Logo Universitas Bale Bandung

Buku Ajar Kriptografi

Program Studi S1 Matematika

Fakultas Matematika dan IPA
Universitas Bale Bandung

8 Oktober 2025

Daftar Isi

Daftar Gambar

Daftar Tabel

Listings

Bab 1

Pengantar Kriptografi

1.1 Definisi dan Terminologi Kriptografi

Kriptografi adalah disiplin ilmu yang mempelajari teknik untuk mengamankan informasi dengan mengubahnya menjadi bentuk yang tidak dapat dibaca tanpa pengetahuan tertentu, biasanya berupa kunci. Dalam konteks modern, kriptografi meliputi kerangka formal untuk kerahasiaan, integritas, otentikasi, dan nir-sangkal, serta didasari oleh asumsi komputasional yang ketat. Istilah-istilah kunci yang digunakan mencakup plainteks, cipherteks, enkripsi, dekripsi, kunci simetri dan kunci publik, serta keamanan terukur komputasional dibandingkan keamanan sempurna. Uraian komprehensif yang terbuka tersedia pada Menezes, Oorschot **and** Vanstone (1996) dan Boneh **and** Shoup (2020).

Kriptografi tidak berdiri sendiri, tetapi berinteraksi erat dengan teori informasi, teori bilangan, dan desain protokol. Perkembangan modern menekankan model serangan yang eksplisit (misalnya chosen-plaintext dan chosen-ciphertext) untuk menurunkan jaminan keamanan yang dapat dibuktikan. Dalam praktik, pemilihan parameter, implementasi tahan-kanal-samping, dan pengelolaan kunci menjadi faktor penentu apakah sifat teoretik tersebut terealisasi. Teks terbuka seperti Boneh **and** Shoup (2020) memberikan kerangka formal untuk memahami jaminan ini.

1.2 Layanan Kriptografi

Layanan kriptografi utama meliputi kerahasiaan, integritas data, otentikasi entitas/pesan, dan nir-sangkal. Kerahasiaan dicapai melalui algoritma enkripsi yang tepat dan mode operasi yang benar; integritas dan otentikasi sering diwujudkan lewat fungsi hash aman, MAC, dan tanda tangan digital. Nir-sangkal menyediakan bukti yang dapat diverifikasi pihak ketiga bahwa suatu entitas melakukan tindakan tertentu, lazimnya dengan tanda tangan digital yang didukung infrastruktur kunci publik.

Pemetaan layanan ke primitif kriptografi tidak selalu satu-ke-satu; misalnya, AEAD

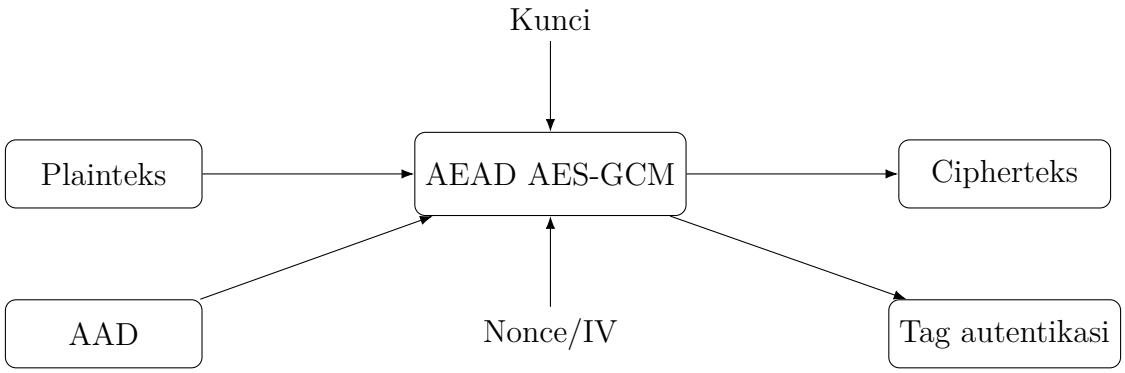
(Authenticated Encryption with Associated Data) menggabungkan kerahasiaan dan integritas sekaligus. Kesesuaian layanan dengan kebutuhan aplikasi harus mempertimbangkan model ancaman dan asumsi trust. Ringkasan layanan dan konstruksi yang relevan dapat ditemukan pada Dworkin (2008) dan Dworkin (2007).

Tabel 1.1: Pemetaan layanan ke primitif dan contoh standar

Layanan	Primitif	Contoh standar
Kerahasiaan	Cipher blok/alir + mode	AES-GCM (Dworkin 2007), ChaCha20-Poly1305 (Langle
Integritas	Fungsi hash, MAC	SHA-256 (*Secure Hash Standard (SHS)* 2015), HMAC
Otentikasi	MAC, tanda tangan digital	RSA-PSS (Schmidt **andothers** 2016), ECDSA
Nir-sangkal	Tanda tangan digital	RSA/ECDSA (Schmidt **andothers** 2016)

1.2.1 Contoh AEAD: AES-GCM

AEAD menyediakan kerahasiaan sekaligus integritas/asli data terkait (AAD) dalam satu primitf terintegrasi. AES-GCM adalah skema AEAD yang distandardisasi luas dan efisien di perangkat keras dan perangkat lunak (Dworkin 2008; Dworkin 2007).



Gambar 1.1: Alur tinggi AEAD AES-GCM (Dworkin 2008; Dworkin 2007).

Listing 1.1: Contoh enkripsi-dekripsi AES-GCM di Python

```
from cryptography.hazmat.primitives.ciphers.aead import AESGCM
import os

key = AESGCM.generate_key(bit_length=256)
aesgcm = AESGCM(key)
nonce = os.urandom(12)  # 96-bit IV
aad = b"header"
plaintext = b"rahasia"

ciphertext = aesgcm.encrypt(nonce, plaintext, aad)
```

```
recovered = aesgcm.decrypt(nonce, ciphertext, aad)
assert recovered == plaintext
```

Contoh diadaptasi dari dokumentasi pustaka `cryptography` (The cryptography developers 2024a).

1.3 Sejarah Kriptografi

Sejarah kriptografi berawal dari teknik-teknik klasik seperti substitusi dan transposisi yang bertujuan menyamarkan pesan dari pihak yang tidak berwenang. Revolusi terjadi saat Perang Dunia II dengan mesin Enigma dan analisis kriptografi terapan berskala besar. Era modern dimulai pada 1970-an dengan publikasi DES dan terobosan kriptografi kunci publik oleh Diffie, Hellman, dan RSA, yang mengubah paradigma distribusi kunci.

Perkembangan selanjutnya ditandai dengan standarisasi AES, penguatan teori keamanan berbasis bukti, serta ekspansi ke ranah protokol seperti TLS, kerangka PKI, dan kriptografi pasca-kuantum. Sumber sejarah primer terbuka meliputi Diffie **and** Hellman (1976) dan Rivest, Shamir **and** Adleman (1978) serta dokumen standar terbuka seperti *Advanced Encryption Standard (AES)* (2001) untuk AES.

1.4 Kriptanalisis

Kriptanalisis adalah studi tentang metode memecahkan atau melemahkan sistem kriptografi. Spektrum serangan mencakup analisis frekuensi pada cipher klasik, serangan diferensial dan linear pada cipher blok modern, hingga serangan berbasis side-channel seperti power analysis dan timing attacks. Kekuatan praktis serangan sering kali bergantung pada detail implementasi dan protokol, bukan semata algoritma inti.

Dalam model serangan modern, penyerang mungkin memiliki kemampuan chosen-plaintext atau chosen-ciphertext, sehingga definisi keamanan seperti IND-CPA dan IND-CCA menjadi acuan formal. Referensi mendalam tentang teknik kriptanalisis dapat ditemukan pada Biham **and** Shamir (1991) **and** Kocher, Jaffe **and** Jun (1999) dan berbagai survei terbuka.

1.5 Kriptografi Simetri dan Kriptografi Nirsimetri

Kriptografi simetri menggunakan kunci yang sama untuk enkripsi dan dekripsi, menawarkan kinerja tinggi serta kesederhanaan distribusi kunci pada skala kecil. Algoritma yang umum termasuk cipher blok seperti AES dan cipher alir seperti ChaCha; keamanan bergantung pada struktur internal serta mode operasi yang tepat. Di sisi lain, kriptografi

nirsimetri menggunakan pasangan kunci publik-pribadi, memungkinkan fungsi seperti pertukaran kunci, enkripsi kunci publik, dan tanda tangan digital.

Perbedaan mendasar antara keduanya tercermin pada model keamanan, ukuran kunci, dan biaya komputasi. Dalam praktik, sistem nyata biasanya mengombinasikan keduanya dalam skema hibrida: kunci sesi dihasilkan via pertukaran kunci nirsimetri lalu digunakan oleh cipher simetri untuk efisiensi. Lihat Rescorla (2018) untuk contoh penerapan pada TLS 1.3.

1.6 Fungsi Hash

Fungsi hash kriptografis memetakan masukan berdimensi arbitrer ke keluaran berukuran tetap dengan sifat tahan-tabrakan, tahan-preimage, dan tahan-second-preimage. Keluarga SHA-2 dan SHA-3 merupakan standar yang banyak digunakan karena desain yang dianalisis secara luas. Fungsi hash menjadi komponen sentral dalam konstruksi MAC, tanda tangan digital, verifikasi integritas, dan komitmen kriptografis.

Pemilihan fungsi hash harus mempertimbangkan kecepatan, keamanan terhadap serangan diferensial, dan dukungan perangkat keras. Standar dan analisis mutakhir dapat ditemukan pada *Secure Hash Standard (SHS)* (2015) **and** *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions* (2015) serta tinjauan ringkas seperti Bertoni **andothers** (2013).

1.7 Kriptografi di Indonesia

Ekosistem kriptografi di Indonesia berkembang melalui adopsi standar internasional dan pengembangan kebijakan nasional terkait keamanan informasi. Lembaga pemerintah dan akademia berperan dalam riset, pelatihan, serta standardisasi penerapan teknologi kriptografi pada layanan publik dan industri. Kerangka regulasi dan pedoman teknis mengarahkan praktik terbaik, termasuk pengelolaan kunci, sertifikasi perangkat, dan interoperabilitas.

Kolaborasi lintas institusi dan partisipasi pada forum internasional menjadi kunci untuk mengikuti perkembangan global termasuk kriptografi pasca-kuantum dan keamanan perangkat lunak. Rujukan umum terhadap praktik dan standar dapat ditelusuri melalui publikasi terbuka NIST dan IETF yang diadopsi luas secara global (*Guideline for Using Cryptographic Standards in the Federal Government: Directives, Mandates and Policies* 2016; Cooper **andothers** 2008).

Bab 2

Landasan Matematika

2.1 Teori Informasi

Teori informasi menyediakan alat kuantitatif untuk mengukur ketidakpastian dan informasi dalam sinyal, yang relevan untuk menilai keamanan skema kriptografi. Konsep entropi Shannon menangkap rata-rata informasi yang dikandung oleh sumber simbol, sehingga berguna untuk memahami redundansi bahasa alami dan implikasinya pada kriptanalisis klasik. Dalam konteks keamanan sempurna, kerangka Shannon menegaskan kondisi one-time pad sebagai skema yang mencapai kerahasiaan sempurna.

Secara praktis, laju bahasa dan distribusi simbol memengaruhi efektivitas teknik analisis frekuensi terhadap cipher substitusi sederhana. Pada ranah modern, teori informasi juga mendasari batas kunci dan kebocoran informasi dalam protokol komunikasi aman. Tinjauan formal tersedia pada Menezes, Oorschot **and** Vanstone (1996).

2.1.1 Entropy

Entropi Shannon untuk variabel acak diskret X dengan distribusi $p(x)$ didefinisikan sebagai $H(X) = -\sum_x p(x) \log_2 p(x)$ (Shannon 1948). Dalam kriptografi, entropi kunci menunjukkan ukuran ruang pencarian efektif bagi penyerang brute-force. Entropi yang rendah menandakan prediktabilitas yang tinggi, mengurangi kekuatan efektif kunci atau sandi.

Estimasi entropi pada sumber dunia nyata harus memperhitungkan korelasi dan struktur; asumsi independensi sering menghasilkan estimasi berlebih. Lihat juga MacKay (2003) untuk pembahasan estimator dan pengaruhnya pada desain sumber entropi dan CSPRNG; standar evaluasi sumber entropi tersedia pada *Recommendation for the Entropy Sources Used for Random Bit Generation* (2018).

2.1.2 Laju Bahasa

Laju bahasa menggambarkan derajat redundansi dalam teks alami dan berdampak pada kerentanan cipher klasik terhadap analisis frekuensi. Bahasa dengan redundansi tinggi lebih mudah dianalisis karena pola kemunculan huruf yang khas tetap terpelihara setelah enkripsi substitusi sederhana. Oleh karena itu, teknik kriptanalisis klasik memanfaatkan statistik unigram dan n-gram untuk memulihkan kunci.

Pada sistem modern, pemodelan distribusi pesan masih relevan untuk mendesain skema padding, kompresi sebelum enkripsi, atau mekanisme perlindungan metadata. Diskusi awal tentang laju bahasa dan implikasinya dapat ditelusuri ke literatur teori informasi dasar (Menezes, Oorschot **and** Vanstone 1996).

2.2 Teori Bilangan

Teori bilangan menjadi tulang punggung banyak algoritma nirsimetri seperti RSA, Diffie–Hellman, dan ElGamal. Sifat pembagian, PBB, dan kombinasi lanjar penting untuk memahami algoritma Euclid dan perluasannya yang digunakan dalam perhitungan invers modulo. Konsep relatif prima dan aritmetika modulo memungkinkan definisi grup pada kelas residu, yang menjadi dasar analisis keamanan berdasarkan kesulitan faktorisasi dan logaritma diskrit.

Bilangan prima dan fungsi totient Euler memfasilitasi teorema Euler dan Fermat kecil, yang digunakan untuk menaikkan pangkat modulo secara efisien dan aman. Studi tentang akar primitif dan logaritma diskrit menyediakan landasan matematis untuk protokol pertukaran kunci modern. Referensi terbuka yang komprehensif dapat ditemukan pada Menezes, Oorschot **and** Vanstone (1996) **and** Boneh **and** Shoup (2020).

2.2.1 Sifat Pembagian Pada Bilangan Bulat

Sifat pembagian mendefinisikan relasi divisibilitas dan mendasari struktur ideal pada ring bilangan bulat. Teorema dasar aritmetika menjamin faktorisasi unik, yang menjadi kunci analisis banyak algoritma. Pada konteks kriptografi, divisibilitas berkaitan dengan pemilihan modulus dan struktur faktor untuk menjamin keamanan.

Pemahaman mengenai kongruensi dan residu diperlukan untuk mendesain operasi modulo yang benar dan efisien. Sumber terbuka menyediakan bukti dan contoh yang memadai untuk aplikasi dalam kriptografi nirsimetri (Menezes, Oorschot **and** Vanstone 1996).

2.2.2 Pembagi Bersama Terbesar (PBB)

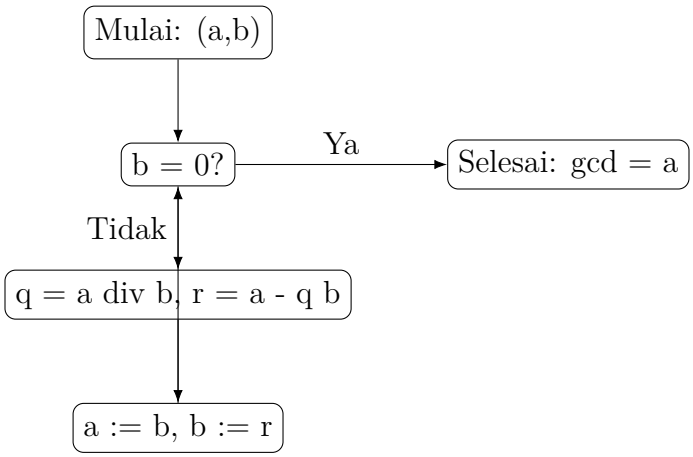
PBB antara dua bilangan dapat dihitung secara efisien dengan algoritma Euclid, memberikan landasan untuk menguji koprimalitas. Konsep ini penting dalam konstruksi kunci RSA, di mana pemilihan eksponen publik harus relatif prima terhadap fungsi totient dari modulus. Keterkaitan PBB dengan kombinasi lanjar memungkinkan pembuktian identitas serta konstruksi invers.

Dalam praktik, perhitungan PBB digunakan pada protokol untuk memastikan parameter valid dan mencegah kelemahan struktural.

Analisis rinci tersedia pada Menezes, Oorschot **and** Vanstone (1996).

2.2.3 Algoritma Euclidean

Algoritma Euclid menghitung PBB dengan proses iteratif berbasis pembagian berulang, dan dapat diperluas (extended Euclid) untuk menemukan koefisien kombinasi lanjar serta invers modulo. Efisiensinya menjadikannya komponen fundamental dalam aritmetika modulo. Lihat Shoup (2009) untuk analisis komputasional.



Gambar 2.1: Alur algoritma Euclid untuk menghitung PBB (Shoup 2009).

2.2.4 Kombinasi Lanjar

Identitas Bézout menyatakan bahwa terdapat kombinasi lanjar dari dua bilangan bulat yang menghasilkan PBB-nya. Fakta ini menyediakan konstruksi langsung untuk invers modulo ketika PBB bernilai satu. Dalam kriptografi, kemampuan menghitung kombinasi lanjar secara efisien memfasilitasi banyak primitif.

Aplikasi praktis meliputi penyelesaian sistem kongruensi linear dan pembuktian sifat-sifat yang diperlukan dalam protokol kunci publik. Bahasan lengkap dapat ditemukan pada Menezes, Oorschot **and** Vanstone (1996).

2.2.5 Relatif Prima

Relatif prima antara bilangan memastikan keberadaan invers modulo, prasyarat untuk banyak operasi kriptografi. Dalam RSA, pemilihan eksponen publik yang relatif prima terhadap $\varphi(n)$ menjamin dapat dihitungnya eksponen privat. Dalam skema berbasis grup siklik, koprimalitas berkaitan dengan struktur orde elemen.

Pengujian relatif prima dalam skala besar dilakukan melalui perhitungan PBB yang efisien, yang secara praktis tidak menjadi bottleneck dalam pembangkitan kunci. Rujukan mendalam disediakan oleh Menezes, Oorschot **and** Vanstone (1996).

2.2.6 Aritmatika Modulo

Aritmatika modulo mendefinisikan operasi pada kelas residu yang tertutup dan memiliki sifat-sifat yang dapat dianalisis secara aljabar. Operasi ini mendasari semua komputasi nirsimetri, dari perpangkatan modular hingga operasi pada kurva eliptik. Implementasi yang aman memerlukan kewaspadaan terhadap side-channel seperti timing.

Optimalisasi seperti Montgomery reduction digunakan untuk mempercepat perpangkatan modular pada modulus besar. Penjelasan rinci dapat ditemukan pada literatur standar (Menezes, Oorschot **and** Vanstone 1996).

2.2.7 Bilangan Prima

Bilangan prima adalah blok bangunan bagi banyak sistem kriptografi, khususnya pada RSA dan skema logaritma diskrit. Pengujian primalitas probabilistik seperti Miller–Rabin memungkinkan penentuan prima yang efisien untuk ukuran kunci modern. Distribusi bilangan prima memengaruhi keamanan faktorisasi.

Teknik pembangkitan prima aman mempertimbangkan struktur khusus agar tidak memperkenalkan kelemahan, seperti prima dengan faktor $p - 1$ yang halus. Referensi terbuka tersedia pada literatur standar (Menezes, Oorschot **and** Vanstone 1996).

2.2.8 Fungsi Totient Euler

Fungsi totient Euler $\varphi(n)$ menghitung jumlah bilangan antara 1 dan n yang koprima terhadap n, memainkan peran sentral pada teorema Euler. Pada RSA, $\varphi(n)$ menentukan hubungan antara eksponen publik dan privat. Perhitungan φ untuk modulus komposit setara sulitnya dengan faktorisasi.

Pemilihan parameter yang tepat memastikan bahwa $\varphi(n)$ tidak diketahui pihak luar dan mencegah serangan aritmetika. Lihat Menezes, Oorschot **and** Vanstone (1996) untuk detail.

2.2.9 Teorema Euler

Teorema Euler menyatakan bahwa untuk a yang koprima dengan n, berlaku $a^{\varphi(n)} \equiv 1 \pmod{n}$. Hasil ini menggeneralisasi teorema Fermat kecil dan digunakan secara luas dalam rancangan dan analisis algoritma kriptografi. Perpangkatan modular yang efisien dibangun di atas teorema ini dan variasinya.

Dalam praktik, penerapan yang aman memerlukan pemilihan basis dan modulus dengan sifat yang sesuai untuk mencegah kebocoran informasi. Uraian formal ada pada Menezes, Oorschot **and** Vanstone (1996).

2.2.10 Akar Primitif dan Logaritma Diskrit

Akar primitif adalah generator dari grup siklik modulo bilangan prima, dan logaritma diskrit mendefinisikan masalah komputasi yang sulit pada grup tersebut. Kesulitan DLP mendasari keamanan Diffie–Hellman klasik dan ElGamal. Pemilihan grup yang aman menghindari struktur yang memudahkan serangan index calculus.

Perkembangan modern mencakup grup eliptik yang menawarkan parameter keamanan lebih baik per bit dan ketahanan terhadap beberapa serangan klasik. Pengantar ringkas tersedia pada literatur standar (Menezes, Oorschot **and** Vanstone 1996; Boneh **and** Shoup 2020).

2.3 Aljabar Abstrak

Aljabar abstrak menyediakan kerangka untuk memodelkan struktur aljabar seperti grup, ring, dan medan yang menjadi basis banyak konstruksi kriptografi. Grup menangkap struktur operasi tunggal yang tertutup, sedangkan ring dan medan memperluas ke dua operasi yang kompatibel. Medan berhingga dan medan Galois sangat penting untuk desain cipher blok dan kode-kode koreksi kesalahan.

Operasi pada polinom di medan berhingga digunakan dalam konstruksi S-box, LFSR, dan mode operasi tertentu. Referensi yang dapat diakses luas untuk aspek ini dapat ditemukan di Menezes, Oorschot **and** Vanstone (1996).

2.3.1 Grup

Grup adalah himpunan dengan satu operasi biner yang tertutup, asosiatif, memiliki elemen identitas, dan setiap elemen memiliki invers. Dalam kriptografi, grup siklik digunakan untuk membangun masalah komputasi sulit seperti DLP. Struktur grup menentukan keberlakuan teorema yang dimanfaatkan dalam keamanan.

Pemilihan grup yang tepat menghindari orde yang rentan terhadap serangan, misalnya grup dengan faktor-faktor kecil yang memudahkan algoritma pohlig–hellman. Penjelasan

lengkap tersedia pada Menezes, Oorschot **and** Vanstone (1996).

2.3.2 Ring

Ring memperkenalkan dua operasi biner dengan sifat distributif, dan muncul dalam aritmetika polinom yang digunakan pada beberapa cipher dan skema kunci publik berbasis kisi. Struktur ring memungkinkan efisiensi komputasi, tetapi juga memerlukan kehati-hatian dalam analisis keamanan.

Skema modern seperti RLWE memanfaatkan ring polinomial dengan modul tertentu untuk mencapai efisiensi sambil mempertahankan asumsi sulit. Tinjauan dasar tersedia pada Menezes, Oorschot **and** Vanstone (1996).

2.3.3 Medan

Medan adalah ring komutatif dengan setiap elemen tak nol memiliki invers, menyediakan ruang aljabar yang kaya untuk konstruksi kriptografi. Medan riil dan kompleks jarang digunakan langsung; yang lebih relevan adalah medan berhingga karena sifat diskritnya.

Operasi dasar pada medan menjadi inti desain S-box dan transformasi linier pada cipher blok modern. Referensi dapat ditemukan pada Menezes, Oorschot **and** Vanstone (1996).

2.3.4 Medan Berhingga

Medan berhingga $\mathbb{F}_q$ dengan $q = p^m$ menyediakan struktur yang kompatibel untuk operasi polinom dan vektor. Representasi elemen, pemilihan polinom primitif, dan implementasi operasi efisien menjadi pertimbangan utama dalam desain kriptografi.

Medan berhingga digunakan secara luas pada AES dan LFSR, dengan properti yang dapat dianalisis secara matematis untuk menetapkan keamanan dan difusi. Pengantar yang dapat diakses tersedia pada Menezes, Oorschot **and** Vanstone (1996).

2.3.5 Medan Galois

Medan Galois adalah medan berhingga yang memiliki struktur automorfisme yang dipahami dengan baik, memungkinkan analisis sifat-sifat yang berguna untuk kriptografi. Pada AES, operasi dilakukan pada $\mathbb{F}_{2^8}$ dengan polinom irreducible tertentu untuk membangun S-box yang aman.

Analisis keamanan S-box melibatkan kriteria nonlinieritas dan resistensi diferensial, yang dapat diformalkan dalam kerangka medan Galois. Rujukan umum dapat ditemukan pada Menezes, Oorschot **and** Vanstone (1996).

2.3.6 Aritmatika Polinom di dalam Medan Galois

Aritmatika polinom memungkinkan konstruksi dan manipulasi elemen medan berhingga, termasuk operasi modulo polinom irreducible. Teknik ini digunakan untuk membangun S-box, LFSR, dan kode koreksi kesalahan.

Efisiensi implementasi sangat bergantung pada representasi dan pilihan polinom; literatur terbuka menyediakan panduan praktis untuk implementasi aman pada berbagai platform (Menezes, Oorschot **and** Vanstone 1996).

2.3.7 Contoh Kode: Algoritma Euclid Terekstensi

Berikut implementasi ringkas untuk menghitung PBB dan koefisien Bézout serta invers modulo bila ada.

Listing 2.1: Extended Euclid dan invers modulo

```
from typing import Tuple

def extended_gcd(a: int, b: int) -> Tuple[int, int, int]:
    old_r, r = a, b
    old_s, s = 1, 0
    old_t, t = 0, 1
    while r != 0:
        q = old_r // r
        old_r, r = r, old_r - q * r
        old_s, s = s, old_s - q * s
        old_t, t = t, old_t - q * t
    return old_r, old_s, old_t  # gcd, x, y such that ax + by =
       gcd

def mod_inverse(a: int, n: int) -> int | None:
    g, x, _ = extended_gcd(a, n)
    if g != 1:
        return None
    return x % n

if __name__ == "__main__":
    g, x, y = extended_gcd(240, 46)
    print(g, x, y)
    print(mod_inverse(17, 3120))
```

Bab 3

Serangan Terhadap Kriptografi

3.1 Serangan

Serangan terhadap sistem kriptografi mencakup pendekatan yang menargetkan algoritma inti, protokol, implementasi, serta ekosistem kunci. Klasifikasi umum meliputi serangan kriptanalisis yang mengeksploitasi struktur matematis, brute-force yang mengandalkan pencarian menyeluruh, dan serangan berbasis kanal samping yang memanfaatkan kebocoran fisik. Pemahaman spektrum serangan membantu merumuskan model ancaman yang realistis.

Pemilihan parameter yang tepat, penerapan mode operasi yang benar, dan mitigasi kanal samping adalah langkah kunci untuk memperkuat sistem. Tinjauan konseptual tersedia pada Menezes, Oorschot **and** Vanstone (1996) dan panduan praktik terbaik terdapat di standar terbuka NIST.

3.1.1 Serangan Kriptanalisis

Kriptanalisis menganalisis kelemahan struktur algoritma untuk mengurangi kompleksitas pemulihan kunci atau plainteks. Teknik klasik termasuk analisis frekuensi; teknik modern meliputi serangan diferensial dan linear terhadap cipher blok. Keberhasilan serangan sering kali bergantung pada ketersediaan data dan kesesuaian model serangan dengan skenario nyata.

Penelitian seminal tentang kriptanalisis diferensial dan linear memberikan landasan untuk evaluasi cipher modern dan desain yang tahan serangan (Biham **and** Shamir 1991; Matsui 1993).

3.1.2 Brute-Force Attack dan Analytical Attack

Brute-force menyerang dengan menjelajahi ruang kunci secara menyeluruh, sehingga keamanan efektif sangat ditentukan oleh entropi kunci. Analytical attack mencari struktur

yang mengurangi kompleksitas pencarian melalui sifat matematis atau statistik. Desain parameter harus memastikan bahwa kompleksitas terendah yang diketahui tetap di luar jangkauan praktis.

Standar ukuran kunci dan pedoman keamanan membantu menentukan tingkat kekuatan yang memadai untuk horizon waktu tertentu. Diskusi dapat ditemukan di sumber terbuka seperti publikasi NIST.

3.1.3 Passive Attack dan Active Attack

Serangan pasif bertujuan menyadap informasi tanpa mengubah komunikasi, sedangkan serangan aktif memodifikasi atau menyuntikkan pesan untuk memperoleh keuntungan. Model keamanan yang kuat harus mengantisipasi kedua jenis serangan dan menyediakan deteksi serta mitigasi yang tepat.

Protokol modern memasukkan mekanisme otentikasi, integritas, dan anti-replay untuk melawan serangan aktif, sementara enkripsi melindungi dari serangan pasif. Contoh desain dapat dilihat pada spesifikasi TLS (Rescorla 2018).

3.2 Keamanan Algoritma Kriptografi

Keamanan algoritma dinilai berdasarkan bukti reduksi, analisis kriptanalisis publik, serta rekam jejak implementasi. Definisi formal seperti IND-CPA dan IND-CCA menetapkan target yang harus dicapai dalam model serangan tertentu. Evaluasi berkelanjutan oleh komunitas menjadi indikator keandalan suatu algoritma.

Praktik yang baik meliputi penggunaan standar yang diakui dan perpustakaan yang telah diaudit, dengan konfigurasi yang mengikuti rekomendasi terkini. Referensi terbuka mencakup standar NIST dan IETF.

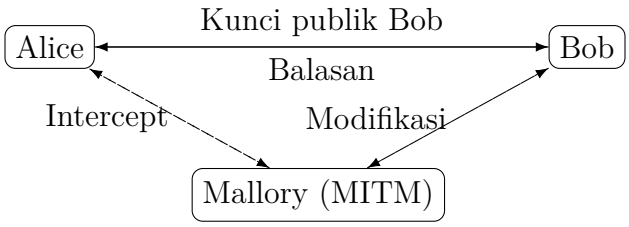
3.3 Kompleksitas Serangan

Kompleksitas serangan mengukur sumber daya yang dibutuhkan untuk menembus skema, termasuk waktu, memori, dan data. Notasi asimtotik memberikan gambaran tren, tetapi penilaian praktis memerlukan perkiraan biaya perangkat keras dan skala data. Evaluasi juga mempertimbangkan kemajuan algoritmik dan perangkat keras di masa depan.

Analisis kompleksitas harus direvisi secara periodik seiring berkembangnya teknik dan teknologi. Diskusi ringkas tersedia dalam literatur terbuka dan ringkasan standar keamanan kunci oleh NIST.

Tabel 3.1: Klasifikasi ringkas beberapa serangan dan mitigasi

Kategori	Contoh	Mitigasi
Kriptanalisis	Linear/diferensial	Desain cipher teruji, parameter an
Brute-force	Pencarian kunci	Ukuran kunci memadai, rate limit
Kanal samping	Timing/cache AES (Bernstein 2005)	Konstanta waktu, masking, blindi
Protokol	Lucky Thirteen (AlFardan **and** Paterson 2013)	AEAD modern, verifikasi konstan,



Gambar 3.1: Skema sederhana serangan Man-in-the-Middle (MITM).

3.3.1 Contoh Kode: Brute-Force Caesar Cipher

Berikut ilustrasi brute-force pada sandi Caesar untuk memulihkan pergeseran dengan eksplorasi seluruh ruang kunci.

Listing 3.1: Brute-force Caesar

```
import string

def caesar_decrypt(ciphertext: str, shift: int) -> str:
    alphabet = string.ascii_lowercase
    result = []
    for ch in ciphertext.lower():
        if ch in alphabet:
            idx = (alphabet.index(ch) - shift) % 26
            result.append(alphabet[idx])
        else:
            result.append(ch)
    return ''.join(result)

def brute_force(ciphertext: str):
    for s in range(26):
        print(s, caesar_decrypt(ciphertext, s))

if __name__ == "__main__":
    brute_force("khoor zruog")
```

Bab 4

Kriptografi Klasik

4.1 Cipher Substitusi

Cipher substitusi mengganti setiap simbol plainteks dengan simbol lain menurut aturan yang ditetapkan oleh kunci. Contoh paling sederhana adalah Caesar cipher, yang menggeser alfabet dengan jarak tetap. Struktur statistik bahasa yang tetap terlihat setelah substitusi sederhana menjadikan cipher ini rentan terhadap analisis frekuensi.

Kekuatan cipher substitusi meningkat dengan memperluas ruang kunci atau menggunakan variasi polialfabetik, tetapi tetap inferior terhadap standar modern. Perbandingan sistematis dan analisis historis tersedia dalam sumber terbuka (Menezes, Oorschot **and** Vanstone 1996).

4.1.1 Caesar Cipher

Caesar cipher menerapkan pergeseran tetap pada alfabet, sehingga mudah diterapkan namun lemah terhadap analisis frekuensi. Penyerang yang mengamati distribusi huruf dapat menebak pergeseran dengan menghitung kecocokan pola. Karena ruang kuncinya kecil, brute-force juga efektif.

Penggunaan Caesar cipher hanya relevan untuk tujuan edukasi dan demonstrasi konsep dasar. Literatur pengantar kriptografi membahas teknik pemulihan kunci dan variasi sederhana (Menezes, Oorschot **and** Vanstone 1996).

4.1.2 Kriptanalisis Terhadap Caesar Cipher

Kriptanalisis terhadap cipher ini memanfaatkan statistik unigram dan bigram untuk mengidentifikasi pergeseran yang paling mungkin. Prosedur ini dapat diotomasi dengan menghitung skor kecocokan frekuensi. Dengan sedikit data, tingkat keberhasilannya sudah tinggi.

Metodologi ini mengilustrasikan prinsip umum bahwa redundansi bahasa melemahkan

substitusi sederhana, memotivasi desain polialfabetik dan mekanisme difusi yang lebih baik. Pembahasan dapat ditemukan pada Menezes, Oorschot **and** Vanstone (1996).

4.2 Jenis-jenis Cipher Substitusi

Cipher abjad-tunggal menggunakan satu pemetaan tetap untuk seluruh pesan, memudahkan analisis frekuensi. Cipher abjad-banyak memperkenalkan variasi pemetaan sepanjang pesan, menyulitkan analisis tetapi tetap menunjukkan pola yang dapat dieksploitasi. Substitusi homofonik dan poligram menambah kerumitan dengan memetakan satu simbol ke banyak simbol atau unit poligram.

Meskipun lebih kuat dari substitusi sederhana, pendekatan ini masih tidak sebanding dengan cipher modern yang dirancang dengan teori keamanan formal. Penjelasan dan contoh tersedia dalam literatur standar (Menezes, Oorschot **and** Vanstone 1996).

4.3 Cipher Transposisi

Cipher transposisi menyusun ulang posisi karakter tanpa mengubah identitas simbol, menghasilkan keluaran yang mempertahankan histogram tetapi mengubah urutan. Keamanan bergantung pada kerumitan aturan penyusunan ulang dan panjang kunci. Analisis sering memanfaatkan pola periodik yang muncul dari struktur transposisi.

Transposisi sering dikombinasikan dengan substitusi untuk meningkatkan kekuatan, namun tetap rentan terhadap serangan yang memanfaatkan statistik tingkat lebih tinggi. Uraian dapat ditemukan pada Menezes, Oorschot **and** Vanstone (1996).

Kunci kolom: 3 1 4 2

P	E	S	A
N	B	A	R
A	H	A	S

Baca per kolom urut kunci: 1,2,3,4

Gambar 4.1: Skema sederhana transposisi kolumnar.

4.4 Super Enkripsi

Super enkripsi menggabungkan beberapa teknik enkripsi, misalnya substitusi diikuti transposisi, untuk meningkatkan keamanan. Kombinasi yang tepat dapat menambah

difusi dan konfusi, tetapi tidak menggantikan kebutuhan akan desain modern.

Praktik historis ini memberikan wawasan tentang prinsip desain yang kelak diformalkan pada cipher blok modern. Tinjauan konsep tersedia pada Menezes, Oorschot **and** Vanstone (1996).

4.5 Metode Analisis Frekuensi

Analisis frekuensi mengeksploitasi distribusi simbol bahasa untuk memecahkan cipher substitusi. Teknik ini mengestimasi pergeseran atau pemetaan dengan mencocokkan histogram cipherteks dengan histogram bahasa sumber. Variasi metode menggunakan n-gram untuk meningkatkan akurasi.

Keterbatasan teknik ini mendorong penggunaan metode polialfabetik dan mekanisme difusi yang menghapus pola statistik.

Diskusi rinci tersedia pada Menezes, Oorschot **and** Vanstone (1996).

4.6 Vigenere Cipher

Vigenère cipher menggunakan kunci berulang untuk memilih pergeseran yang berbeda pada setiap posisi, sehingga mengurangi pola sederhana. Namun, periodisitas kunci dapat diungkap melalui metode Kasiski atau indeks koinsidensi. Setelah panjang kunci diketahui, pesan dapat dipecahkan sebagai serangkaian Caesar cipher.

Variasi Vigenère memperkenalkan tabel atau alfabet yang dimodifikasi, tetapi inti kelemahan periodisitas tetap ada. Rujukan tersedia di Menezes, Oorschot **and** Vanstone (1996).

4.6.1 Metode Kasiski untuk menentukan panjang kunci vigenere Cipher

Metode Kasiski mengidentifikasi pengulangan pola dalam cipherteks untuk memperkirakan panjang kunci. Dengan menghitung jarak antar pengulangan dan mencari faktor-faktornya, estimasi panjang kunci dapat diperoleh. Pendekatan ini efektif pada teks cukup panjang dengan kunci berulang.

Setelah panjang kunci diperoleh, analisis frekuensi per-posisi memungkinkan pemulihan kunci dan plainteks. Ringkasan metode ini tersedia pada Menezes, Oorschot **and** Vanstone (1996).

Tabel 4.1: Perbandingan ringkas beberapa cipher klasik

Cipher	Inti operasi	Catatan keamanan
Caesar	Pergeseran abjad tetap	Ruang kunci kecil, mudah brute-force
Vigenère	Pergeseran polialfabetik	Periodisitas diekspos Kasiski/IoC (Friedman 1922)
Transposisi	Penyusunan ulang posisi	Histogram terjaga, pola periodik
Playfair	Substitusi digraf	Rentan analisis digraf
Affine	Transformasi linear modulo	Linearitas mudah dibalik
Hill	Matriks linear per blok	Linear, butuh data cukup

4.6.2 Variasi Vigenere Cipher

Variasi Vigenère mencakup penggunaan alfabet yang berubah-ubah atau kunci yang tidak berulang untuk mengurangi periodisitas. Meskipun meningkatkan kerumitan, variasi tersebut biasanya masih kalah dari cipher modern. Fokus pendidikan tetap pada pemahaman konsep konfusi dan difusi.

Sumber pengantar membahas kompromi antara kompleksitas dan keamanan dalam pendekatan ini pada literatur standar (Menezes, Oorschot **and** Vanstone 1996).

4.7 Playfair Cipher

Playfair cipher mengenkripsi digraf, sehingga memutus pola unigram tetapi masih menyisakan struktur yang dapat dianalisis. Matriks kunci 5x5 menentukan substitusi pasangan huruf dengan aturan khusus. Kelemahannya termasuk sensitivitas terhadap struktur bahasa dan keterbatasan ruang kunci.

Teknik kriptanalisis terhadap Playfair mengeksploitasi statistik digraf dan aturan transformasi. Uraian dapat ditemukan pada Menezes, Oorschot **and** Vanstone (1996).

4.8 Affine Cipher

Affine cipher menerapkan transformasi linear pada indeks huruf modulo ukuran alfabet, menggabungkan skala dan pergeseran. Keamanan lemah karena struktur linear yang mudah dibalik dengan sedikit pasangan plainteks-cipherteks. Ruang kunci terbatas memudahkan brute-force.

Analisis terhadap affine cipher memberikan ilustrasi penting mengenai kelemahan struktur linear dalam kriptografi klasik. Rujukan tersedia pada Menezes, Oorschot **and** Vanstone (1996).

4.9 Hill Cipher

Hill cipher mengoperasikan blok huruf menggunakan matriks invertibel modulo ukuran alfabet, memperkenalkan difusi melalui operasi linear multivariabel. Kelemahan muncul ketika ukuran blok kecil dan data cukup untuk menyelesaikan sistem linear. Keamanan tidak memadai untuk standar modern.

Diskusi Hill cipher membantu memahami pentingnya nonlinieritas pada desain cipher modern. Lihat Menezes, Oorschot **and** Vanstone (1996).

4.10 Enigma Cipher

Mesin Enigma menggunakan rotor untuk menerapkan substitusi yang berubah-ubah per posisi, menghasilkan pola kompleks. Namun, kelemahan desain seperti *no letter encrypts to itself* dan prosedur operasional memungkinkan kriptanalisis yang sukses. Analisis historis menunjukkan pentingnya prosedur keamanan selain kekuatan algoritma.

Sumber terbuka mengenai Enigma dan kriptanalisisnya memberikan wawasan sejarah dan teknik (Menezes, Oorschot **and** Vanstone 1996).

4.11 One-Time Pad

One-time pad mencapai keamanan sempurna ketika kunci acak benar-benar seragam, sepanjang pesan, dan digunakan sekali. Tantangan praktis mencakup distribusi dan manajemen kunci dalam skala besar. Karena persyaratan ini, OTP jarang digunakan di luar konteks khusus.

OTP mengilustrasikan batas ideal keamanan dan memotivasi pendekatan yang mencapai keamanan komputasional dengan efisiensi lebih baik. Penjelasan formal tersedia pada Menezes, Oorschot **and** Vanstone (1996).

4.11.1 Contoh Kode: Vigenère, Kasiski, dan Index of Coincidence

Berikut contoh Python untuk menduga panjang kunci dengan Kasiski dan IoC, serta mendekripsi Vigenère.

Listing 4.1: Vigenère: Kasiski dan IoC

```
import re, collections, string

def kasiski(ciphertext: str, min_len: int = 3):
    repeats = collections.defaultdict(list)
    text = re.sub('[^A-Za-z]', '', ciphertext.upper())
```

```
    for L in range(min_len, min_len+3):
        for i in range(len(text) - L + 1):
            sub = text[i:i+L]
            for j in range(i+1, len(text) - L + 1):
                if text[j:j+L] == sub:
                    repeats[sub].append(j - i)
    distances = []
    for dists in repeats.values():
        distances.extend(dists)
    return distances

def gcd(a, b):
    while b:
        a, b = b, a % b
    return a

def guess_key_lengths(distances):
    from math import gcd as mgcd
    if not distances:
        return []
    g = distances[0]
    for d in distances[1:]:
        g = mgcd(g, d)
    return [g] if g > 1 else []

def index_of_coincidence(text: str):
    text = re.sub('[^A-Za-z]', '', text.upper())
    N = len(text)
    counts = collections.Counter(text)
    return sum(c*(c-1) for c in counts.values()) / (N*(N-1) or 1)

def vigenere_decrypt(ct: str, key: str) -> str:
    alphabet = string.ascii_uppercase
    pt = []
    k = key.upper()
    ki = 0
    for ch in ct.upper():
        if ch in alphabet:
            p = (alphabet.index(ch) - alphabet.index(k[ki % len(k
                )])) % 26
            pt.append(alphabet[p])
```

```
            ki += 1
        else:
            pt.append(ch)
    return ''.join(pt)

if __name__ == '__main__':
    ct = 'LXFOPVEFRNHR'  # contoh untuk 'ATTACKATDAWN' dengan
       kunci 'LEMON'
    dists = kasiski(ct)
    print('Kasiski distances:', dists)
    print('IoC:', index_of_coincidence(ct))
    print(vigenere_decrypt(ct, 'LEMON'))
```

Bahan latar tersedia pada Practical Cryptography (2020), Friedman (1922) **and** Houtven (2016).

Bab 5

Kriptografi Modern

5.1 Rangkaian Bit dan Operasinya

Kriptografi modern beroperasi pada bit dan blok data, memanfaatkan operasi boolean dan aritmetika modular. Representasi bit memungkinkan desain yang efisien dan analisis mendalam terhadap difusi dan konfusi dalam cipher. Pemahaman operasi dasar ini menjadi landasan untuk menilai implementasi dan optimisasi.

Operasi XOR, rotasi, dan perpindahan bit sering digunakan untuk membangun primitif yang cepat dan sederhana. Kombinasi operasi ini yang tepat membantu mencapai properti keamanan yang diinginkan pada cipher modern.

5.2 algoritma enkripsi dengan XOR sederhana

Enkripsi berbasis XOR dengan keystream adalah komponen inti cipher alir, di mana keamanan sepenuhnya bergantung pada keacakan keystream. Jika keystream dapat diprediksi atau digunakan berulang, kebocoran terjadi sehingga plainteks dapat dipulihkan. Oleh karena itu, keystream harus dibangkitkan dari sumber yang kuat dan unik per pesan.

Skema yang aman menggunakan nonce dan kunci untuk mencegah reuse, serta konstruksi yang terbukti aman untuk menghasilkan keystream. Standar AEAD mengintegrasikan enkripsi dan autentikasi untuk mitigasi serangan manipulasi (Dworkin 2008).

5.3 Kategori Cipher untuk Data Digital

Cipher modern terbagi menjadi cipher alir dan cipher blok, masing-masing dengan keunggulan dan tantangan berbeda. Cipher alir menawarkan latensi rendah dan granularitas bit, sedangkan cipher blok bekerja pada blok tetap dengan mode operasi yang menyediakan fleksibilitas. Pemilihan kategori bergantung pada kebutuhan aplikasi dan lingkungan eksekusi.

Standar seperti AES untuk cipher blok dan ChaCha20 untuk cipher alir digunakan luas karena analisis keamanan yang ekstensif dan kinerja yang baik pada perangkat modern.

5.4 Cipher alir

Cipher alir menghasilkan keystream yang dikombinasikan dengan plainteks menggunakan XOR, membutuhkan sumber entropi yang kuat dan parameter unik per pesan. Desain harus mencegah korelasi dan siklus pendek pada generator internal. Implementasi harus menangani inisialisasi dan pemutakhiran keadaan secara aman.

Contoh yang banyak dipakai adalah ChaCha20 dengan Poly1305 sebagai MAC dalam konfigurasi AEAD, yang memberikan keamanan dan kinerja baik pada perangkat lunak modern (Langley, Nir **and** Benjamin 2018).

5.5 Pembangkit kunci alir

Pembangkit keystream sering menggunakan register geser dengan umpan balik dan komponen nonlinier untuk menghindari prediktabilitas. Struktur harus dianalisis terhadap serangan korelasi dan algebraik untuk menjamin keamanan. Inisialisasi dengan kunci dan nonce harus dirancang untuk mencegah state recovery.

Literatur terbuka menyediakan analisis konstruksi yang kuat dan pedoman implementasi untuk mencegah kelemahan yang diketahui.

5.6 Linear Feedback Shift Register (LFSR)

LFSR menyediakan struktur linear efisien yang sering menjadi komponen dalam pembangkit keystream. Karena linearitasnya, LFSR sendiri tidak aman dan memerlukan kombinasi nonlinier untuk menghindari serangan linier. Pemilihan polinom umpan balik menentukan periode dan distribusi keluaran.

Analisis LFSR dan kombinasinya terdokumentasi luas dalam literatur kriptografi dan komunikasi digital (Menezes, Oorschot **and** Vanstone 1996).

5.7 Serangan Terhadap Cipher Alir

Serangan terhadap cipher alir mencakup serangan korelasi, algebraik, dan state recovery, yang mengeksploitasi struktur internal pembangkit. Kesalahan implementasi seperti reuse nonce atau kunci membuka peluang serangan praktis yang menghancurkan keamanan. Evaluasi keamanan harus meliputi model data yang realistis dan asumsi penyerang yang kuat.

Pedoman standar merekomendasikan penggunaan konstruksi AEAD modern untuk mencegah manipulasi dan memastikan integritas, selain kerahasiaan (Dworkin 2008).

5.8 Cipher Blok

Cipher blok memproses data dalam blok tetap menggunakan struktur seperti jaringan Feistel atau substitusi-permutasi. Desain yang baik mencapai difusi dan konfusi yang kuat, dinilai melalui analisis diferensial dan linear. Keamanan juga ditentukan oleh ukuran blok dan parameter internal.

AES sebagai standar de facto menawarkan keamanan kuat dengan kinerja tinggi pada perangkat keras dan lunak, didukung bukti dan analisis luas (*Advanced Encryption Standard (AES)* 2001).

Tabel 5.1: Ringkasan beberapa mode operasi cipher blok

Mode	Difusi antarblok	Keacakan IV/Nonce	Catatan
ECB	Tidak	-	Bocor pola, hindari
CBC	Ya	IV acak	Perlu autentikasi (hindari padding oracle) (*Recomm*
CTR	-	Nonce/counter unik	Paralelis, butuh keunikan ketat (*Recommendation fo*
GCM	Ya	Nonce unik	AEAD: kerahasiaan+integritas (Dworkin 2007)

5.9 Electronic Code Book (ECB)

ECB mengenkripsi setiap blok secara independen, mempertahankan kesamaan blok identik sehingga membocorkan pola. Mode ini tidak direkomendasikan untuk data dengan struktur, kecuali pada konteks sangat terbatas atau untuk keperluan determinisme yang dikontrol.

Sebagian besar aplikasi seharusnya menggunakan mode yang menyediakan difusi antarblok dan/atau autentikasi untuk mencegah manipulasi. Referensi dan pedoman terdapat pada publikasi NIST.

5.10 Cipher Block Chaining (CBC)

CBC menggabungkan blok sebelumnya dengan plainteks saat ini sebelum enkripsi, memberikan difusi yang lebih baik dibanding ECB. Keamanan membutuhkan IV acak tak dapat diprediksi dan mekanisme autentikasi terpisah untuk mencegah serangan padding oracle.

Standar merekomendasikan penggunaan AEAD untuk menghindari kebutuhan akan MAC terpisah dan mitigasi kelas serangan ini (Dworkin 2007).

5.11 Cipher Feedback (CFB)

CFB memperlakukan cipher blok sebagai pembangkit keystream dengan cara memproses blok keluaran untuk XOR dengan plainteks. Mode ini useful untuk aplikasi streaming dengan panjang tidak kelipatan blok. Namun, integritas tetap memerlukan mekanisme autentikasi terpisah.

Pemilihan ukuran segmen memengaruhi latensi dan kinerja, yang harus disesuaikan dengan kebutuhan aplikasi.

5.12 Output Feedback (OFB)

OFB menghasilkan keystream independen dari plainteks sehingga cocok untuk saluran dengan error. Karena deterministik untuk IV yang sama, IV harus unik mutlak untuk mencegah reuse keystream. OFB tidak menyediakan autentikasi secara bawaan.

Kombinasi dengan MAC seperti HMAC dapat menyediakan integritas, tetapi AEAD sering menjadi pilihan yang lebih sederhana.

5.13 Counter Mode

CTR mengubah cipher blok menjadi pembangkit keystream dengan mengenkripsi nilai counter yang meningkat. Mode ini sangat paralelis dan memiliki kinerja tinggi, tetapi menuntut nonce/counter unik per kunci. Reuse counter menyebabkan kebocoran fatal melalui XOR dari plainteks.

Dalam praktik, CTR sering digunakan sebagai bagian dari GCM yang juga menyediakan autentikasi, seperti dijelaskan pada Dworkin (2007).

5.14 Prinsip-prinsip Perancangan Cipher Blok

Prinsip desain meliputi konfusi dan difusi Shannon, jumlah ronde yang memadai, nonlinieritas kuat pada S-box, dan struktur yang menghambat kriptanalisis diferensial dan linear. Evaluasi melibatkan metrik seperti nonlinieritas, uniformitas diferensial, dan campuran linear.

Pendekatan sistematis membantu menyeimbangkan keamanan dan kinerja sambil menghindari jebakan desain yang umum. Diskusi dapat ditemukan pada literatur terbuka.

5.14.1 Prinsip Confusion dan Diffusion dari Shannon

Konfusi mengaburkan hubungan antara kunci dan cipherteks, sementara difusi menyebarkan pengaruh satu bit ke banyak bit keluaran. Keduanya bekerja bersama untuk

mencegah inferensi langsung terhadap kunci dari observasi cipherteks. Struktur berulang memungkinkan akumulasi efek ini sepanjang ronde.

Analisis formal menggunakan teknik statistik dan kombinatorial untuk menilai kekuatan properti tersebut pada desain konkret.

5.14.2 Cipher Berulang

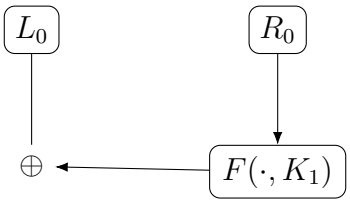
Cipher berulang menerapkan fungsi ronde berkali-kali dengan subkunci berbeda. Pendekatan ini memudahkan desain modular dan analisis keamanan. Pemilihan jumlah ronde merupakan kompromi antara kinerja dan margin keamanan.

Studi empiris menunjukkan bahwa ronde yang terlalu sedikit membuka peluang serangan yang efektif; oleh karena itu margin keamanan harus konservatif.

5.14.3 Jaringan Feistel

Jaringan Feistel membagi blok menjadi dua bagian dan menerapkan fungsi ronde pada satu bagian dengan kunci ronde, kemudian menukar bagian. Properti dapat dibalik dengan mudah, sehingga memudahkan implementasi enkripsi dan dekripsi menggunakan fungsi yang sama. Banyak cipher klasik seperti DES menggunakan struktur ini.

Analisis jaringan Feistel menilai properti avalanche dan resistensi terhadap serangan standar untuk memastikan keamanan memadai.



$$L_1 = R_0 \qquad R_1 = L_0 \oplus F(R_0, K_1)$$

Gambar 5.1: Skema satu ronde jaringan Feistel.

5.14.4 Contoh Kode: AES-CTR

Berikut contoh ringkas enkripsi/dekripsi AES-CTR menggunakan pustaka Python `cryptography` (The cryptography developers 2024d).

Listing 5.1: AES-CTR dengan cryptography

```
from cryptography.hazmat.primitives.ciphers import Cipher,
    algorithms, modes
from cryptography.hazmat.backends import default_backend
import os

key = os.urandom(32)    # AES-256
nonce = os.urandom(16) # 128-bit counter initial value
cipher = Cipher(algorithms.AES(key), modes.CTR(nonce), backend=
    default_backend())
encryptor = cipher.encryptor()
ct = encryptor.update(b"pesan rahasia") + encryptor.finalize()

decryptor = cipher.decryptor()
pt = decryptor.update(ct) + decryptor.finalize()
assert pt == b"pesan rahasia"
```

5.14.5 Kotak-S

S-box menyediakan nonlinieritas melalui substitusi tabel yang dirancang dengan kriteria ketat. Desain yang buruk membuka peluang serangan diferensial dan linear. Evaluasi melibatkan metrik seperti nonlinieritas, bias Walsh, dan uniformitas diferensial.

Standar AES mendokumentasikan konstruksi S-box yang kuat berbasis invers di medan $\mathbb{F}_{2^8}$ yang diikuti transformasi afine (*Advanced Encryption Standard (AES)* 2001).

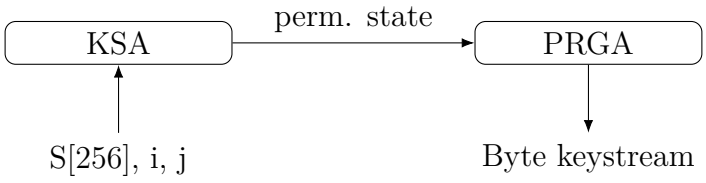
Bab 6

Review Beberapa Cipher Alir dan Cipher Blok

6.1 RC4

RC4 adalah cipher alir yang pernah digunakan luas, tetapi kini tidak direkomendasikan karena kelemahan bias awal dan serangan praktis. State internal berbasis permutasi byte menghasilkan keystream, namun inisialisasi yang tidak memadai membuka kebocoran statistik. Standar modern menyarankan migrasi ke alternatif seperti ChaCha20.

Analisis bias RC4 menunjukkan bahwa keystream tidak mendekati acak sempurna sehingga memungkinkan pemulihan kunci dalam skenario tertentu. Oleh karena itu, banyak standar telah mendeprikasikannya (Melnikov **and** Nir 2015; Mironov 2002).



Gambar 6.1: Skema tinggi RC4: KSA (Key Scheduling Algorithm) dan PRGA.

6.2 A5

Keluarga A5 digunakan pada GSM untuk enkripsi suara dan data, dengan variasi tingkat keamanan yang berbeda. Beberapa varian mengalami serangan yang memungkinkan pemulihan kunci dengan data yang relatif sedikit. Desain berbasis LFSR menyoroti pentingnya nonlinieritas yang kuat.

Kajian publik mengilustrasikan trade-off antara efisiensi dan keamanan pada sistem komunikasi bergerak.

6.3 DES

DES adalah cipher blok 56-bit yang bersejarah, kini dianggap tidak aman karena ruang kunci kecil dan serangan brute-force yang layak. Struktur Feistel dan S-box DES tetap berpengaruh dalam desain cipher modern. Penggantinya, 3DES, memperpanjang umur DES dengan komposisi berulang.

Standar saat ini merekomendasikan AES sebagai pengganti karena keamanan dan kinerja superior. Lihat *Advanced Encryption Standard (AES)* (2001) untuk detail.

6.4 Double DES dan Triple DES

Double DES tidak memberikan peningkatan keamanan yang signifikan karena serangan meet-in-the-middle, sedangkan Triple DES memperbaiki kelemahan ini dengan komposisi tiga kali. Namun, kinerja 3DES lebih rendah dibanding AES, dan parameter keamanan efektifnya kini dibatasi dalam standar.

Transisi ke AES direkomendasikan untuk aplikasi baru demi keamanan jangka panjang.

6.5 GOST

GOST 28147-89 adalah cipher blok yang digunakan di Rusia dengan struktur dan S-box yang berbeda dari DES. Evaluasi publik terbatas di masa awal, namun analisis selanjutnya mengungkap berbagai sifat desain yang menarik. Interoperabilitas global mendorong adopsi AES di banyak sistem.

Kajian perbandingan menekankan pentingnya proses standardisasi dan analisis terbuka.

6.6 RC5

RC5 adalah cipher blok yang dapat dikonfigurasi jumlah rondenya, ukuran blok, dan ukuran kunci. Desain sederhana berbasis rotasi, XOR, dan penjumlahan menawarkan kinerja baik. Keamanan bergantung pada parameter; ronde yang terlalu sedikit rentan terhadap kriptanalisis.

Pemilihan parameter konservatif diperlukan untuk mencapai keamanan yang memadai dalam praktik.

6.7 RC6

RC6 memperluas RC5 dengan operasi tambahan untuk meningkatkan keamanan dan kinerja pada platform tertentu. Meskipun kuat, pemilihan AES akhirnya jatuh pada Rijndael karena kombinasi keamanan, kinerja, dan kesederhanaan implementasi.

Studi kandidat AES memberikan pelajaran berharga tentang evaluasi terbuka.

6.8 Advanced Encryption Standard (AES)

AES adalah standar cipher blok modern yang menggantikan DES dan banyak diadopsi secara global. Desain berbasis SPN dengan S-box dari medan berhingga dan transformasi linier yang kuat memberikan keamanan dan kinerja tinggi. Implementasi perangkat keras dan lunak tersedia luas.

Standar *Advanced Encryption Standard (AES)* (2001) mendokumentasikan spesifikasi dan rasional desain AES yang telah diuji waktu.

Tabel 6.1: Ringkasan status keamanan beberapa cipher

Cipher	Status	Catatan
RC4	Dilarang	Bias keystream, didepresiasi (Melnikov **and** Nir 2015)
DES	Tidak aman	Kunci 56-bit, brute-force layak
3DES	Legacy	Batasan penggunaan (*Transitioning the Use of Cryptographic Algorithn*
AES	Direkomendasikan	Standar modern (*Advanced Encryption Standard (AES)* 2001)
A5/1	Rentan	Serangan real-time (Biryukov, Shamir **and** Wagner 2000)

6.9 Penerapan Kriptografi Simetri dalam Kehidupan Sehari-hari

Kriptografi simetri digunakan dalam berbagai aplikasi seperti perbankan elektronik, komunikasi seluler, dan penyimpanan data terenkripsi. Kinerja tinggi dan efisiensi menjadikannya pilihan untuk enkripsi bulk data. Keamanan nyata bergantung pada mode operasi, manajemen kunci, dan integrasi sistem.

Penerapan yang tepat memerlukan kepatuhan terhadap standar dan audit implementasi.

6.9.1 Transaksi dengan Mesin ATM

Transaksi ATM memanfaatkan enkripsi simetri untuk melindungi PIN dan data sensitif selama transmisi dan penyimpanan. Infrastruktur kunci dan modul keamanan perangkat keras (HSM) mengelola kunci secara aman. Prosedur rotasi dan injeksi kunci yang ketat penting untuk mencegah kompromi.

Standar industri menyediakan pedoman implementasi yang mengikat untuk lembaga keuangan.

6.9.2 Komunikasi dengan Telepon Seluler

Komunikasi seluler menggunakan cipher simetri untuk melindungi suara dan data melintasi jaringan radio yang tidak dipercaya. Protokol keamanan mengelola autentikasi, perjanjian kunci, dan enkripsi end-to-air. Evolusi generasi jaringan membawa perbaikan kriptografis dan mitigasi kelemahan historis.

Adopsi algoritma modern dan konfigurasi konservatif diperlukan untuk ketahanan jangka panjang.

6.9.3 Contoh Kode: ChaCha20-Poly1305 (AEAD)

Contoh berikut menggunakan pustaka Python `cryptography` untuk AEAD yang direkomendasikan di banyak protokol modern (Langley, Nir **and** Benjamin 2018).

Listing 6.1: ChaCha20-Poly1305 AEAD

```
from cryptography.hazmat.primitives.ciphers.aead import
   ChaCha20Poly1305
import os

key = ChaCha20Poly1305.generate_key()
chacha = ChaCha20Poly1305(key)
nonce = os.urandom(12)
aad = b"header"
pt = b"pesan rahasia"
ct = chacha.encrypt(nonce, pt, aad)
rt = chacha.decrypt(nonce, ct, aad)
assert rt == pt
```

Bab 7

Kriptografi Kunci Publik

7.1 Konsep Kriptografi Kunci Publik

Kriptografi kunci publik memperkenalkan pasangan kunci asimetris yang memungkinkan distribusi kunci yang skalabel dan layanan tanda tangan digital. Dengan memisahkan kunci untuk enkripsi dan dekripsi, skema ini mengatasi masalah distribusi kunci pada sistem simetri. Keamanan bergantung pada asumsi komputasi seperti faktorisasi atau logaritma diskrit.

Penggunaan kunci publik memfasilitasi protokol yang sebelumnya sulit dicapai, seperti otentikasi tanpa berbagi rahasia. Referensi terbuka menggambarkan landasan formal dan praktiknya (Diffie **and** Hellman 1976).

7.2 Sejarah Kriptografi Kunci Publik

Gagasan kunci publik dipopulerkan oleh Diffie dan Hellman pada 1976 dan segera diikuti oleh skema RSA pada 1978. Sejarah ini menunjukkan bagaimana kemajuan teoretis dengan cepat diadopsi menjadi standar dan aplikasi nyata. Evolusi selanjutnya memperkenalkan skema berbasis kurva eliptik untuk efisiensi lebih baik.

Sumber primer tersedia secara terbuka dan menjadi bacaan utama untuk memahami motivasi dan desain awal (Diffie **and** Hellman 1976; Rivest, Shamir **and** Adleman 1978).

7.3 Perbandingan Kriptografi Kunci Simetri dan Publik

Kunci simetri unggul dalam kinerja dan kesederhanaan, tetapi menantang pada distribusi kunci luas. Kunci publik mendukung skalabilitas dan layanan lanjutan, tetapi lebih mahal secara komputasi. Sistem nyata sering mengadopsi pendekatan hibrida untuk memanfaatkan kekuatan masing-masing.

Analisis biaya dan risiko harus mempertimbangkan kasus penggunaan spesifik serta infrastruktur yang tersedia.

7.4 Aplikasi Kriptografi Kunci Publik

Aplikasi mencakup enkripsi kunci publik, tanda tangan digital, dan pertukaran kunci aman untuk sesi simetri. Kerangka PKI memungkinkan verifikasi identitas melalui sertifikat yang ditandatangani otoritas tepercaya. Model kepercayaan harus jelas dan sesuai konteks organisasi.

Contoh implementasi nyata dapat ditemukan pada TLS dan sistem email aman.

7.5 Algoritma RSA

RSA bergantung pada kesulitan faktorisasi modulus komposit besar dan menggunakan perpangkatan modular untuk enkripsi/dekripsi. Pemilihan parameter yang tepat seperti ukuran kunci dan padding OAEP menentukan keamanan. Implementasi harus tahan kanal samping dan menghindari kerentanan seperti ketidakacakan pada pembangkitan kunci.

Standar dan rekomendasi terbuka memberikan pedoman konfigurasi yang aman untuk berbagai aplikasi.

7.6 Pertukaran Kunci Diffie-Hellman

Diffie–Hellman memungkinkan dua pihak menyepakati kunci rahasia melalui saluran publik berdasarkan kesulitan logaritma diskrit. Versi eliptiknya (ECDH) menawarkan efisiensi lebih baik dengan keamanan sebanding. Pemilihan grup yang aman sangat penting untuk mencegah serangan terkait struktur.

Panduan standar merekomendasikan parameter terverifikasi dan praktik implementasi yang ketat.

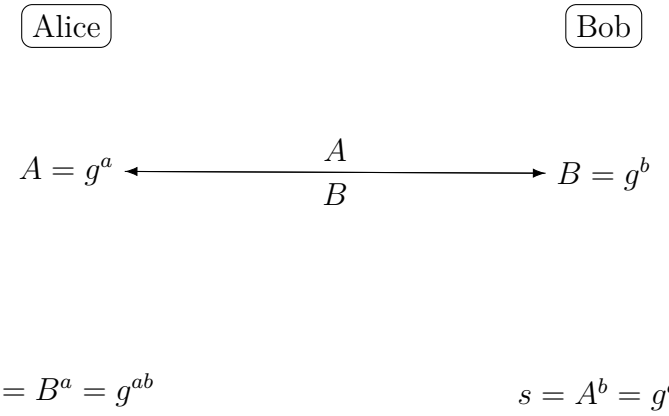
7.6.1 Contoh Kode: ECDH X25519

Berikut contoh pertukaran kunci menggunakan X25519 sesuai Langley, Hamburg **and** Turner (2016); gunakan kanal autentikasi untuk mencegah MITM.

Listing 7.1: ECDH X25519 di Python

```
from cryptography.hazmat.primitives.asymmetric import x25519

alice_private = x25519.X25519PrivateKey.generate()
```



Gambar 7.1: Skema tinggi Diffie–Hellman klasik. Untuk ECDH, ganti operasi eksponensial dengan perkalian titik pada kurva.

```
bob_private = x25519.X25519PrivateKey.generate()

alice_public = alice_private.public_key()
bob_public = bob_private.public_key()

alice_shared = alice_private.exchange(bob_public)
bob_shared = bob_private.exchange(alice_public)
assert alice_shared == bob_shared
```

Parameter kurva dan praktik implementasi direkomendasikan pada Langley, Hamburg **and** Turner (2016) **and** *Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography* (2018).

7.7 Algoritma ElGamal

ElGamal adalah skema enkripsi kunci publik berbasis DLP yang menyediakan kerangka probabilistik untuk keamanan IND-CPA. Variasi dan optimalisasi meningkatkan efisiensi, namun penanganan pesan dan padding harus benar. Analisis keamanan menekankan pentingnya randomness yang kuat.

Skema ini juga mendasari konstruksi tanda tangan digital tertentu.

7.8 Algoritma Knapsack

Skema knapsack awalnya menarik karena efisiensi, namun banyak varian awal telah dipatahkan melalui reduksi ke struktur yang dapat dipecahkan. Pengalaman ini menekankan pentingnya analisis publik dan kehati-hatian terhadap asumsi yang tidak mapan. Penelitian berlanjut pada skema berbasis masalah kombinatorial lainnya.

Pembelajaran dari knapsack memengaruhi kehati-hatian dalam mengevaluasi skema baru.

7.9 Algoritma untuk Perpangkatan Modulo

Perpangkatan modular efisien menggunakan teknik seperti square-and-multiply dan Montgomery reduction. Pilihan representasi dan mitigasi kanal samping sangat penting dalam implementasi praktis. Perbedaan antara enkripsi dan penandatanganan memengaruhi parameter dan jalur eksekusi.

Optimasi perangkat keras dan lunak mendukung performa pada kunci besar tanpa mengorbankan keamanan.

7.10 Pembangkitan Bilangan Prima

Pembangkitan prima aman memerlukan uji primalitas probabilistik dan sumber entropi kuat. Struktur prima harus menghindari pola yang melemahkan keamanan. Proses harus menghasilkan bukti atau indikator kuat terhadap keacakan yang memadai.

Panduan praktik yang baik tersedia luas dalam literatur terbuka.

7.11 Kode Program Algoritma RSA

Contoh kode sering disertakan dalam literatur untuk ilustrasi, tetapi implementasi produksi harus menggunakan pustaka tepercaya yang diaudit. Kesalahan kecil seperti padding salah atau pengelolaan kunci yang lemah dapat berakibat fatal. Evaluasi keamanan harus memasukkan pengujian dan audit menyeluruh.

Standarisasi antarmuka dan penggunaan API yang benar membantu mengurangi risiko implementasi.

Bab 8

Pembangkit Bilangan Acak

8.1 Linear Congruential Generator (LCG)

LCG menghasilkan deret semu-acak menggunakan relasi linear sederhana modulo bilangan bulat, menawarkan efisiensi tinggi namun kualitas acak yang terbatas. Korelasi dan periode yang relatif pendek membuatnya tidak cocok untuk kriptografi. Analisis spektral menunjukkan struktur yang dapat dieksploitasi.

Untuk aplikasi kriptografi, LCG tidak direkomendasikan dan hanya layak untuk simulasi non-keamanan. Literatur terbuka menyoroti kelemahan fundamentalnya.

8.2 Pembangkit Bilangan Acak yang aman untuk kriptografi

Pembangkit bilangan acak aman (CSPRNG) dirancang untuk menghasilkan output yang tidak dapat diprediksi bahkan dengan akses ke sebagian keadaan internal. Konstruksi modern menggabungkan entropi dari berbagai sumber dan menggunakan fungsi derivasi yang tahan serangan. Properti keamanan formal mencakup resistance terhadap state compromise dan backtracking.

Standar dan pedoman implementasi tersedia dari NIST dan organisasi lain untuk memastikan kualitas dan keamanan keluaran.

Tabel 8.1: Ringkasan TRNG, PRNG, dan CSPRNG

Jenis	Sumber	Catatan
TRNG	Fenomena fisik	Butuh pengkondisi, uji kesehatan (*Recommendation for t*
PRNG	Deterministik	Tidak cocok untuk keamanan
CSPRNG	Deterministik + seed rahasia	Sesuai SP 800-90A/B/C (*Recommendation for Random N*

Gambar 8.1: Arsitektur tinggi: sumber entropi, kondisioner, dan DRBG (*Recommendation for Random Number Generation Using Deterministic Random Bit Generators* 2015; *Recommendation for the Entropy Sources Used for Random Bit Generation* 2018; *Recommendation for Random Bit Generator (RBG) Constructions* 2012).

8.3 Blum Blum Shub (BBS)

BBS adalah CSPRNG berbasis masalah faktorisasi yang menawarkan jaminan teoretis kuat namun relatif lambat. Mekanismenya bergantung pada operasi kuadrat modulo produk dua bilangan prima besar. Keamanannya terkait erat dengan kesulitan mengambil akar kuadrat modulo komposit.

BBS lebih cocok sebagai contoh teoretis daripada generator praktik, tetapi tetap penting dalam pendidikan kriptografi.

8.4 CSPRNG Berbasis Algoritma RSA

Generator berbasis RSA memanfaatkan sifat perpangkatan modular untuk menghasilkan bit yang sulit diprediksi. Efisiensi lebih rendah dibanding konstruksi berbasis hash atau blok cipher, tetapi menyediakan jaminan berbasis asumsi faktorisasi. Desain harus mempertimbangkan mitigasi kanal samping.

Penggunaan praktis sering terbatas oleh kebutuhan performa tinggi pada aplikasi modern.

8.5 CSPRNG Berbasis Chaos

Pendekatan berbasis chaos mencoba memanfaatkan dinamika nonlinier untuk menghasilkan deret semu-acak. Namun, banyak proposal kekurangan analisis keamanan formal dan menunjukkan kelemahan ketika dimodelkan secara digital. Penggunaan harus sangat hati-hati dan mengikuti evaluasi komunitas yang ketat.

Konsensus saat ini mendukung penggunaan konstruksi yang telah distandardisasi dan diuji luas.

8.5.1 Contoh Kode: Penggunaan secrets di Python

Untuk kebutuhan kriptografi umum, gunakan modul `secrets` yang memanfaatkan CSPRNG sistem (Eastlake, Schiller **and** Crocker 2005).

Listing 8.1: Contoh penggunaan secrets

```
import secrets

key = secrets.token_bytes(32)
nonce = secrets.token_bytes(12)
rand_int = secrets.randbelow(1 << 128)
print(len(key), len(nonce), rand_int.bit_length())
```

Bab 9

Fungsi Hash Satu Arah

9.1 Fungsi Hash Satu Arah

Fungsi hash satu arah memetakan masukan arbitrer ke keluaran berukuran tetap dengan sifat tahan-preimage, tahan-second-preimage, dan tahan-tabrakan. Sifat-sifat ini memungkinkan aplikasi luas pada verifikasi integritas, komitmen, dan konstruksi autentikasi. Desain modern menghindari kelemahan yang ditemukan pada algoritma lama seperti MD5.

Analisis formal memodelkan keamanan terhadap penyerang yang mampu melakukan query adaptif; standar terbuka menyediakan spesifikasi dan panduan penggunaan yang aman.

Tabel 9.1: Sifat keamanan fungsi hash

Sifat	Deskripsi singkat
Preimage-resistant	Sulit menemukan x dari h(x)
Second-preimage	Sulit menemukan x' s.t. h(x')=h(x)
Collision-resistant	Sulit menemukan pasangan (x,x') dengan h(x)=h(x')

9.2 Algoritma MD5

MD5 adalah fungsi hash yang kini tidak aman karena serangan tabrakan praktis yang terpublikasi. Penggunaan dalam konteks keamanan tidak direkomendasikan dan harus diganti dengan keluarga SHA-2 atau SHA-3. Studi kasus tabrakan mendemonstrasikan risiko integritas.

Meskipun historis penting, MD5 kini terutama menjadi contoh pedagogis mengenai desain yang rentan dan deprecasi algoritma.

9.3 Secure Hash Algorithm (SHA)

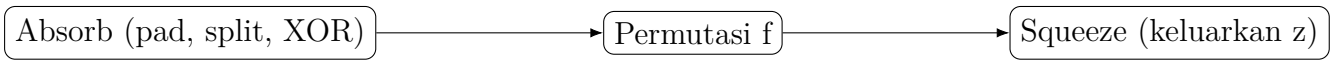
Keluarga SHA-2 (SHA-256/384/512) memberikan keamanan yang kuat dengan performa baik, sementara SHA-1 didepresiasi karena tabrakan praktis. Spesifikasi terbuka memudahkan interoperabilitas dan implementasi yang konsisten. Penggunaan SHA-2 direkomendasikan luas.

Penerapan harus memperhatikan mode penggunaan, padding, dan potensi kanal samping untuk menjaga keamanan.

9.4 SHA-3 (Keccak)

SHA-3 (Keccak) adalah standar hash berbasis sponge yang menawarkan desain berbeda dari SHA-2 dan ketahanan terhadap serangan yang diketahui. Struktur sponge memberikan fleksibilitas untuk hash dan XOF, serta konstruksi terkait. Analisis keamanan mendukung adopsi luas pada aplikasi baru.

Spesifikasi tersedia secara terbuka dan menyediakan parameterisasi untuk berbagai kebutuhan keamanan (*SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions* 2015).



Gambar 9.1: Skema tinggi sponge (Keccak) (*SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions* 2015; Bertoni **andothers** 2013).

9.5 Message Authentication Code (MAC)

MAC menyediakan autentikasi dan integritas pesan berbasis kunci rahasia. Konstruksi populer meliputi HMAC yang dibangun di atas fungsi hash dan CMAC di atas cipher blok. Keamanan menuntut pengelolaan kunci yang tepat dan pemilihan parameter konservatif.

Kesesuaian MAC dengan protokol harus mempertimbangkan kebutuhan anti-replay dan penandatanganan entitas.

9.6 Algoritma MAC

HMAC menawarkan keamanan yang kuat dengan asumsi hash dasar aman dan banyak distandardisasi; CMAC memberikan alternatif berbasis AES dengan kualitas serupa. Pemilihan algoritma sering ditentukan oleh ketersediaan akselerasi perangkat keras atau pustaka yang telah diaudit. Implementasi harus menghindari kebocoran waktu pada perbandingan tag.

Dokumentasi standar menyediakan panduan interoperabilitas dan parameter yang disarankan.

9.6.1 Contoh Kode: SHA-256 dan HMAC

Berikut contoh hash dan HMAC di Python.

Listing 9.1: SHA-256 dan HMAC

```
import hashlib, hmac

d = hashlib.sha256(b"pesan").hexdigest()
tag = hmac.new(b"kunci", b"pesan", hashlib.sha256).hexdigest()
print(d, tag)
```

Lihat Krawczyk, Bellare **and** Canetti (1997) untuk HMAC dan Krawczyk **and** Eronen (2010) untuk HKDF; BLAKE2 terdokumentasi pada Aumasson **andothers** (2015).

Bab 10

Tanda Tangan Digital

10.1 Review Layanan Kriptografi

Tanda tangan digital mengimplementasikan layanan otentikasi, integritas, dan nir-sangkal dalam kerangka kunci publik. Dengan mengikat identitas pada pesan melalui kunci privat, pihak ketiga dapat memverifikasi keaslian menggunakan kunci publik. Keamanan bergantung pada asumsi komputasi dan fungsi hash yang kuat.

Definisi formal keamanan seperti unforgeability under chosen-message attack (EUF-CMA) menjadi standar penilaian skema tanda tangan. Implementasi harus memastikan sumber randomness yang kuat untuk menghindari kebocoran kunci.

10.2 Penandatanganan dengan Cara Mengenkripsi Pesan

Model naif yang menyamakan tanda tangan dengan enkripsi menggunakan kunci privat tidak mencukupi dan rentan terhadap manipulasi struktur. Skema tanda tangan modern menggunakan padding dan hash untuk mengikat pesan dengan aman. Rancangan harus mencegah serangan substitusi dan replay.

Standar seperti PKCS#1 mendefinisikan skema yang benar untuk RSA signatures yang menghindari kelemahan konstruksi sederhana (Schmidt **andothers** 2016).

10.3 Tanda-tangan Digital dengan Kombinasi Fungsi Hash dan Kriptografi Kunci Publik

Skema tanda tangan modern memanfaatkan fungsi hash untuk mengompresi pesan sebelum operasi kunci publik, meningkatkan efisiensi dan keamanan. Kombinasi ini mengurangi

kerentanan terhadap serangan struktural dan memastikan kompatibilitas dengan pesan berukuran arbitrer. Penggunaan hash yang aman adalah kunci.

Pengikatan domain dan pengkodean yang benar diperlukan untuk mencegah ambiguitas dan serangan framing.

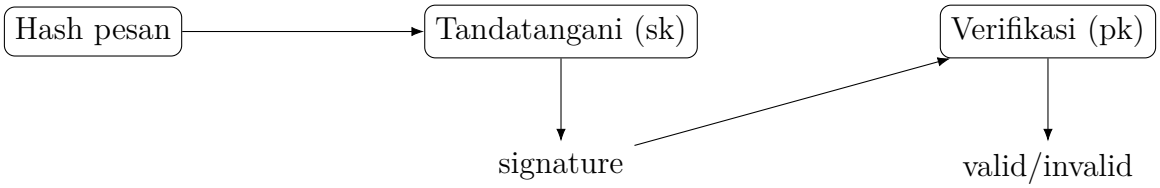
10.4 Tanda-tangan Digital dengan Fungsi Hash dan Kriptografi Kunci Publik

Pendekatan praktis seperti RSA-PSS dan ECDSA menggunakan hash sebagai inti, dengan padding dan pemetaan yang dirancang untuk keamanan formal. Parameter yang salah atau randomness lemah dapat merusak keamanan seluruh sistem. Oleh itu, pedoman standar harus diikuti secara ketat.

Analisis formal dan pengalaman industri mendukung penggunaan skema ini dalam aplikasi luas.

Tabel 10.1: Ringkasan skema tanda tangan

Skema	Basis	Catatan
RSA-PSS	Faktorisasi	Padding probabilistik, standar (Schmidt **andothers** 2016)
ECDSA	DLP kurva eliptik	Nonce deterministik (Pornin 2013)
Ed25519	EdDSA (Curve25519)	Deterministik, performa baik (Josefsson **and** Liusvaara 2017)



Gambar 10.1: Alur umum tanda tangan digital: hash, tanda tangan, verifikasi.

10.5 Digital Standard Algorithm (DSA)

DSA adalah skema tanda tangan berbasis DLP yang mengandalkan randomness unik per tanda tangan. Kegagalan menjaga nonce unik dapat mengungkap kunci privat, seperti ditunjukkan dalam berbagai insiden nyata. Versi eliptiknya, ECDSA, menawarkan efisiensi lebih baik.

Standar dan praktik terbaik menekankan pentingnya generator randomness yang kuat dan audit implementasi.

10.5.1 Contoh Kode: Ed25519

Contoh penandatanganan dan verifikasi menggunakan Ed25519 di Python (The cryptography developers 2024b).

Listing 10.1: Ed25519 sign/verify

```
from cryptography.hazmat.primitives.asymmetric import ed25519

sk = ed25519.Ed25519PrivateKey.generate()
pk = sk.public_key()

msg = b"pesan"
sig = sk.sign(msg)
pk.verify(sig, msg)
```

Bab 11

Sertifikat Digital dan Infrastruktur Kunci Publik

11.1 Sertifikat Digital

Sertifikat digital mengikat kunci publik dengan identitas melalui tanda tangan otoritas sertifikat (CA). Struktur sertifikat menyertakan informasi identitas, kunci publik, periode validitas, dan ekstensi kebijakan. Verifikasi melibatkan pengecekan tanda tangan, validitas waktu, dan status pencabutan.

Sertifikat membentuk dasar kepercayaan pada banyak protokol jaringan dan aplikasi perangkat lunak.

Tabel 11.1: Bidang utama dalam sertifikat X.509

Bidang	Deskripsi
Subject	Identitas pemegang sertifikat
Issuer	CA penerbit
Validity	NotBefore/NotAfter
SubjectPublicKeyInfo	Algoritma dan kunci publik
Extensions	KeyUsage, EKU, SAN, dll.

11.2 X.509

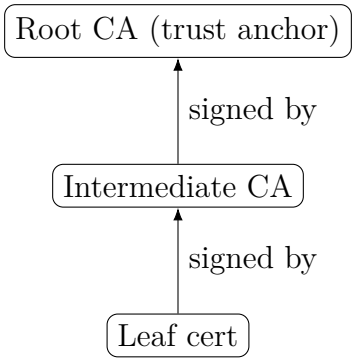
Standar X.509 mendefinisikan format sertifikat dan jalur validasi yang digunakan dalam infrastruktur kunci publik modern. Ekstensi seperti KeyUsage dan ExtendedKeyUsage mengontrol tujuan penggunaan sertifikat. Kepatuhan terhadap profil X.509 memastikan interoperabilitas antar implementasi.

Dokumen profil seperti Cooper **andothers** (2008) menyediakan pedoman rinci untuk penerapan.

11.3 Verifikasi Sertifikat Digital

Verifikasi mencakup pengecekan rantai ke CA tepercaya, validitas waktu, kebijakan, dan status pencabutan melalui CRL atau OCSP. Kesalahan konfigurasi atau validasi yang lemah membuka peluang serangan man-in-the-middle. Implementasi harus tegas terhadap kesalahan untuk menjaga keamanan.

Pedoman industri menekankan praktik validasi yang kuat dan manajemen trust store yang aman.



Gambar 11.1: Rantai kepercayaan: leaf → intermediate → root.

11.4 Menggunakan Sertifikat Digital untuk Enkripsi Pesan

Sertifikat memungkinkan distribusi kunci publik yang autentik sehingga pihak lain dapat mengenkripsi pesan secara aman. Protokol harus memastikan bahwa sertifikat yang digunakan sesuai tujuan dan belum dicabut. Integrasi dengan sistem email atau komunikasi lainnya mengikuti standar yang ditetapkan.

Kebijakan organisasi menentukan siapa yang dapat mengeluarkan dan menggunakan sertifikat untuk berbagai layanan.

11.5 Public Key Infrastructure (PKI)

PKI adalah ekosistem kebijakan, perangkat lunak, dan entitas yang mengelola siklus hidup kunci publik dan sertifikat. Komponen mencakup CA, Registration Authority, repositori, dan mekanisme pencabutan. Keamanan PKI tidak hanya teknis tetapi juga bergantung pada tata kelola dan proses operasional.

Standar seperti Cooper **andothers** (2008) mendasarkan interoperabilitas dan kepercayaan sistem yang luas.

11.5.1 OCSP, CT, dan ACME

OCSP menyediakan status pencabutan real-time (Chokhani **andothers** 2013); Certificate Transparency menambah auditabilitas penerbitan (Laurie, Langley **and** Kasper 2013); dan ACME mengotomatiskan penerbitan sertifikat (Barnes **andothers** 2019).

11.5.2 Contoh Kode: Membaca X.509 di Python

Berikut contoh mem-parsing sertifikat X.509 menggunakan pustaka `cryptography`.

Listing 11.1: Parse X.509

```
from cryptography import x509
from cryptography.hazmat.backends import default_backend

with open("cert.pem", "rb") as f:
    data = f.read()
cert = x509.load_pem_x509_certificate(data, default_backend())
print(cert.subject, cert.issuer, cert.not_valid_after)
```

Bab 12

Protokol Kriptografi

12.1 Protokol Komunikasi dengan Sistem Kriptografi Simetri

Protokol berbasis kunci simetri membutuhkan mekanisme distribusi kunci yang aman, seringkali melalui kanal sebelumnya atau infrastruktur terpusat. Desain protokol harus menangani sinkronisasi, manajemen nonce, dan integritas pesan. Kesalahan desain seperti reuse nonce dapat menghancurkan keamanan.

Model ancaman harus mempertimbangkan penyerang aktif dan pasif untuk menetapkan mekanisme perlindungan yang sesuai.

12.2 Protokol Komunikasi dengan Sistem Kriptografi Kunci Publik

Kunci publik memungkinkan negosiasi kunci sesi melalui saluran publik dan otentikasi identitas melalui sertifikat. Desain protokol harus mencakup validasi sertifikat, pengecekan status pencabutan, dan perlindungan terhadap downgrade. Integrasi dengan PKI menambah kompleksitas tetapi meningkatkan skalabilitas.

Contoh konkret dapat ditemukan pada protokol TLS modern (Rescorla 2018).

Tabel 12.1: Ringkasan beberapa protokol

Protokol	Tujuan	Catatan
TLS 1.3	Saluran aman	Forward secrecy, AEAD (Rescorla 2018)
Kerberos	Otentikasi jaringan	Tiket berbasis kunci simetri (Kohl **and** Neuman 200
Signal (X3DH+DR)	Pesan aman end-to-end	Forward secrecy, post-compromise security (Perrin **a**

12.3 Protokol untuk Tanda-tangan Digital

Protokol tanda tangan menyertakan mekanisme pengikatan konteks, penandatanganan data, dan verifikasi oleh pihak penerima. Desain harus mencegah pemutaran ulang dan ambiguitas data yang ditandatangani. Pengelolaan kunci yang baik dan audit menjadi elemen penting.

Standar format tanda tangan dan pengodean data membantu interoperabilitas antar sistem.

12.4 Protokol untuk Tanda-tangan Digital Plus Enkripsi

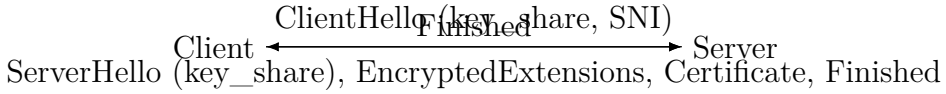
Kombinasi tanda tangan dan enkripsi harus memperhatikan urutan operasi untuk mencapai sifat keamanan yang diinginkan. Tanda tangan-lalu-enkripsi dan enkripsi-lalu-tanda tangan memiliki implikasi keamanan dan privasi yang berbeda. Pilihan harus didasarkan pada model ancaman dan kebutuhan aplikasi.

Literatur terbuka membahas skema yang mengintegrasikan keduanya dengan keamanan terbukti.

12.5 Protokol Pertukaran Kunci Diffie-Hellman

Protokol Diffie–Hellman memerlukan validasi parameter dan proteksi terhadap serangan man-in-the-middle melalui autentikasi. Versi modern menggabungkan ephemeral keys untuk mencapai forward secrecy. Detail implementasi memengaruhi keamanan secara signifikan.

TLS 1.3 mengilustrasikan praktik terbaik dalam perancangan protokol pertukaran kunci (Rescorla 2018).



Gambar 12.1: Skema ringkas handshake TLS 1.3.

12.6 Otentikasi

Otentikasi memastikan identitas entitas sebelum melanjutkan pertukaran data sensitif. Skema dapat berbasis kata sandi, token, atau kunci publik, masing-masing dengan

tantangan keamanan yang khas. Mekanisme harus tahan terhadap phishing, replay, dan brute-force.

Desain sistem sering memanfaatkan multi-faktor untuk meningkatkan jaminan identitas.

12.7 Secure Socket Layer (SSL)

SSL/TLS menyediakan saluran aman di atas TCP dengan negosiasi kriptografi, autentikasi, dan integritas. Evolusi ke TLS 1.3 menyederhanakan suite kriptografi dan menguatkan keamanan. Implementasi harus selalu mengikuti praktik terbaru dan menonaktifkan versi lama yang rentan.

Spesifikasi Rescorla (2018) memberikan rujukan utama untuk penerapan modern.

12.7.1 Contoh Kode: TLS Client Sederhana

Contoh berikut membuka koneksi TLS menggunakan modul standar Python.

Listing 12.1: TLS client

```
import socket, ssl

context = ssl.create_default_context()
with socket.create_connection(("example.com", 443)) as sock:
    with context.wrap_socket(sock, server_hostname="example.com")
        as ssock:
        ssock.sendall(b"GET / HTTP/1.1\r\nHost: example.com\r\n\r
            \n")
        print(ssock.recv(1024))
```

Bab 13

Manajemen Kunci

13.1 Pembangkitan Kunci

Pembangkitan kunci harus menggunakan CSPRNG yang kuat dan parameter yang sesuai dengan target keamanan. Proses harus menghindari struktur yang memperlemah, misalnya kunci dengan entropi rendah atau pola yang dapat ditebak. Audit berkala dan bukti kualitas randomness dianjurkan.

Standar dan pedoman terbuka menyediakan rekomendasi ukuran kunci dan prosedur pembangkitan yang aman.

Tabel 13.1: Tahap siklus hidup kunci dan ringkasannya

Tahap	Ringkasan
Generate	CSPRNG, ukuran kunci sesuai (*Recommendation for Cryptographic Key Generation* 201
Distribute	Kanal aman/PKI (Cooper **andothers** 2008)
Store	Proteksi HSM/enclave
Use	Parameter benar (nonce/IV), audit
Rotate	Sesuai kebijakan risiko (*Recommendation for Key Management, Part 1: General* 2020)
Destroy	Penghapusan terverifikasi

13.2 Penyebaran Kunci

Penyebaran kunci memerlukan kanal aman atau mekanisme kunci publik untuk menghindari penyadapan dan substitusi. Prosedur harus memverifikasi integritas dan autentikasi sumber. Kesalahan pada tahap ini sering kali berakibat kompromi sistem secara keseluruhan.

Penggunaan perangkat keras tepercaya dan protokol yang telah distandardisasi membantu mengurangi risiko.

13.3 Penyimpanan Kunci

Kunci harus disimpan dalam lingkungan yang dilindungi dari akses tidak sah dan kebocoran, termasuk perlindungan memori dan perangkat keras. Mekanisme seperti HSM atau enclave menyediakan proteksi tambahan. Kebijakan rotasi dan pencatatan akses meningkatkan keamanan operasional.

Prinsip paling sedikit hak akses dan pemisahan tugas harus diterapkan untuk mencegah penyalahgunaan.

13.4 Penggunaan Kunci

Penggunaan kunci harus mematuhi tujuan yang ditentukan dan kebijakan organisasi. Parameter seperti nonce dan IV harus dikelola dengan benar untuk mencegah reuse. Pustaka kriptografi yang tepercaya dan antarmuka yang aman mengurangi kemungkinan kesalahan pemrograman.

Pengujian dan monitoring membantu mendeteksi anomali yang dapat mengindikasikan penyalahgunaan kunci.

13.5 Perubahan Kunci

Rotasi kunci berkala mengurangi dampak kompromi dan menyesuaikan dengan perubahan tingkat ancaman. Proses harus memastikan transisi yang mulus tanpa kehilangan data atau interoperabilitas. Dokumentasi dan validasi penting untuk menjaga jejak audit.

Kebijakan rotasi didasarkan pada nilai risiko aset dan eksposur operasional.

13.6 Penghancuran Kunci

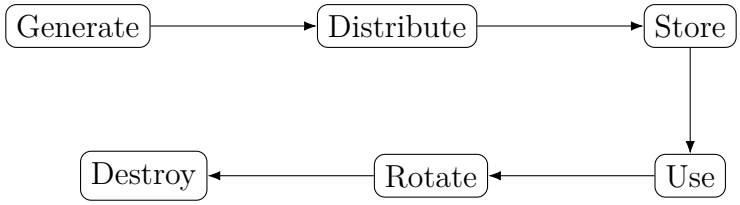
Penghancuran kunci yang aman memastikan bahwa materi kunci tidak dapat dipulihkan setelah masa pakai berakhir. Teknik termasuk overwriting, degaussing, atau penghancuran fisik sesuai konteks media. Proses harus terdokumentasi dan dapat diaudit.

Kepatuhan terhadap standar keamanan informasi memastikan penghentian yang terverifikasi.

13.6.1 Contoh Kode: HKDF

Contoh derivasi kunci dengan HKDF sesuai NIST dan dokumentasi pustaka Python (Krawczyk **and** Eronen 2010; The cryptography developers 2024c).

Listing 13.1: HKDF



Gambar 13.1: Siklus hidup kunci menurut pedoman NIST (*Recommendation for Key Management, Part 1: General* 2020).

```
from cryptography.hazmat.primitives.kdf.hkdf import HKDF
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.backends import default_backend

ikm = b"input keying material"
salt = b"salt"
info = b"context"
hkdf = HKDF(algorithm=hashes.SHA256(), length=32, salt=salt, info
   =info, backend=default_backend())
okm = hkdf.derive(ikm)
```

Bab 14

Steganografi

14.1 Sejarah Steganografi

Steganografi memfokuskan pada penyembunyian keberadaan pesan dengan menanam informasi ke dalam media pembawa. Sejarahnya panjang, mencakup teknik fisik tradisional hingga metode digital pada citra, audio, dan video. Perkembangan kriptografi dan komputasi memperluas baik teknik maupun deteksi steganalisis.

Kajian sejarah menggarisbawahi tujuan yang berbeda dari kriptografi: bukan menyamarkan isi, melainkan menyembunyikan keberadaan komunikasi.

14.2 Konsep dan Terminologi Steganografi

Terminologi utama meliputi cover object, stego object, embedding rate, dan payload. Tujuan desain adalah meminimalkan distorsi yang terdeteksi sambil mempertahankan kapasitas yang memadai. Model ancaman mempertimbangkan adversary yang melakukan uji statistik untuk mendeteksi keberadaan pesan.

Kualitas skema dinilai melalui ketaksempurnaan statistik dan ketahanan terhadap deteksi yang canggih.

14.3 Kriteria Steganografi yang baik

Kriteria meliputi imperceptibility, kapasitas yang memadai, dan robustness terhadap transformasi yang wajar. Trade-off antara kriteria ini harus dikelola sesuai aplikasi. Evaluasi empiris menggunakan dataset standar dan metrik yang disepakati.

Desain yang baik mempertimbangkan model serangan dan kemampuan deteksi modern.

14.4 Ranah Penyembunyian Data

Ranah umum mencakup domain ruang (spatial) untuk citra, domain frekuensi seperti DCT pada JPEG, dan domain transformasi lain untuk audio/video. Pemilihan ranah memengaruhi kapasitas dan visibilitas distorsi. Teknik embedding disesuaikan dengan karakteristik media.

Pemahaman format file dan proses kompresi penting untuk menjaga imperceptibility setelah pengolahan.

Tabel 14.1: Perbandingan ranah penyisipan

Ranah	Kelebihan	Kekurangan
Spatial (LSB)	Implementasi sederhana	Rentan deteksi/statistik (Petitcolas 2000)
DCT (JPEG)	Lebih tahan kompresi	Implementasi kompleks (Westfeld 2001)
Audio/Video	Kapasitas lebih besar	Sinkronisasi sulit

14.5 Metode LSB

Least Significant Bit (LSB) mengganti bit paling tidak signifikan pada piksel atau sampel, memberikan kapasitas tinggi namun rentan terhadap deteksi statistik. Variasi adaptif mencoba mengurangi jejak statistik dengan memilih lokasi embedding berdasarkan konten. Kekokohan terhadap kompresi lossy terbatas.

Metode LSB menyoroti tantangan menjaga keseimbangan antara kapasitas dan ketak-terdeteksian.

Gambar 14.1: Skema sederhana embed/extract LSB.

14.6 Kombinasi Steganografi dan Kriptografi

Menggabungkan steganografi dengan kriptografi meningkatkan keamanan dengan melindungi isi sekaligus menyamarkan keberadaannya. Enkripsi dilakukan sebelum embedding untuk mencegah kebocoran informasi jika stego terdeteksi. Kunci embedding terpisah dapat mengatur lokasi penyisipan.

Praktik ini menambah lapisan keamanan namun juga kompleksitas dalam manajemen kunci dan proses.

14.7 Steganalisis

Steganalisis mengevaluasi media untuk mendeteksi jejak embedding menggunakan uji statistik dan pembelajaran mesin. Fitur tingkat tinggi dan model modern dapat mencapai deteksi yang akurat terhadap banyak teknik klasik. Skema yang aman harus diuji terhadap detektor canggih.

Evaluasi berkelanjutan diperlukan karena teknik deteksi terus berkembang.

14.8 Watermarking

Watermarking menanamkan informasi kepemilikan atau integritas ke dalam media dengan tujuan bertahan terhadap transformasi dan serangan. Berbeda dengan steganografi, watermarking tidak selalu bertujuan tidak terdeteksi total, tetapi lebih pada robustnes. Aplikasi mencakup hak cipta digital dan pelacakan distribusi.

Desain watermarking memerlukan kompromi antara ketahanan, kualitas media, dan kapasitas payload.

14.9 Noiseless Steganography

Noiseless steganography berupaya menyisipkan informasi tanpa menambah noise yang terdeteksi secara statistik. Teknik ini sering memanfaatkan struktur redundansi data dan model generatif. Tantangannya adalah menjaga konsistensi statistik sambil mempertahankan kapasitas.

Perkembangan terkini mengeksplorasi pendekatan berbasis pembelajaran mendalam untuk meningkatkan ketakterdeteksian.

14.9.1 Contoh Kode: LSB pada citra

Contoh sederhana menyisipkan bit ke LSB kanal biru menggunakan `Pillow`.

Listing 14.1: LSB sederhana

```
from PIL import Image

def embed_lsb(img_path: str, bits: bytes, out_path: str):
    img = Image.open(img_path).convert('RGB')
    pixels = img.load()
    w, h = img.size
    bit_iter = iter(bits)
    for y in range(h):
        for x in range(w):
```

```
            try:
                b = next(bit_iter)
            except StopIteration:
                img.save(out_path)
                return
            r, g, bl = pixels[x, y]
            bl = (bl & 0xFE) | (b & 1)
            pixels[x, y] = (r, g, bl)
    img.save(out_path)

def extract_lsb(img_path: str, nbits: int) -> bytes:
    img = Image.open(img_path).convert('RGB')
    pixels = img.load()
    w, h = img.size
    out = bytearray()
    count = 0
    for y in range(h):
        for x in range(w):
            if count == nbits:
                return bytes(out)
            bl = pixels[x, y][2]
            out.append(bl & 1)
            count += 1
    return bytes(out)
```

Bibliografi

Advanced Encryption Standard (AES) (2001). techreport FIPS PUB 197. NIST. URL: https://csrc.nist.gov/publications/detail/fips/197/final.

AlFardan, Nadhem J. **and** Kenneth G. Paterson (2013). **?**Lucky Thirteen: Breaking the TLS and DTLS Record Protocols**? in***IEEE Symposium on Security and Privacy*: URL: https://www.isg.rhul.ac.uk/tls/Lucky13.html.

Aumasson, Jean-Philippe **andothers** (2015). *The BLAKE2 Cryptographic Hash and Message Authentication Code (MAC)*. RFC 7693. URL: https://www.rfc-editor.org/rfc/rfc7693.

Barnes, Richard **andothers** (2019). *Automatic Certificate Management Environment (ACME)*. RFC 8555. URL: https://www.rfc-editor.org/rfc/rfc8555.

Bernstein, Daniel J. (2005). *Cache-timing attacks on AES*. techreport. URL: https://cr.yp.to/antiforgery/cachetiming-20050414.pdf.

Bertoni, Guido **andothers** (2013). **?**Keccak**? in***Advances in Cryptology—EUROCRYPT 2013 Tutorials*: **pages** 313–314. URL: https://keccak.team/keccak_tutorial.pdf.

Biham, Eli **and** Adi Shamir (1991). **?**Differential Cryptanalysis of DES-like Cryptosystems**? in***Advances in Cryptology—CRYPTO'90*: **pages** 2–21. URL: https://www.cs.technion.ac.il/~biham/Reports/differential-cryptanalysis-of-des-like.pdf.

Biryukov, Alex, Adi Shamir **and** David Wagner (2000). **?**Real Time Cryptanalysis of A5/1 on a PC**? in***Fast Software Encryption*: URL: https://link.springer.com/chapter/10.1007/3-540-44706-7_20.

Boneh, Dan **and** Victor Shoup (2020). *A Graduate Course in Applied Cryptography*. URL: https://toc.cryptobook.us/.

Chokhani, Santosh **andothers** (2013). *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP*. RFC 6960. URL: https://www.rfc-editor.org/rfc/rfc6960.

Cooper, David **andothers** (2008). *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. RFC 5280. URL: https://www.rfc-editor.org/rfc/rfc5280.

Diffie, Whitfield **and** Martin E Hellman (1976). **?**New Directions in Cryptography**?** **in***IEEE Transactions on Information Theory*: 22.6, **pages** 644–654. URL: https://ee.stanford.edu/~hellman/publications/24.pdf.

Dworkin, Morris (2007). *Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*. techreport SP 800-38D. NIST. URL: https://csrc.nist.gov/publications/detail/sp/800-38d/final.

— (2008). *An Interface and Algorithms for Authenticated Encryption*. RFC 5116. URL: https://www.rfc-editor.org/rfc/rfc5116.

Eastlake, Donald, Steve Schiller **and** Jeffery I. Crocker (2005). *Randomness Requirements for Security*. techreport. IETF. URL: https://www.rfc-editor.org/rfc/rfc4086.

Friedman, William F. (1922). *The Index of Coincidence and its Applications in Cryptography*. techreport. Riverbank Laboratories. URL: https://www.nsa.gov/portals/75/documents/news-features/declassified-documents/friedman/Riverbank-Publications/Publications/RB22.pdf.

Guideline for Using Cryptographic Standards in the Federal Government: Directives, Mandates and Policies (2016). techreport SP 800-175B. NIST. URL: https://csrc.nist.gov/publications/detail/sp/800-175b/final.

Houtven, Laurens van (2016). *Crypto 101: the introductory book on cryptography*. URL: https://www.crypto101.io/Crypto101.pdf.

Josefsson, Simon **and** Ilari Liusvaara (2017). *Edwards-Curve Digital Signature Algorithm (EdDSA)*. RFC 8032. URL: https://www.rfc-editor.org/rfc/rfc8032.

Kocher, Paul, Joshua Jaffe **and** Benjamin Jun (1999). **?**Differential Power Analysis**?** **in***Advances in Cryptology—CRYPTO'99*: **pages** 388–397. URL: https://link.springer.com/content/pdf/10.1007/3-540-48405-1_25.pdf.

Kohl, J. **and** C. Neuman (2005). *The Kerberos Network Authentication Service (V5)*. RFC 4120. URL: https://www.rfc-editor.org/rfc/rfc4120.

Krawczyk, Hugo, Mihir Bellare **and** Ran Canetti (1997). *HMAC: Keyed-Hashing for Message Authentication*. RFC 2104. URL: https://www.rfc-editor.org/rfc/rfc2104.

Krawczyk, Hugo **and** P. Eronen (2010). *HMAC-based Extract-and-Expand Key Derivation Function (HKDF)*. RFC 5869. URL: https://www.rfc-editor.org/rfc/rfc5869.

Langley, Adam, Mike Hamburg **and** Sean Turner (2016). *Elliptic Curves for Security*. RFC 7748. URL: https://www.rfc-editor.org/rfc/rfc7748.

Langley, Adam, Yoav Nir **and** David Benjamin (2018). *ChaCha20 and Poly1305 for IETF Protocols*. RFC 8439. URL: https://www.rfc-editor.org/rfc/rfc8439.

Laurie, Ben, Adam Langley **and** Emilia Kasper (2013). *Certificate Transparency*. RFC 6962. URL: https://www.rfc-editor.org/rfc/rfc6962.

MacKay, David J. C. (2003). *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press. URL: https://www.inference.org.uk/itila/book.pdf.

Matsui, Mitsuru (1993). **?**Linear Cryptanalysis Method for DES Cipher**? in***Advances in Cryptology—EUROCRYPT '93***:** URL: https://www.iacr.org/archive/eurocrypt1993/0590/059000386.pdf.

Melnikov, Alan **and** Yoav Nir (2015). *Prohibiting RC4 Cipher Suites*. RFC 7465. URL: https://www.rfc-editor.org/rfc/rfc7465.

Menezes, Alfred J, Paul C van Oorschot **and** Scott A Vanstone (1996). *Handbook of Applied Cryptography*. CRC Press. URL: https://cacr.uwaterloo.ca/hac/.

Mironov, Ilya (2002). **?**(Not so) random shuffles of RC4**? in***Annual International Cryptology Conference***:** URL: https://link.springer.com/chapter/10.1007/3-540-45708-9_24.

Perrin, Trevor **and** Moxie Marlinspike (2016a). *The Double Ratchet Algorithm*. URL: https://signal.org/docs/specifications/doubleratchet/.

— (2016b). *The X3DH Key Agreement Protocol*. URL: https://signal.org/docs/specifications/x3dh/.

Petitcolas, Fabien A. P. (2000). *Least Significant Bit (LSB) Image Steganography*. URL: https://www.cl.cam.ac.uk/~fapp2/steganography/lsb/lsb.html.

Pornin, Thomas (2013). *Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA)*. RFC 6979. URL: https://www.rfc-editor.org/rfc/rfc6979.

Practical Cryptography (2020). *Kasiski Examination for the Vigenère Cipher*. URL: https://practicalcryptography.com/cryptanalysis/stochastic-searching/kasiski-examination/.

Recommendation for Block Cipher Modes of Operation: Methods and Techniques (2001). techreport NIST SP 800-38A. NIST. URL: https://csrc.nist.gov/publications/detail/sp/800-38a/final.

Recommendation for Cryptographic Key Generation (2019). techreport NIST SP 800-133 Rev.2. NIST. URL: https://csrc.nist.gov/publications/detail/sp/800-133/rev-2/final.

Recommendation for Key Management, Part 1: General (2020). techreport NIST SP 800-57 Part 1 Rev.5. NIST. URL: https://csrc.nist.gov/publications/detail/sp/800-57-part-1/rev-5/final.

Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography (2018). techreport NIST SP 800-56A Rev.3. NIST. URL: https://csrc.nist.gov/publications/detail/sp/800-56a/rev-3/final.

Recommendation for Random Bit Generator (RBG) Constructions (2012). techreport SP 800-90C. NIST. URL: https://csrc.nist.gov/publications/detail/sp/800-90c/draft.

Recommendation for Random Number Generation Using Deterministic Random Bit Generators (2015). techreport NIST SP 800-90A Rev.1. NIST. URL: https://csrc.nist.gov/publications/detail/sp/800-90a/rev-1/final.

Recommendation for the Entropy Sources Used for Random Bit Generation (2018). techreport NIST SP 800-90B. NIST. URL: https://csrc.nist.gov/publications/detail/sp/800-90b/final.

Rescorla, Eric (2018). *The Transport Layer Security (TLS) Protocol Version 1.3*. RFC 8446. URL: https://www.rfc-editor.org/rfc/rfc8446.

Rivest, Ronald L, Adi Shamir **and** Leonard Adleman (1978). **?**A Method for Obtaining Digital Signatures and Public-Key Cryptosystems**? in***Communications of the ACM***:** 21.2, **pages** 120–126. URL: https://people.csail.mit.edu/rivest/Rsapaper.pdf.

Schmidt, J"orn-Marc **andothers** (2016). *PKCS #1: RSA Cryptography Specifications Version 2.2*. RFC 8017. URL: https://www.rfc-editor.org/rfc/rfc8017.

Secure Hash Standard (SHS) (2015). techreport FIPS PUB 180-4. NIST. URL: https://csrc.nist.gov/publications/detail/fips/180/4/final.

SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions (2015). techreport FIPS PUB 202. NIST. URL: https://csrc.nist.gov/publications/detail/fips/202/final.

Shannon, Claude E (1948). **?**A Mathematical Theory of Communication**? in***Bell System Technical Journal***:** 27.3, **pages** 379–423. URL: https://people.math.harvard.edu/~ctm/home/text/others/shannon/entropy/entropy.pdf.

Shoup, Victor (2009). *A Computational Introduction to Number Theory and Algebra*. URL: https://shoup.net/ntb/.

The cryptography developers (2024a). *AESGCM — cryptography documentation*. URL: https://cryptography.io/en/latest/hazmat/primitives/aead/#aesgcm (**urlseen** 07/10/2025).

— (2024b). *Ed25519 — cryptography documentation*. URL: https://cryptography.io/en/latest/hazmat/primitives/asymmetric/ed25519/ (**urlseen** 07/10/2025).

— (2024c). *HKDF — cryptography documentation*. URL: https://cryptography.io/en/latest/hazmat/primitives/key-derivation-functions/#hkdf (**urlseen** 07/10/2025).

— (2024d). *Symmetric Encryption — cryptography documentation*. URL: https://cryptography.io/en/latest/hazmat/primitives/symmetric-encryption/ (**urlseen** 07/10/2025).

Transitioning the Use of Cryptographic Algorithms and Key Lengths (2019). techreport NIST SP 800-131A Rev.2. NIST. URL: https://csrc.nist.gov/publications/detail/sp/800-131a/rev-2/final.

Westfeld, Andreas (2001). **?**F5—A Steganographic Algorithm**? in***Information Hiding*: URL: https://link.springer.com/chapter/10.1007/3-540-45496-9_21.