, pengguna perlu memastikan aspek teknis berikut:

a. Dasar keamanan.

HMAC-DRBG berasumsi bahwa fungsi hash yang digunakan, sebagaimana direkomendasikan dalam Algoritma AKI, merupakan bagian dari famili fungsi *pseudorandom*. Oleh karena itu, selama kunci HMAC tidak diketahui penyerang, *output* HMAC tetap tampak acak meskipun input dapat diketahui atau dikontrol. Bahkan fungsi *hash* yang relatif lemah menjadi lebih kuat ketika digunakan dalam konstruksi HMAC. Dengan demikian, keamanan HMAC-DRBG sangat bergantung pada sifat *pseudorandomness* dari fungsi *hash* yang mendasarinya.

- b. Kekuatan keamanan
Fungsi *hash* yang digunakan dalam HMAC-DRBG harus memiliki tingkat keamanan yang sama atau lebih tinggi daripada kebutuhan keamanan aplikasi yang memanfaatkan *output* DRBG.
- c. Kinerja
HMAC-DRBG menghasilkan *output pseudorandom* dengan kecepatan yang lebih rendah dibandingkan fungsi *hash* dasarnya. Misalnya, untuk SHA-256, proses pembangkitan bit acak berjalan sekitar seperempat dari kecepatan pemrosesan input SHA-256, dengan tambahan *overhead* sekitar 2048 bit per permintaan pembangkitan. Namun, karena fungsi *hash* umumnya cepat, hal ini tidak menimbulkan dampak signifikan bagi sebagian besar aplikasi.
- d. Masa Berlaku dan Transisi Keamanan
Berdasarkan NIST SP 800-131A Rev. 3 [32], penggunaan fungsi *hash* SHA-1 atau varian 224 bit (SHA-224, SHA-512/224, SHA3-224) masih dinyatakan *akseptabel* hingga 31 Desember 2030, namun dikategorikan *deprecated* dan akan *disallowed* setelahnya. Penggunaan HMAC-DRBG dengan fungsi *hash* dari keluarga SHA-2 dan SHA-3 lain, seperti SHA-384, SHA-512/256, SHA3-256, SHA3-384, dan SHA3-512, tetap dianggap aman dan direkomendasikan dalam Algoritma AKI.
- e. Fleksibilitas Implementasi
Meskipun tidak tercantum langsung dalam Algoritma AKI, fungsi *hash* SHA-256 dan SHA-512 tetap dapat digunakan sebagai komponen dalam HMAC-DRBG karena keduanya termasuk dalam keluarga SHA-2 dan memberikan tingkat keamanan yang memadai.

C. Pembangkit Bit Acak Deterministik Berbasis Sandi Blok (*Deterministic Random Bit Generators Based On Block Cipher*)

Mekanisme berbasis *block cipher* (untuk selanjutnya dinyatakan sebagai CTR-DRBG) merujuk pada penggunaan *Deterministic Random Bit Generator* yang menggunakan algoritma *block cipher*. Mekanisme CTR-DRBG yang dijelaskan dalam panduan ini dirancang agar kompatibel dengan *approved block cipher* (*block cipher* yang disetujui) dan tercantum dalam Algoritma AKI.

1. Algoritma

Terdapat 3 (tiga) buah algoritma CTR-DRBG yang direkomendasikan sebagai bagian dari Algoritma AKI. Algoritma CTR-DRBG dinyatakan dalam standar internasional sebagai berikut:

- NIST SP 800-90A Rev. 1 *Recommendation for Random Number Generation Using Deterministic Random Bit Generators* [1].
- ISO/IEC 18031:2025 *Information technology — Security techniques — Random bit generation* [30].
- SNI ISO IEC 18031:2025 *Teknologi informasi – Teknik keamanan – Pembangkit bit random (ISO/IEC 18031:2025, IDT, Eng)* [31].

Algoritma CTR-DRBG tersebut masih berstatus akseptabel sampai tahun 2030. Ketiga algoritma CTR-DRBG beserta sifatnya dapat dilihat dalam Tabel 8.

Tabel 8. Daftar Algoritma CTR_DRBG dalam Algoritma AKI.

Kriteria	AES 128-CTR-DRBG	AES 192-CTR-DRBG	AES 256-CTR-DRBG
Kekuatan Keamanan (<i>security strength</i>)	128	192	256
Panjang blok <i>input</i> dan <i>output</i> (<i>blocklen</i>)	128		
<i>Counter field length</i> (<i>ctr_len</i>)	$4 \leq ctr_len \leq blocklen$		
Entropi minimum yang dibutuhkan untuk <i>instantiate</i> dan <i>reseed</i>	128	192	256
Panjang <i>seed</i> (<i>seedlen</i>)	256	320	384
Jika fungsi derivasi digunakan, <i>Minimum entropy input length</i> (<i>min_length</i>)	128	192	256
Jika fungsi derivasi digunakan, <i>Maximum entropy input length</i> (<i>max_length</i>)	2^{35}		
Jika fungsi derivasi tidak digunakan, <i>Minimum entropy input length</i> (<i>min_length = blocklen + keylen</i>)	256	320	384

Kriteria	AES 128-CTR-DRBG	AES 192-CTR-DRBG	AES 256-CTR-DRBG
Jika fungsi derivasi tidak digunakan, <i>Maximum entropy input length</i> ($max_length = blocklen + keylen$)	256	320	384
<i>max_number_of_bits_per_request</i> (for $B = (2^{ctr_len} - 4) \times blocklen$)	$min(B, 2^{19})$		
<i>Maximum number of requests between reseeds</i> (<i>reseed_interval</i>)	2^{48}		

2. Penggunaan

Dalam penerapan algoritma CTR-DRBG, pengguna harus memastikan aspek-aspek teknis dan keamanan berikut:

- Menggunakan algoritma *block cipher* yang telah disetujui dan tercantum dalam Algoritma AKI.
- Memastikan panjang kunci dan algoritma yang digunakan dalam seluruh proses operasi DRBG tetap konsisten, mulai dari instansiasi, *reseed*, hingga *generate*. Algoritma *block cipher* dan panjang kunci yang dipilih harus memenuhi atau melebihi tingkat kekuatan keamanan yang dibutuhkan oleh aplikasi pengguna.
- CTR-DRBG dapat digunakan oleh aplikasi dengan kebutuhan tingkat keamanan yang berbeda, selama algoritma *block cipher* dan panjang kunci yang digunakan sesuai dengan tingkat keamanan yang dibutuhkan dan entropi untuk pembentukan *seed* tersedia dalam jumlah yang mencukupi.
- Menentukan parameter internal secara tepat, yaitu panjang blok (*blocklen*) sebesar 128 bit dan panjang *counter* (*ctr_len*) antara 4 hingga 128 bit, disesuaikan dengan konfigurasi sistem.
- Menyediakan input entropi saat proses instansiasi dengan panjang minimal yang sesuai dengan kekuatan keamanan yang diklaim, sebagai contoh sekurang-kurangnya 128 bit input entropi diperlukan untuk tingkat keamanan 128 bit.
- Menggunakan fungsi derivasi jika sumber entropi tidak menjamin *output* dengan *full entropy* (entropi penuh). Dalam hal ini, panjang

maksimum untuk *input entropy*, *personalization string*, dan *additional input* masing-masing dibatasi hingga 2^{35} bit.

- g. Tidak menggunakan fungsi derivasi apabila sumber entropi telah memenuhi kriteria entropi penuh. Dalam skenario ini, panjang *input entropy*, *personalization string*, dan *additional input* harus sama dengan panjang *seed*, yang bervariasi bergantung pada panjang kunci AES yaitu 256 bit untuk AES-128, 320 bit untuk AES-192, dan 384 bit untuk AES-256.
- h. Pembangkitan bit *pseudorandom* dalam satu permintaan fungsi *generate()* tidak boleh melebihi batas maksimum, yaitu nilai $\min(B, 2^{13})$ dengan $B = (2^{ctr_len} - 4) \times 128$.
- i. Melakukan proses *reseeding* secara berkala untuk menjaga keamanan. Jumlah maksimum permintaan fungsi *generate()* yang diperbolehkan sebelum dilakukan *reseeding* adalah sebanyak 2^{48} kali.
- j. Memastikan bahwa setiap proses fungsi *generate()* diikuti oleh pemutakhiran status internal menggunakan fungsi CTR_DRBG_Update, untuk menjaga kerahasiaan internal state dan mencegah serangan *backtracking*.
- k. Mendukung pengaktifan *prediction resistance* apabila diperlukan. Dalam mode ini, setiap permintaan fungsi *generate()* harus selalu didahului oleh proses *reseeding* dengan input entropi baru.
- l. Menyusun status internal CTR-DRBG secara tepat, yang terdiri atas nilai *counter* (*V*) sepanjang 128 bit, nilai kunci internal (*Key*) sesuai panjang AES yang digunakan, dan penghitung *reseed* (*reseed_counter*) yang diinisialisasi dengan nilai 1 dan bertambah setiap kali *generate* atau *reseed* dijalankan.
- m. Menginisiasi CTR-DRBG dengan membentuk *seed_material* dari *entropy input*, *personalization string*, dan parameter *security strength* (serta *nonce* jika menggunakan fungsi derivasi dalam inisiasi CTR-DRBG). *Seed_material* ini selanjutnya diproses dengan menggunakan ataupun tanpa menggunakan fungsi derivasi, untuk

menghasilkan nilai awal V , Key , dan $reseed_counter$ yang diperlukan dalam CTR-DRBG *Internal State*.

- n. Mengimplementasikan seluruh prosedur operasi secara lengkap dan benar, sesuai dengan spesifikasi [1] yang mencakup lima fungsi utama: *Instantiate()*, *Reseed()*, *Generate()*, *CTR_DRBG_Update()*, dan *Uninstantiate()*.

VII. ALGORITMA ASIMETRIK

Algoritma asimetrik merupakan jenis algoritma kriptografi yang menggunakan sepasang kunci berbeda yaitu *public key* (kunci publik) untuk proses enkripsi atau verifikasi, dan *private key* (kunci privat) untuk proses dekripsi atau penandatanganan. Pendekatan ini memungkinkan komunikasi yang aman tanpa memerlukan pertukaran kunci rahasia secara langsung.

Secara umum, algoritma asimetrik terbagi dalam tiga kategori utama, yaitu:

1. Primitif asimetrik, yaitu komponen dasar yang menjadi dasar konstruksi algoritma asimetrik, seperti fungsi *trapdoor* dan masalah matematika sulit (*hard problem*).
2. Skema enkripsi asimetrik (*asymmetric encryption scheme*), yang dirancang untuk menjamin kerahasiaan pesan melalui proses enkripsi menggunakan kunci publik dan dekripsi menggunakan kunci privat.
3. Skema tanda tangan digital asimetrik (*asymmetric digital signature scheme*), yang digunakan untuk menjamin integritas dan autentikasi data dengan cara membubuhkan tanda tangan digital menggunakan kunci privat dan memverifikasinya menggunakan kunci publik.

A. Primitif Asimetrik

Primitif asimetrik merupakan algoritma dasar yang digunakan untuk menghasilkan pasangan kunci publik dan kunci privat. Pasangan kunci ini kemudian digunakan dalam berbagai skema kriptografi, termasuk skema enkripsi dan tanda tangan digital. Primitif asimetrik diklasifikasikan ke dalam tiga kelompok utama berdasarkan *hard problem* yang menjadi dasar keamanannya, yaitu:

1. Algoritma berbasis *Integer Factorization Problem* (IFP)
Mengandalkan kesulitan faktorisasi integer besar menjadi faktor-faktor prima. Contoh paling dikenal adalah algoritma RSA.
2. Algoritma berbasis *Discrete Logarithm Problem* (DLP)
Berdasarkan kesulitan menghitung logaritma diskrit dalam grup bilangan modulus prima. Contoh algoritma dalam kategori ini adalah ElGamal dan DSA.
3. Algoritma berbasis *Elliptic Curve Discrete Logarithm Problem* (ECDLP)
Memanfaatkan kesulitan menghitung logaritma diskrit pada grup titik-titik dalam kurva eliptis. Algoritma seperti ECDSA dan ECDH termasuk dalam kelompok ini, yang secara umum menawarkan tingkat keamanan tinggi dengan ukuran kunci yang lebih kecil dibanding algoritma berbasis IFP atau DLP.

Berikut adalah penjelasan detail dari tiga kelompok primitif asimetrik:

1. Problem faktorisasi integer

a. Algoritma

Algoritma yang berbasis pada masalah faktorisasi integer terdiri atas satu jenis, yaitu RSA dengan ukuran kunci minimum sebesar 3072 bit. Spesifikasi teknis RSA-3072 dijabarkan dalam standar berikut:

- 1) SNI ISO/IEC 18033-2:2016 — Teknologi informasi — Teknik keamanan — Algoritma enkripsi — Bagian 2: Penyandian asimetris (ISO/IEC 18033-2:2006 dan ISO/IEC 18033-2/Amd 1:2017, IDT, Eng) [33];
- 2) ISO/IEC 18033-2: 2006 — *Information technology — Security techniques — Encryption algorithms Part 2: Asymmetric ciphers* [34];
- 3) NIST SP 800-57 *Part 1 Rev. 5 Recommendation for Key Management: Part 1 – General* [35];
- 4) IETF RFC 8017 : PKCS #1: *RSA Cryptography Specifications Version 2.2* [36];
- 5) FIPS 186-5 *Digital Signature Standard (DSS)* [7].

b. Penggunaan

Dalam implementasi algoritma RSA, perhatian utama terletak pada pembangkitan kunci publik dan kunci privat. Berdasarkan Lampiran A.1 FIPS 186-5 [7], proses pembangkitan pasangan kunci dijelaskan secara terperinci. Pasangan kunci pada algoritma dengan problem faktorisasi integer, seperti RSA, terdiri atas kunci publik (n, e) dan kunci privat (n, d) , dengan n merupakan modulus hasil perkalian dari dua bilangan prima, yaitu p dan q , e merupakan eksponen publik dan d merupakan eksponen privat.

Tingkat keamanan RSA sangat bergantung pada kerahasiaan bilangan prima p dan q serta eksponen privat d . Proses pembangkitan p dan q harus mengikuti salah satu metode berikut:

- 1) Membangkitkan p dan q yang merupakan bilangan prima secara acak (*random primes*), dan harus memenuhi salah satu kriteria berikut:
 - a) Bilangan terbukti prima (*provable primes*) sebagaimana Lampiran A.1.2 FIPS 186-5 [7]; atau
 - b) Bilangan yang kemungkinan prima (*probable primes*) sebagaimana Lampiran A.1.3 FIPS 186-5 [7].

Kedua metode tersebut menghasilkan p dan q dengan panjang masing-masing sekitar setengah dari panjang modulus n .

- 2) Membangkitkan p dan q yang merupakan bilangan prima acak secara acak serta memenuhi ketentuan tambahan berikut (*primes with conditions*):
 - a) $(p - 1)$ memiliki faktor prima p_1
 - b) $(p + 1)$ memiliki faktor prima p_2
 - c) $(q - 1)$ memiliki faktor prima q_1
 - d) $(q + 1)$ memiliki faktor prima q_2

Faktor-faktor p_1, p_2, q_1 , dan q_2 ini dikenal sebagai bilangan prima bantu (*auxiliary primes*) yang dapat digunakan untuk pengujian keamanan tambahan atau penguatan struktur bilangan prima dalam implementasi RSA.

Dengan menggunakan metode 2), salah satu dari kondisi berikut akan berlaku:

- 1) Bilangan prima p, q, p_1, p_2, q_1 , dan q_2 harus *provable primes* sesuai dengan metode pada Lampiran A.1.4 FIPS 186-5 [7];
- 2) Bilangan prima p_1, p_2, q_1 , dan q_2 harus *provable primes*, dan bilangan prima p dan q harus *probable primes* sesuai metode pada Lampiran A.1.5 FIPS 186-5 [7];
- 3) Bilangan prima p, q, p_1, p_2, q_1 , dan q_2 harus *probable primes* sesuai metode pada Lampiran A.1.6 FIPS 186-5 [7].

Panjang minimum setiap *auxiliary primes* p_1, p_2, q_1 , dan q_2 ditentukan berdasarkan $nlen$, yaitu panjang dari modulus n dalam satuan bit yang merepresentasikan ukuran kunci. Panjang dari *auxiliary primes* ini dapat ditentukan secara tetap maupun dipilih secara acak, dengan tetap mengacu pada batasan yang tercantum dalam Tabel 9. Panjang maksimum *auxiliary primes* ditentukan dengan mempertimbangkan dua faktor utama, yaitu:

- 1) nilai $nlen$ yang digunakan; dan
- 2) jenis bilangan prima p dan q termasuk ke dalam *probable primes* atau *provable primes*.

Tabel 9. Panjang minimum dan maksimum *auxiliary primes* p_1, p_2, q_1 , dan q_2 .

$nlen$ (bit)	Panjang minimum <i>auxiliary primes</i> p_1, p_2, q_1 , dan q_2	Panjang maksimum dari $len(p_1) + len(p_2)$ dan $len(q_1) + len(q_2)$	
		p, q <i>probable primes</i>	p, q <i>provable primes</i>
$2048 \leq nlen \leq 3071$	> 140 bit	≤ 1007 bit	≤ 494 bit
$3072 \leq nlen \leq 4095$	> 170 bit	≤ 1518 bit	≤ 750 bit
$4096 \leq nlen$	> 200 bit	≤ 2030 bit	≤ 1005 bit

Sesuai dengan kriteria khusus Algoritma AKI, panjang $nlen$ yang direkomendasikan adalah **minimal 3072**.

Selain itu, seluruh kunci RSA juga harus memenuhi kriteria berikut:

- 1) Eksponen publik e harus dipilih dengan ketentuan sebagai berikut:
 - a) Eksponen publik e harus dipilih sebelum membangkitkan bilangan prima p dan q , serta eksponen privat d .
 - b) Eksponen e harus berupa bilangan integer positif ganjil yang memenuhi:

$$2^{16} < e < 2^{256}$$

Nilai e dapat berupa nilai tetap atau nilai acak selama memenuhi ketentuan 1) b).

- 2) Bilangan prima p dan q harus dipilih dengan ketentuan sebagai berikut:
 - a) $(p - 1)$ dan $(q - 1)$ harus relatif prima terhadap eksponen publik e .
 - b) Faktor prima privat p harus dipilih secara acak dan memenuhi:
$$(\sqrt{2})(2^{(nlen/2)-1}) \leq p \leq (2^{(nlen/2)} - 1)$$
dengan $nlen$ adalah panjang yang sesuai dengan tingkat keamanan yang diinginkan.
 - c) Faktor prima privat q harus dipilih secara acak dan memenuhi:
$$(\sqrt{2})(2^{(nlen/2)-1}) \leq q \leq (2^{(nlen/2)} - 1)$$
dengan definisi $nlen$ yang sama seperti di atas.
 - d) $|p - q| > 2^{(nlen/2)-100}$
- 3) Eksponen tanda tangan privat d harus dipilih dengan ketentuan sebagai berikut (setelah p dan q dibangkitkan):
 - a) Eksponen d harus berupa bilangan integer positif yang memenuhi:

$$2^{nlen/2} < d < \text{LCM}(p - 1, q - 1)$$

- b) $d = e^{-1} \bmod (\text{LCM}(p - 1, q - 1))$.

Pertidaksamaan pada 3)a) dan $1 \equiv (ed) \bmod (\text{LCM}(p - 1, q - 1))$ harus terpenuhi.

Pada kondisi $d \leq 2^{n_{len}/2}$ (meskipun sangat jarang terjadi) nilai baru untuk p, q dan d harus ditentukan kembali. Nilai e yang berbeda dapat digunakan, meskipun hal ini tidak diwajibkan.

Ketentuan pemilihan parameter p, q, e , dan d harus diikuti secara ketat untuk mencegah berbagai serangan kriptanalisis yang memanfaatkan kelemahan dalam pemilihan parameter. Tabel 10 merangkum jenis serangan yang relevan, mekanisme kelemahan kunci (*weak key*), dan penjelasan singkatnya.

Tabel 10. Jenis Serangan dan Mekanisme *Weak Key* pada RSA

Jenis Serangan	Mekanisme <i>Weak Key</i>	Penjelasan Singkat
Faktor prima lemah (<i>small primes/too close primes</i>)	p dan q terlalu kecil atau terlalu dekat nilainya	Jika p dan q dipilih terlalu dekat, <i>Fermat Factorization</i> dapat digunakan untuk memfaktorkan n . Jika p atau q kecil, serangan <i>brute force</i> menjadi fisibel. ¹
<i>Shared prime factors (common factor attack)</i>	Dua kunci RSA berbeda berbagi salah satu faktor prima	Jika dua modulus (n_1 dan n_2) memiliki faktor prima yang sama (misalnya p), penyerang dapat menghitung $GCD(n_1, n_2)$ untuk memecahkan keduanya sekaligus. Serangan ini pernah ditemukan pada jutaan kunci RSA di Internet akibat pembangkit bilangan acak yang buruk ²
<i>Small Public Exponent (e = 3 Attack)</i>	e terlalu kecil, misalnya $e = 3$ tanpa <i>padding</i> yang aman	Jika pesan dienkripsi tanpa <i>padding</i> (misalnya <i>textbook RSA</i>), <i>ciphertext</i> dapat dipangkatkan balik untuk menemukan <i>plaintext</i> .

¹ <https://arstechnica.com/information-technology/2022/03/researcher-uses-600-year-old-algorithm-to-crack-crypto-keys-found-in-the-wild/>

² <https://github.com/trevorlinton/rsa-common-factors-gcd-attack>

Jenis Serangan	Mekanisme Weak Key	Penjelasan Singkat
		Contoh: <i>Hastad's Broadcast Attack</i> . ³
<i>Small private exponent (Wiener's attack)</i>	d terlalu kecil	Jika eksponen privat $d < n^{0,25}$, maka <i>Wiener's attack</i> dapat menemukan kunci privat dengan <i>continued fractions</i> . ⁴
<i>Non-random/predictable primes</i>	p dan q dibangkitkan dengan PRNG yang lemah	Jika pembangkit bilangan acak lemah (prediktif), penyerang dapat menebak atau melakukan serangan <i>brute force</i> kandidat p dan q . Kasus nyata pernah terjadi pada perangkat <i>embedded</i> dan <i>smartcard</i> . ⁵
<i>Reuse public modulus (key reuse attack)</i>	n digunakan ulang untuk kunci berbeda (dengan e berbeda)	Jika dua pihak berbagi modulus n tetapi menggunakan eksponen e_1, e_2 berbeda, penyerang dapat menggunakan <i>Extended Euclidean Algorithm</i> (EEA) untuk menghitung kembali <i>plaintext</i> . ^{6 7 8}
<i>Partial key exposure</i>	Sebagian bit d atau faktor prima diketahui	Serangan matematis (<i>Boneh–Durfee attack</i> atau <i>lattice-based attacks</i>) dapat

³ <https://cic.iacr.org/p/1/3/29>

⁴ <https://github.com/pablocelayes/rsa-wiener-attack>

⁵ <https://www.keyfactor.com/blog/the-irony-and-dangers-of-predictable-randomness/>

⁶ <https://www.utc.edu/sites/default/files/2021-04/course-paper-5600-rsa.pdf>

⁷ <https://eprint.iacr.org/2024/1903.pdf>

⁸ <https://eprint.iacr.org/2024/1903.pdf>

Jenis Serangan	Mekanisme Weak Key	Penjelasan Singkat
		menemukan d jika cukup banyak bit diketahui. ⁹
<i>Fault injection/ side channel</i>	Kunci dihasilkan dengan kesalahan (misalnya p atau q tidak prima)	Jika perangkat terkena <i>fault attack</i> , bisa dihasilkan modulus komposit yang mudah difaktorkan. Selain itu, serangan <i>side-channel (timing, power analysis)</i> juga dapat membocorkan bit d . ¹⁰

2. Problem logaritma diskrit

a. Algoritma

Salah satu algoritma yang berbasis pada *Discrete Logarithm Problem* adalah Diffie-Hellman, yang memerlukan ukuran kunci minimal $l(p^n) \geq 3072$, dan $l(p), l(q) \geq 256$. Standar teknis yang mendefinisikan algoritma Diffie-Hellman tercantum dalam dokumen berikut:

- 1) NIST SP 800-56A Rev.3 *Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography* [37];
- 2) IETF RFC 3526: *More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)* [38];
- 3) NIST SP 800-57 Part 1 Rev. 5 *Recommendation for Key Management: Part 1 – General* [35];

b. Penggunaan

Salah satu implementasi algoritma Diffie-Hellman terdapat pada *Digital Signature Algorithm* (DSA). Parameter - parameter Diffie-Hellman yang digunakan dalam DSA mengacu pada FIPS 186-4 [39] karena tidak ditemukan dalam FIPS 186-5, yang meliputi:

⁹ <https://eprint.iacr.org/2024/2070>

¹⁰ https://link.springer.com/chapter/10.1007/978-3-031-84151-4_24

- 1) p , bilangan prima yang berfungsi sebagai modulus, dengan ketentuan nilai p berada dalam rentang $2^{(L-1)}$ dan 2^L , dengan L adalah panjang bit dari p .
- 2) q , bilangan prima yang membagi $(p - 1)$, dengan ketentuan nilai q berada dalam rentang $2^{(N-1)}$ dan 2^N , dengan N adalah panjang bit dari q .
- 3) g , sebuah generator dari subkelompok dengan berorde q pada grup perkalian dalam *Finite Field* $GF(p)$, dengan nilai g berada dalam rentang $1 < g < p$.
- 4) x , kunci privat berupa integer acak atau semu-acak (*pseudorandom*) dalam rentang $[1, q - 1]$.
- 5) y , kunci publik yang dihitung sebagai $y = g^x \bmod p$.
- 6) k , bilangan rahasia unik untuk setiap pesan. Nilai k dibangkitkan secara acak atau *pseudorandom* dan berada dalam $[1, q - 1]$.

Panjang nilai L dan N harus memenuhi ketentuan yang ditetapkan oleh Algoritma AKI, yaitu minimum $L = 3072$ dan minimum $N = 256$.

Meskipun belum ditemukan serangan yang efektif terhadap $L = 2048$, $L = 3072$ lebih disarankan untuk sistem baru karena memberikan struktur keamanan yang lebih kuat. $L = 2048$ dapat digunakan, terutama untuk menjaga kompatibilitas dengan sistem yang telah ada (*existing system*). NIST SP 800-56A Rev. 3 [37] menetapkan parameter yang digunakan dalam proses pembangkitan pasangan kunci, sebagaimana ditampilkan dalam Tabel 11.

Tabel 11. Parameter pembangkitan pasangan kunci sesuai NIST SP 56a Rev 3

Proses	Parameter	Keterangan
INPUT	(p, q, g)	Nilai p , q , dan g integer p dan q bilangan prima
OUTPUT	(x, y)	Pasangan kunci privat dan publik yang dibangkitkan.

Proses	Parameter	Keterangan
		x adalah kunci privat, integer dalam rentang $[1, \min(2^N - 1, q - 1)]$ y adalah kunci publik, integer dalam rentang $[2, p - 2]$

3. Problem logaritma diskrit kurva eliptis

a. Algoritma

Terdapat empat algoritma berbasis *Elliptic Curve Discrete Logarithm Problem* (ECDLP), yaitu (i) kurva eliptis P, (ii) Brainpool, (iii) Curve, dan (iv) Edwards, masing-masing tersedia dalam beberapa variasi panjang parameter n . Rincian variasi algoritma primitif asimetrik yang mengacu pada ECDLP disajikan pada Tabel 12.

Tabel 12. Algoritma Primitif Asimetrik berbasis ECDLP

No	Algoritma	Variasi panjang n	Standar/ Referensi
1	Kurva eliptis P	P-521	a. NIST SP 800-186 <i>Recommendations for Discrete Logarithm-based Cryptography: Elliptic Curve Domain Parameters</i> [40]; b. FIPS 186-5 <i>Digital Signature Standard (DSS)</i> [7]
2	Brainpool	Brainpool 256r1 Brainpool 384r1 Brainpool 512r1	IETF RFC 5639 : <i>Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation</i> [41]
3	Curve	Curve 25519 Curve 448	a. NIST SP 800-186 <i>Recommendations for Discrete Logarithm-based Cryptography: Elliptic Curve Domain Parameters</i> [40]; b. IETF RFC 7748 : <i>Elliptic Curves for Security</i> [42].

No	Algoritma	Variasi panjang n	Standar/ Referensi
4	Edward	Edward25519 Edward448	a. NIST SP 800-186 <i>Recommendations for Discrete Logarithm-based Cryptography: Elliptic Curve Domain Parameters</i> [40]; b. IETF RFC 7748 : <i>Elliptic Curves for Security</i> [42]; c. IETF RFC 8032 : <i>Edwards-Curve Digital Signature Algorithm (EdDSA)</i> [43].

b. Penggunaan

1) Kurva eliptis P

Berdasarkan dokumen NIST SP 800-186 *Recommendations for Discrete Logarithm-based Cryptography: Elliptic Curve Domain Parameters* [40], kurva eliptis P-521 adalah kurva Weierstrass $W_{a,b}$ yang didefinisikan di atas *field* prima $GF(p)$.

Kurva ini memiliki:

- Orde sebesar $h \cdot n$, dengan $h = 1$, dan n adalah bilangan prima untuk kedua kurva.
- Quadratic twist* dari kurva P-521 memiliki orde sebesar $h_1 \cdot n_1$, dengan h_1 sekurang-kurangnya $5 \times 7 \times 69697531 \times 635884237$ dan n_1 merupakan bilangan prima.

Kurva ini memiliki parameter domain yang dituliskan sebagai:

$$D = (p, h, n, Type, a, b, G, \{Seed, c\})$$

dengan penjelasan sebagai berikut:

- p Bilangan prima yang mendefinisikan *finite field* $GF(p)$
- h Kofaktor dari kurva utama (dalam hal ini, $h = 1$)
- n Orde dari subgrup utama kurva, yaitu bilangan prima
- $Type$ Jenis kurva, yaitu "Weierstrass curve"

a, b	Koefisien dalam persamaan kurva
G	Titik dasar (<i>base point</i>) pada kurva dengan orde n
<i>Seed</i>	Nilai <i>seed</i> yang digunakan untuk menghasilkan parameter secara deterministik, umumnya sebagai bukti parameter dipilih secara acak yang dapat diverifikasi
c	Konstanta tambahan yang digunakan dalam pengujian atau pembangkitan parameter (misalnya konstanta validasi <i>hash</i> dari <i>seed</i>)

2) Brainpool

Berdasarkan IETF RFC 5639 [41], spesifikasi parameter domain Algoritma Brainpool didefinisikan sebagai berikut:

- Untuk setiap kurva, diberikan sebuah ID yang digunakan sebagai penanda unik kurva.
- p adalah bilangan prima yang menentukan *base field*. Nilai p harus kongruen dengan $3 \bmod 4$ ($p \equiv 3 \bmod 4$).
- A dan B adalah koefisien dari persamaan kurva eliptis:
$$y^2 = x^3 + A * x + B \bmod p$$
yang mendefinisikan kurva eliptis.
- $G = (x, y)$ adalah *base point*, yaitu titik pada kurva E yang memiliki orde prima, dengan x dan y sebagai koordinatnya.
- q adalah orde prima yang dibangkitkan oleh G .
- h adalah kofaktor G pada E . Contohnya, $\#E(GF(p))/q$.
- Untuk *twisted curve*, diberikan koefisien z , yaitu nilai yang digunakan untuk mendefinisikan isomorfisme F .

Metode pembangkitan parameter yang diperlukan dalam kurva Brainpool dijelaskan dalam Lampiran A dokumen IETF RFC 5639 [41].

Dalam implementasinya, algoritma simetris dan fungsi *hash* yang digunakan bersama parameter domain kurva eliptis harus memiliki tingkat keamanan yang setara atau lebih tinggi dibandingkan parameter domain tersebut. Rekomendasi panjang minimum kunci simetris dan fungsi *hash* yang setara atau lebih

tinggi dari parameter domain kurva eliptis Brainpool disajikan pada Tabel 13.

Tabel 13. Panjang minimum kunci simetris dan fungsi *hash* yang setara atau lebih tinggi dari parameter domain Brainpool

Parameter Domain Kurva Eliptis	Minimum Panjang Kunci Simetris (bit)	Fungsi Hash
brainpoolP256r1	128	SHA-256, SHA-384, SHA-512
brainpoolP384r1	192	SHA-384, SHA-512
brainpoolP512r1	256	SHA-512

3) Curve

Pembangkitan deterministik pada kurva algoritma *Curve* merujuk pada spesifikasi yang dijelaskan dalam Lampiran A dokumen IETF RFC 7748 [42]. Ketentuan parameter untuk kurva yang direkomendasikan, yaitu Curve25519 dan Curve448, dijabarkan secara rinci dalam Klausul 4 dokumen tersebut [42]. Parameter Algoritma Curve ditunjukkan pada Tabel 14.

Tabel 14. Parameter algoritma *Curve*

Notasi	Deskripsi	Parameter	
		Curve25519	Curve448
p	Bilangan prima yang mendefinisikan <i>base field</i> .	$2^{255} - 19$	$2^{448} - 2^{224} - 1$
A	Elemen di $GF(p)$, tidak boleh sama dengan -2 atau 2 .	486662	156326
<i>order</i>	Orde dari subgrup	$2^{255} + 0 \times 14def9dea2f79cd65812631a5cf5d3ed$	$2^{446} + 0 \times 8335dc163bb124b65129c96fde933d8d723a70aad8c873$

Notasi	Deskripsi	Parameter	
		Curve25519	Curve448
	dengan orde prima.		<i>d6d54a7bb0d</i>
<i>cofactor</i>	-	8	4
$U(P)$	Koordinat u dari titik P pada kurva Montgomery.	9	5
$V(P)$	Koordinat v dari titik P pada kurva Montgomery.	1478161944758954479 1020593568409986887 2646061346164752889 64881837755586237401	3552939267855681 7526412750206378 3334808976399387 7142718318808984 3516908878696741 0002932673765864 5509101427741472 6810583898559529 0606362
u, v	Koordinat titik pada kurva Montgomery.	(9,147816194475895447 91020593568409986887 26460613461647528896 4881837755586237401)	(5,35529392678556 8175264127502063 7833348089763993 8771427183188089 8435169088786967 4100029326737658 6455091014277414 7268105838985595 29)

4) Edwards

Parameter Edwards didefinisikan dalam IETF RFC 7748 [42] dan IETF RFC 8032 [43]. Parameter Edwards25519 dan Edwards448 dapat diimplementasikan berdasarkan Tabel 15 dan Tabel 16.

Tabel 15. Parameter Edwards25519

Parameter	Keterangan
p	p dari Edwards25519 dalam RFC 7748 [42] yaitu $2^{255} - 19$
b	256
Encoding dari $GF(p)$	255 bit <i>little-endian encoding</i> dari $\{0, 1, \dots, p - 1\}$
$H(x)$	$SHA - 512(dom2(phflag, context) \parallel x)$ [RFC6234]
c	Logaritma basis 2 dari <i>cofactor</i> edwards25519 dalam RFC 7748 [42] (yaitu, 3)
n	254
d	d dari Edwards25519 dalam RFC 7748 [42]
a	-1
B	$(X(P), Y(P))$ dari Edwards25519 dalam RFC 7748 [42]
L	<i>order</i> dari Edwards25519 dalam RFC 7748 [42]
$PH(x)$	x (yaitu, fungsi identifikasi)

Tabel 16. Parameter Edwards448

Parameter	Keterangan
p	p dari Edwards448 dalam RFC 7748 [42] (yaitu, $2^{488} - 2^{224} - 1$)
b	456
Encoding dari $GF(p)$	455 bit <i>little-endian encoding</i> dari $\{0, 1, \dots, p - 1\}$
$H(x)$	$SHAKE256(dom4(phflag, context) \parallel x, 114)$
$phflag$	0
c	Logaritma basis 2 dari kofaktor Edwards448 dalam RFC 7748 [42] (yaitu, 2)
n	447
d	d dari Edwards448 dalam RFC 7748 [42] (contoh - 39081)
a	1

Parameter	Keterangan
B	$(X(P), Y(P))$ dari Edwards448 dalam RFC 7748 [42]
L	<i>order</i> dari Edwards448 dalam RFC 7748 [42]
$PH(x)$	x (yaitu, fungsi identifikasi)

Implementasi Edwards, termasuk proses pembangkitan kunci, penandatanganan, dan verifikasi, telah dijelaskan secara lengkap dalam IETF RFC 7748 [42] dan RFC 8032 [43]. Penggunaan algoritma primitif asimetrik pada skema enkripsi maupun tanda tangan tetap dianggap akseptabel hingga tahun 2030, namun perlu mempertimbangkan potensi ancaman dari perkembangan teknologi kuantum yang terus berkembang.

B. Skema Enkripsi Asimetrik

Skema enkripsi asimetrik melibatkan dua operasi utama, yaitu enkripsi dan dekripsi. Proses enkripsi mengubah *plaintext* menjadi *ciphertext* menggunakan kunci publik milik penerima. Sebaliknya, dekripsi memulihkan *ciphertext* menjadi *plaintext* dengan menggunakan kunci privat yang sesuai [36].

1. Algoritma

Terdapat dua skema enkripsi asimetrik berbasis RSA yang direkomendasikan sebagai Algoritma AKI. Keduanya menggunakan mekanisme kriptografi yang didasarkan pada masalah faktorisasi integer, dan telah ditetapkan dalam berbagai standar internasional.

a. RSA-OAEP

Skema *RSA-Optimal Asymmetric Encryption Padding* (RSA-OAEP) merupakan algoritma enkripsi RSA yang telah distandarkan dalam dokumen berikut:

- 1) IETF RFC 8017: PKCS #1: *RSA Cryptography Specifications Version 2.2* [36];

- 2) NIST SP 800-56B *Revision 2: Recommendation for Pair-Wise Key Establishment Using Integer Factorization Cryptography* [44];
- 3) ISO/IEC 18033-2:2006 *Information technology — Security techniques — Encryption algorithms Part 2: Asymmetric ciphers* [34];
- 4) SNI ISO/IEC 18033-2:2006 Teknologi informasi — Teknik keamanan — Algoritma enkripsi — Bagian 2: Penyandian asimetris [33].

b. RSA-KEM

RSA-Key Encapsulation Mechanism (RSA-KEM) adalah mekanisme enkapsulasi kunci yang memungkinkan pengirim mengirimkan material kunci secara aman menggunakan kunci publik penerima [45]. Standar yang mencakup RSA-KEM antara lain:

- 1) IETF RFC 9690: *Use of the RSA-KEM Algorithm in the Cryptographic Message Syntax (CMS)* [45];
- 2) ISO/IEC 18033-2:2006 *Information technology — Security techniques — Encryption algorithms Part 2: Asymmetric ciphers* [34];
- 3) SNI ISO/IEC 18033-2:2006 Teknologi informasi — Teknik keamanan — Algoritma enkripsi — Bagian 2: Penyandian asimetris [33].

2. Penggunaan

Penggunaan skema enkripsi RSA perlu memperhatikan sejumlah aspek teknis berikut [44]:

a. Panjang modulus

Modulus RSA (n) harus memiliki panjang minimum 3072 bit dan dibentuk dari hasil perkalian dua bilangan prima, yaitu $n = p \cdot q$, dengan $p \neq q$. Panjang modulus yang digunakan menentukan tingkat kekuatan keamanannya, sebagaimana disajikan pada Tabel 17.

Meskipun belum ditemukan serangan yang efektif terhadap RSA dengan modulus 2048, modulus minimum 3072 lebih disarankan untuk sistem baru karena memberikan struktur keamanan yang lebih

kuat. RSA modulus 2048 dapat digunakan terutama untuk menjaga kompatibilitas dengan *existing system*.

Tabel 17. Panjang modulus yang direkomendasikan dan tingkat kekuatan keamanannya

Panjang modulus (bit)	Perkiraan maksimum kekuatan keamanan
3072	128
4096	152
6144	176
8192	200

b. Pembangkitan kunci

Pasangan kunci RSA (kunci publik dan kunci privat) harus dibangkitkan menggunakan metode dan kriteria yang ditetapkan dalam NIST SP 800-56B *Revision 2* [44].

c. Nilai eksponen publik e

Eksponen e harus berupa integer positif ganjil yang memenuhi ketentuan $2^{16} < e < 2^{256}$. Hal ini konsisten dengan FIPS 186-5 [7] dan praktik umum (nilai $e = 65537$ dipilih untuk efisiensi).

d. Pembangkitan bilangan acak

Bilangan acak yang digunakan dalam proses enkripsi dan pembangkitan kunci harus dibangkitkan menggunakan DRBG yang diakui dalam Algoritma AKI.

e. Verifikasi autentikasi kunci publik

Penerima harus memverifikasi autentikasi kunci publik, misalnya dengan memeriksa sertifikat digital yang ditandatangani oleh *Certificate Authority* (CA) tepercaya, sebagaimana diatur dalam Appendix B.3 NIST FIPS 186-5 [7].

f. Pemisahan penggunaan kunci

Pasangan kunci tidak boleh digunakan untuk lebih dari satu tujuan, seperti enkripsi dan penandatanganan, sesuai ketentuan pada NIST SP 800-56B Rev 2 [44] dan Lampiran B.3 NIST FIPS 186-5 [7].

Berikut merupakan uraian terkait penggunaan skema enkripsi asimetrik yang direkomendasikan sebagai bagian dari Algoritma AKI.

a. RSA-OAEP

RSA-OAEP terdiri atas operasi enkripsi dan dekripsi asimetris yang didasarkan pada sebuah fungsi *hash*, DRBG, *mask generation function* (MGF), *RSA Encryption Primitive* (RSAEP) dan *RSA Decryption Primitive* (RSADP). Operasi-operasi ini digunakan oleh skema *key transport*. Penjelasan dari masing-masing komponen adalah sebagai berikut:

- 1) Fungsi *hash* (*H*) yang digunakan adalah fungsi *hash* yang diakui dalam Algoritma AKI.
- 2) MGF merupakan fungsi yang menghasilkan rangkaian bit untuk *masking* rangkaian bit lainnya, dengan menggunakan sebuah fungsi *hash*. Ketentuan penggunaan MGF adalah sebagai berikut:
 - a) Fungsi *hash* yang digunakan oleh MGF tidak harus sama dengan fungsi *hash* *H* pada proses enkripsi dan dekripsi.
 - b) Setiap pihak hanya boleh menggunakan MGF satu kali untuk setiap transaksi dengan *seed* MGF tertentu. Dengan kata lain, dari satu *seed* MGF hanya boleh dihasilkan satu *mask* oleh setiap pihak.
- 3) Pembangkit bilangan acak yang digunakan harus berupa DRBG yang diakui dalam Algoritma AKI.
- 4) RSAEP menghasilkan *ciphertext* dari material kunci dan input tambahan menggunakan kunci publik RSA. RSA-OAEP dapat memproses hingga $nLen - 2HLen - 2$ byte dari material kunci, dengan $nLen$ adalah panjang modulus RSA milik penerima (dalam byte), dan $HLen$ adalah panjang *output* fungsi *hash* yang digunakan.
- 5) RSADP mengembalikan material kunci dari sebuah *ciphertext* dan input tambahan dengan menggunakan kunci privat RSA.

Setiap proses yang mengimplementasikan MGF, RSAEP, atau RSADP wajib *men-destroy* (menghancurkan) seluruh *local copies*

(salinan lokal) dari nilai-nilai sensitif yang digunakan atau dihasilkan selama eksekusi proses. Ruang lingkup penghancuran mencakup:

- 1) seluruh salinan lokal dari nilai input yang bersifat sensitif;
- 2) seluruh bagian *output* yang *locally store* (tersimpan secara lokal);
- 3) seluruh nilai lain yang bersifat sensitif dan tersimpan secara lokal yang digunakan atau dihasilkan selama eksekusi proses.

Penghancuran nilai-nilai yang tersimpan secara lokal tersebut harus dilakukan sebelum atau selama proses keluar dari proses, baik ketika keluar lebih awal karena terjadi kesalahan maupun ketika keluar secara normal dengan menghasilkan *output*. Ketentuan ini juga mencakup setiap bagian dari material kunci atau *output* yang tersimpan secara lokal, agar tidak ada data sensitif yang tertinggal setelah eksekusi proses selesai.

RSA-OAEP dengan panjang modulus 3072 bit diperkirakan memberikan tingkat keamanan sekitar 128 bit, dan akseptabel hingga tahun 2030. Skema ini dinilai lebih kuat secara kriptografis dan dianjurkan untuk digunakan dalam sistem baru [44].

b. RSA-KEM

PROFESIONAL • ELABORATIF • RESPONSIF
INTEGRATIF • SOLID • AKUNTABEL • INOVATIF

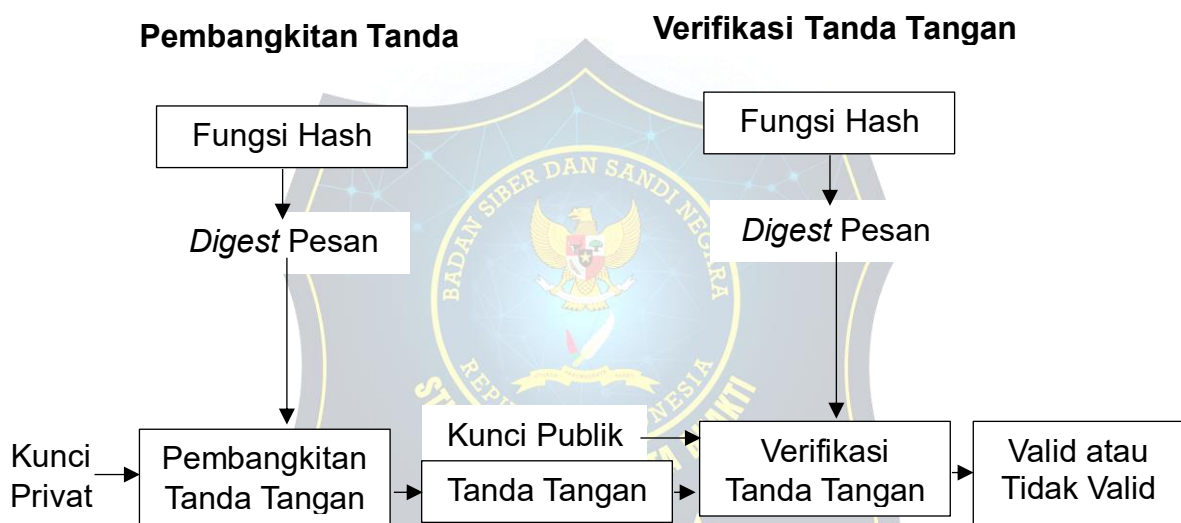
RSA-KEM digunakan untuk mengenkripsi *keying material* yang selanjutnya dimanfaatkan dalam skema kriptografi simetris melalui mekanisme *Data Encapsulation Mechanism* (DEM). Spesifikasi RSA-KEM dan DEM tercantum dalam ISO/IEC 18033-2 [33].

RSA-KEM dengan panjang modulus 3072 bit memberikan tingkat keamanan sekitar 128 bit dan dinilai masih akseptabel hingga tahun 2030. Meskipun belum ditemukan serangan yang efektif terhadap RSA-KEM, RSA-OAEP lebih disarankan untuk sistem baru karena memberikan struktur keamanan yang lebih kuat. RSA-KEM tetap digunakan terutama untuk menjaga kompatibilitas dengan *existing system*.

C. Skema Tanda Tangan Asimetrik

1. Algoritma

Berdasarkan FIPS 186-5 [7], tanda tangan digital adalah bentuk elektronik dari tanda tangan basah yang berfungsi menjamin keaslian penanda tangan dan integritas data. Jaminan ini tetap berlaku selama proses transmisi maupun penyimpanan data, asalkan algoritma kriptografi yang digunakan memenuhi standar dan diimplementasikan dengan benar. Secara umum, skema tanda tangan digital digambarkan pada Gambar 11.



Gambar 11. Skema tanda tangan digital digambar ulang dari [7]

Sebagaimana ditunjukkan pada Gambar 11, tanda tangan digital dibangkitkan oleh penanda tangan menggunakan kunci privat dan diverifikasi oleh pemverifikasi menggunakan kunci publik. Verifikasi dilakukan dengan membandingkan *digest* pesan dengan hasil dekripsi tanda tangan menggunakan kunci publik. Kunci privat bersifat rahasia dan hanya boleh diakses oleh pemilik sah untuk mencegah pemalsuan tanda tangan. Sementara itu, meskipun kunci publik tidak bersifat rahasia, integritasnya harus dijaga karena digunakan untuk memverifikasi keaslian tanda tangan melalui *digest* pesan.

Algoritma seperti RSA, ECDSA, dan EdDSA menggunakan fungsi *hash* untuk mengubah pesan menjadi *digest* pesan berukuran tetap yang

digunakan dalam proses pembangkitan tanda tangan digital. Sebelum menyatakan bahwa sebuah tanda tangan digital valid, pemverifikasi perlu memperoleh sejumlah jaminan, yaitu:

- a. kepastian atas identitas penanda tangan yang diklaim;
- b. keabsahan parameter domain yang digunakan (khususnya untuk ECDSA dan EdDSA);
- c. validitas kunci publik yang digunakan untuk proses verifikasi;
- d. jaminan bahwa penanda tangan yang diklaim benar-benar memiliki kunci privat yang sesuai saat membangkitkan tanda tangan digital tersebut.

Metode untuk memperoleh jaminan-jaminan tersebut dijelaskan secara rinci dalam NIST SP 800-89 [46].

Terdapat 4 (empat) algoritma skema tanda tangan digital asimetrik yang direkomendasikan sebagai bagian Algoritma AKI. Keempat algoritma tersebut diklasifikasikan ke dalam tiga teknik utama, yaitu:

- a. Algoritma tanda tangan digital RSA (*RSA digital signature algorithm*)
Algoritma ini menggunakan mekanisme kriptografi berbasis faktorisasi integer (*integer factorization*). Dua skema tanda tangan digital yang termasuk dalam teknik algoritma tanda tangan digital RSA dan direkomendasikan sebagai Algoritma AKI adalah:

- 1) RSASSA-PKCS1-v1

Skema ini dinyatakan dalam standar internasional berikut:

- a) ISO/IEC 14888-2 *Digital Signatures with Appendix Part 2: Integer Factorization-Based Mechanisms* [47];
- b) IETF RFC 8017 : *Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.2* [36];
- c) FIPS 186-5 *Digital Signature Standard (DSS)* [7].

- 2) RSSASSA-PSS

Skema ini dinyatakan dalam standar internasional berikut:

- a) ISO/IEC 14888-2:2008 *Digital Signatures with Appendix Part 2: Integer Factorization-Based Mechanisms* [47];
- b) IETF RFC 8017: *Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.2* [36];

c) FIPS 186-5 *Digital Signature Standard (DSS)* [7].

b. Algoritma tanda tangan digital kurva eliptis (*Elliptic Curve Digital Signature Algorithm - ECDSA*)

ECDSA merupakan algoritma tanda tangan digital yang menggunakan mekanisme berbasis logaritma diskrit pada kurva eliptis. Satu skema tanda tangan digital dalam teknik ini yang direkomendasikan sebagai Algoritma AKI yaitu skema ECDSA.

Skema ini dinyatakan dalam standar internasional berikut:

- 1) ISO/IEC 14888-3:2018 *IT Security techniques — Digital signatures with appendix Part 3: Discrete logarithm based mechanisms* [48];
- 2) NIST SP 800-186 *Recommendations for Discrete Logarithm-based Cryptography: Elliptic Curve Domain Parameters* [40];
- 3) FIPS 186-5 *Digital Signature Standard (DSS)* [7];
- 4) IETF RFC 6979 *Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA)*.

c. Algoritma tanda tangan digital Kurva Edward (*Edwards Curve Digital Signature Algorithm — EdDSA*)

EdDSA adalah skema tanda tangan digital yang menggunakan varian tanda tangan Schnorr berbasis Kurva Edward terpuntir (*twisted Edwards curves*). Mekanisme ini mengandalkan prinsip logaritma diskrit pada kurva eliptis. Satu skema tanda tangan digital yang direkomendasikan sebagai Algoritma AKI adalah EdDSA.

Skema ini dinyatakan dalam standar internasional berikut:

- 1) NIST SP 800-186 *Recommendations for Discrete Logarithm-based Cryptography: Elliptic Curve Domain Parameters* [40];
- 2) FIPS 186-5 *Digital Signature Standard (DSS)* [7];
- 3) IETF RFC 8032 *Edwards-Curve Digital Signature Algorithm (EdDSA)* [43].

2. Penggunaan

Dalam penerapan skema tanda tangan digital, baik penanda tangan maupun pemverifikasi wajib memastikan aspek-aspek teknis dan keamanan berikut:

a. Pembangkitan pasangan kunci

- 1) Apabila menggunakan Infrastruktur Kunci Publik (IKP) milik pihak ketiga (*Public Certification Authority (Public CA)/PSrE publik*), proses pembangkitan pasangan kunci wajib dilakukan pada sistem milik instansi/organisasi secara luring, kemudian sertifikat ditandatangani oleh CA.
- 2) *Public CA* hanya dapat digunakan untuk sistem yang dapat diakses langsung oleh publik. Untuk sistem internal diutamakan menggunakan IKP tersendiri.
- 3) Untuk infrastruktur kritis dan strategis milik negara wajib menggunakan IKP yang dikelola secara mandiri atau menggunakan IKP yang disediakan oleh penyelenggara negara.

b. Validasi kunci publik

Pemverifikasi harus memastikan bahwa kunci publik penandatanganan valid. Hal ini dapat dilakukan dengan memverifikasi sertifikat digital yang memuat kunci publik tersebut yang ditandatangani oleh pihak ketiga tepercaya seperti CA.

c. Penggunaan khusus pasangan kunci

Pasangan kunci yang digunakan untuk pembangkitan dan verifikasi tanda tangan digital tidak boleh digunakan untuk tujuan kriptografi lainnya, seperti pembentukan kunci (*key establishment*) untuk mencegah risiko keamanan dan kerentanan kriptografi.

Berikut merupakan uraian terkait penggunaan skema-skema tanda tangan digital asimetrik yang direkomendasikan sebagai bagian dari Algoritma AKI.

a. RSASSA-PKCS1-v1.5

Selain ketentuan yang dijelaskan dalam IETF RFC 8017 [36], penggunaan skema RSASSA-PKCS1-v1.5 harus memenuhi batasan tambahan sebagai berikut:

1) Panjang modulus

Modulus RSA (n) harus memiliki panjang minimum 3072 bit dan dibentuk dari hasil perkalian dua bilangan prima (p dan q).

2) Pembangkitan kunci

Pasangan kunci RSA (kunci privat dan kunci publik) harus dibangkitkan menggunakan kriteria dan metode yang ditetapkan dalam Lampiran B.3 FIPS 186-5 [7].

3) Fungsi *hash*

Fungsi *hash* yang dapat digunakan meliputi algoritma dalam keluarga SHA2 dan SHA3 yang telah diakui dalam Algoritma AKI yaitu:

a) SHA-224, SHA-384, SHA-512/224 dan SHA-512/256 (SHA2);

b) SHA3-224, SHA3-256, SHA3-384, dan SHA3-512 (SHA3).

SHA-256 dan SHA-512 tetap dapat digunakan dalam implementasi RSASSA-PKCS1-v1.5 karena telah terbukti aman dan banyak digunakan secara global, meskipun tidak tercantum secara eksplisit dalam daftar Algoritma AKI. Hal ini karena keduanya bukan merupakan satu-satunya komponen kriptografi dalam pembangkitan tanda tangan digital RSA.

4) Nilai eksponen publik e

Eksponen e harus berupa integer ganjil positif yang memenuhi ketentuan $2^{16} < e < 2^{256}$.

5) Pembangkitan bilangan acak

Bilangan acak harus dibangkitkan menggunakan DRBG yang diakui dalam Algoritma AKI.

Skema RSASSA-PKCS1-v1_5 dengan modulus 3072 memberikan tingkat keamanan sekitar 128 bit. Penggunaan skema ini masih akseptabel hingga pertengahan tahun 2030. Meskipun belum diketahui adanya serangan terhadap RSASSA-PKCS1-v1-5, untuk

meningkatkan ketahanan terhadap potensi serangan di masa depan, penggunaan skema RSASSA-PSS lebih dianjurkan dalam sistem baru. Skema RSASSA-PKCS1-v1_5 dipertahankan untuk keperluan kompatibilitas dengan *existing system*.

b. RSASSA-PSS

Penggunaan skema RSASSA-PSS juga mengacu pada ketentuan dalam IETF RFC 8017 [36] serta batasan tambahan berikut:

1) Panjang modulus dan pembangkitan pasangan kunci

Sama seperti RSASSA-PKCS1-v1.5, panjang modulus minimum adalah 3072 bit dengan dua bilangan prima penyusun dan pasangan kunci dibangkitkan sesuai metode dalam Lampiran B.3 FIPS 186-5 [7].

2) Fungsi *hash*

Fungsi *hash* yang direkomendasikan meliputi:

- a) SHA-224, SHA-384, SHA-512/224 dan SHA-512/256 (SHA2);
- b) SHA3-224, SHA3-256, SHA3-384, dan SHA3-512 (SHA3);
- c) XOF (*extendable-output function*) yang terdiri atas SHAKE128 dengan panjang *output* sebesar 256 bit dan SHAKE256 dengan panjang *output* sebesar 512 bit.

SHA-256 dan SHA-512 tetap dapat digunakan dalam implementasi RSASSA-PSS karena telah terbukti aman dan banyak digunakan secara global, meskipun tidak tercantum secara eksplisit dalam daftar Algoritma AKI. Hal ini karena keduanya bukan merupakan satu-satunya komponen kriptografi dalam pembangkitan tanda tangan digital RSA.

3) Eksponen dan bilangan acak

Ketentuan eksponen e dan pembangkitan bilangan acak sama seperti pada skema RSASSA-PKCS1-v1.5.

Skema RSASSA-PSS dengan panjang modulus 3072 bit memberikan tingkat keamanan sekitar 128 bit dan akseptabel hingga tahun 2030. Skema ini merupakan pilihan yang lebih kuat secara kriptografis dan dianjurkan untuk digunakan dalam sistem baru.

c. ECDSA

Sesuai ISO/IEC 14888-3:2016 [48] dan FIPS 186-5 [7], penanda tangan dan pemverifikasi harus memastikan pembangkitan parameter publik dilakukan secara benar. Kurva eliptis yang direkomendasikan dalam Algoritma AKI adalah P-384 dan P-521. Untuk masing-masing kurva, fungsi *hash* yang dapat digunakan meliputi:

- 1) ECDSA dengan kurva P-384,
 - a) SHA-224, SHA-384, SHA-512/224 dan SHA-512/256 (SHA2);
 - b) SHA3-224, SHA3-256, SHA3-384, dan SHA3-512 (SHA3);
 - c) XOF (*extendable-output function*) yang terdiri atas SHAKE128 dan SHAKE256.
- 2) ECDSA dengan kurva P-521
 - a) SHA-224, SHA-384, SHA-512/224 dan SHA-512/256 (SHA2);
 - b) SHA3-224, SHA3-256, SHA3-384, dan SHA3-512 (SHA3);
 - c) XOF (*extendable-output function*) yang terdiri atas SHAKE128 dan SHAKE256.

SHA-256 dan SHA-512 tetap dapat digunakan dalam implementasi ECDSA karena telah terbukti aman dan banyak digunakan secara global, meskipun tidak tercantum secara eksplisit dalam daftar Algoritma AKI. Hal ini karena keduanya bukan merupakan satu-satunya komponen kriptografi dalam pembangkitan tanda tangan digital ECDSA.

ECDSA dengan kurva P-384 memberikan tingkat keamanan sekitar 192 bit, sedangkan ECDSA dengan kurva P-521 memberikan tingkat keamanan sekitar 256 bit. Skema ini akseptabel hingga tahun 2030.

d. EdDSA

Kurva eliptis yang direkomendasikan Algoritma AKI untuk EdDSA adalah Edwards25519 dan Edwards448.

Keamanan skema EdDSA bergantung pada parameter-parameter domain sebagai berikut:

- 1) *Base point* $G = (x_G, y_G)$ dengan orde prima n ;
- 2) Kunci privat d , kunci publik Q ;

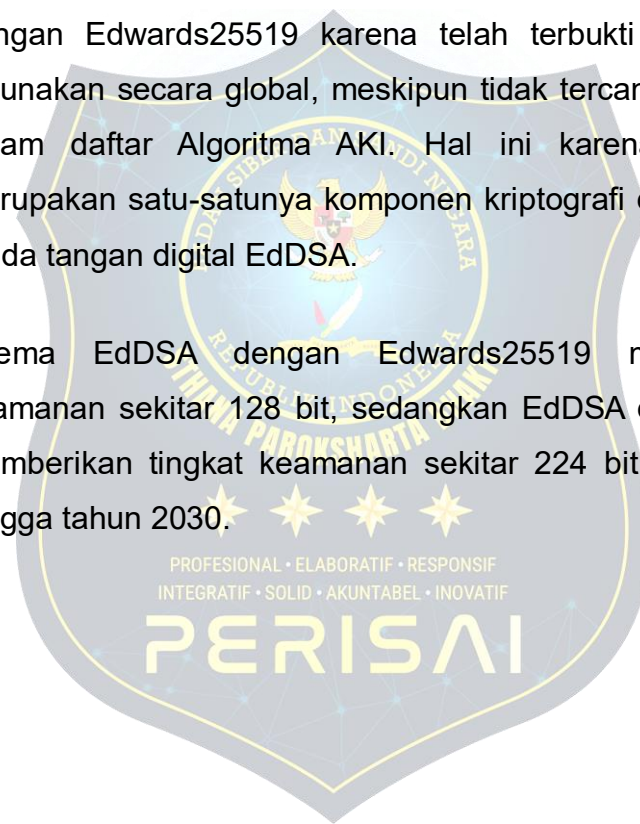
- 3) Parameter integer b , dan konstanta c dengan $c = 3$ untuk EdDSA dengan Edwards25519 dan $c = 2$ untuk EdDSA dengan Edwards448).

Fungsi *hash* yang dapat digunakan dalam EdDSA sesuai IETF RFC 8032 [43] dan FIPS 186-5 [7] adalah:

- 1) Untuk EdDSA dengan Edwards25519: SHA-512.
- 2) Untuk EdDSA dengan Edwards448: SHAKE256.

SHA-512 tetap dapat digunakan dalam implementasi skema EdDSA dengan Edwards25519 karena telah terbukti aman dan banyak digunakan secara global, meskipun tidak tercantum secara eksplisit dalam daftar Algoritma AKI. Hal ini karena keduanya bukan merupakan satu-satunya komponen kriptografi dalam pembangkitan tanda tangan digital EdDSA.

Skema EdDSA dengan Edwards25519 memberikan tingkat keamanan sekitar 128 bit, sedangkan EdDSA dengan Edwards448 memberikan tingkat keamanan sekitar 224 bit. EdDSA akseptabel hingga tahun 2030.



DAFTAR PUSTAKA

- [1] NIST, NIST SP 800-90A Rev. 1, Recommendation for Random Number Generator Using Deterministic Random Bit Generators, NIST, 2015.
- [2] E. Barker, NIST SP 800-175B Rev. 1 Guideline for Using Cryptographic Standards in the Federal Government: Cryptographic Mechanisms, Gaithersburg, MD: NIST U.S. Department of Commerce, 2020.
- [3] N. Mouha dan M. Dworkin, "NIST IR 8459 Report on the Block Cipher Modes of Operation in the NIST SP 800-38 Series," NIST U.S. Department of Commerce, Gaithersburg, MD, 2024.
- [4] ISO/IEC, ISO/IEC 29192-1:2012 Information technology — Security techniques — Lightweight cryptography — Part 1: General, Geneva: International Organization for Standardization/International Electrotechnical Commission, 2012.
- [5] NIST. [Online]. Available: https://csrc.nist.gov/glossary/term/provable_prime.
- [6] NIST. [Online]. Available: https://csrc.nist.gov/glossary/term/probable_prime.
- [7] NIST, FIPS PUB 186-5 Digital Signature Standard (DSS), Gaithersburg, MD: NIST U.S. Department of Commerce, 2023.
- [8] ISO/IEC 18033-3:2010 Information technology – Security techniques – Encryption algorithms, Part 3: Block ciphers.
- [9] SNI ISO/IEC 18033-3:2010 Teknologi informasi – Teknik keamanan – Algoritma enkripsi – Bagian 3: Penyandian blok (ISO/IEC 18033-3:2010 dan ISO/IEC 18033-3:2010 Amd.1:2021, IDT, Eng), Jakarta: Badan Standardisasi Nasional, 2022.
- [10] NIST, FIPS PUB 197 Advanced Encryption Standard (AES), Gaithersburg, MD: NIST U.S. Department of Commerce, 2023.
- [11] M. Dworkin, NIST SP 800-38A Recommendation for Block Cipher Modes of Operations: Methods and Techniques, NIST U.S. Department of Commerce, 2001.
- [12] ISO/IEC 11016:2017 Information technology - Security techniques - Modes of operation for an n-bit block cipher.
- [13] SNI ISO/IEC 10116:2017 Teknologi informasi - Teknik keamanan - Mode operasi untuk penyandian blok n-bit (ISO/IEC 10116:2017 dan ISO/IEC

10116:2017/Amd1:2021, IDT, Eng), Jakarta: Badan Standardisasi Nasional, 2025.

- [14] M. Dworkin, NIST SP 800-38C Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality, NIST U.S. Department of Commerce, 2007.
- [15] M. Dworkin, NIST SP 800-38D Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC, NIST U.S. Department of Commerce, 2007.
- [16] E. Barker, NIST SP 800-57 Part 1 Rev. 4 Recommendation for Key Management, Part 1: General, Gaithersburg, MD: NIST U.S. Department of Commerce, 2016.
- [17] H. Ferraiolo, K. Mehta, S. Francomacaro, R. Chandramouli dan S. Gupta, NIST SP 800-73-5 Interfaces for Personal Identity Verification: Part 1 – PIV Card Application Namespace, Data Model and Representation, NIST U.S. Department of Commerce, 2024.
- [18] H. Ferraiolo, K. Mehta, S. Francomacaro, R. Chandramouli dan S. Gupta, NIST SP 800-73-5 Interfaces for Personal Identity Verification: Part 2 – PIV Card Application Card Command Interface, NIST U.S. Department of Commerce, 2024.
- [19] H. Ferraiolo, K. Mehta, S. Francomacaro, R. Chandramouli dan S. Gupta, NIST SP 800-73-5 Interfaces for Personal Identity Verification: Part 3 – PIV Client Application Programming Interface, NIST U.S. Department of Commerce, 2024.
- [20] H. Wu, "The Stream Cipher HC-128," eSTREAM Final Report, 2008.
- [21] D. Bernstein, "The Salsa20 Family of Stream Ciphers," 2008.
- [22] C. B. e. al., "Sosemanuk, a fast software-oriented stream cipher," eSTREAM Submission, 2006.
- [23] SNI ISO/IEC 18033-4:2011 Teknologi informasi – Teknik keamanan – Algoritma enkripsi – Bagian 4: Penyandian alir, Jakarta: Badan Standardisasi Nasional, 2022.
- [24] Y. Nir dan A. Langley, RFC 8439 ChaCha20 and Poly1305 for IETF Protocols, RFC Editor, 2018.
- [25] NIST, FIPS 180-4 Secure Hash Standard (SHS), Gaithersburg, MD: NIST U.S. Department of Commerce, 2015.
- [26] ISO/IEC 10118-3:2004, Information technology - Security technique - Hash function - Part 3: Dedicated hash - functions, Geneva: International Organization for Standardization, 2004.

- [27] NIST, FIPS 202 SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions, Gaithersburg, MD: NIST U.S. Department of Commerce, 2015.
- [28] M.-J. O. Saarinen dan J.-P. Aumasson, RFC 7693 The BLAKE2 Cryptographic Hash and Message Authentication Code (MAC), RFC Editor, 2015.
- [29] NIST SP 800-107 Rev 1, Recommendation for Applications Using Approved Hash Algorithms, Gaithersburg, MD: NIST U.S. Department of Commerce, 2012.
- [30] ISO/IEC 18031 Information technology — Security techniques — Random bit generation, Third Edition, 2025.
- [31] SNI ISO IEC 18031:2025 Ditetapkan oleh BSN tahun 2025 Teknologi informasi – Teknik keamanan – Pembangkit bit random (ISO/IEC 18031:2025, IDT, Eng), Jakarta: Badan Stand, 2025.
- [32] NIST SP 800-131Ar3 ipd, Transitioning the Use of Cryptographic Algorithms and Key Lengths, Gaithersburg, MD: NIST U.S. Department of Commerce, 2024.
- [33] SNI ISO/IEC 18033-2:2021 Teknologi informasi – Teknik keamanan – Algoritma penyandian – Bagian 2 : Penyandian asimetris, Jakarta: Badan Standardisasi Nasional, 2022.
- [34] ISO/IEC 18033-2:2006 Information technology — Security techniques — Encryption algorithms Part 2: Asymmetric ciphers.
- [35] E. Barker, NIST SP 800-57 Part 1 Rev. 5 Recommendation for Key Management: Part 1 – General, Gaithersburg, MD: NIST U.S. Department of Commerce, 2020.
- [36] B. & S. J. Kaliski, RFC 8017 PKCS #1: RSA Cryptography Specifications Version 2.2, RFC Editor, 2016.
- [37] E. Baker, NIST SP 800-56A Rev. 3, Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography, Gaithersburg, MD: NIST U.S. Department of Commerce, 2018.
- [38] M. Kojo dan T. Kivinen, RFC 3526 More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE), RFC Editor, 2003.
- [39] NIST, FIPS PUB 186-4 Digital Signature Standard (DSS), Gaithersburg, MD: NIST U.S. Department of Commerce, 2013.
- [40] L. Chen, D. Moody, A. Regenscheid, A. Robinson dan K. Randall, NIST SP 800-186 Recommendations for Discrete Logarithm-based Cryptography: Elliptic Curve Domain Parameters, Gaithersburg, MD: NIST U.S. Department of Commerce, 2023.

- [41] M. L. a. J. Merkle, RFC 5639 Elliptic Curve Cryptography Brainpool Standard Curves and Curve Generation, RFC Editor, 2010.
- [42] M. H. a. S. A. Langley, RFC 7748 Elliptic Curves for Security, RFC Editor, 2016.
- [43] S. Josefsson dan I. Liusvaara, RFC 8032 Edwards-Curve Digital Signature Algorithm (EdDSA), RFC Editor, 2017.
- [44] NIST, NIST Special Publication 800-56B Revision 2 : Recommendation for Pair-Wise Key Establishment Using Integer Factorization Cryptography, NIST, 2019.
- [45] R. Housley dan S. Turner, RFC 9690 Use of the RSA-KEM Algorithm in the Cryptographic, RFC Editor, 2025.
- [46] E. Barker, NIST SP 800-89 Recommendation for Obtaining Assurances for Digital Signature Applications, U.S. Department of Commerce, 2006.
- [47] ISO/IEC 14888-2:2008 Information technology — Security techniques — Digital signatures with appendix Part 2: Integer factorization based mechanisms, 2008.
- [48] ISO/IEC 14888-3:2018 IT Security techniques — Digital signatures with appendix Part 3: Discrete logarithm based mechanisms, 2018.





**BADAN SIBER
DAN SANDI
NEGARA**



**Deputi Bidang Strategi dan Kebijakan Keamanan Siber dan Sandi
Badan Siber dan Sandi Negara**

Jl. Cijayanti No. 70, Kec. Babakan Madang, Kabupaten Bogor

2025

PUBLIK