SKRIPSI

IMPLEMENTASI *HYBRID CRYPTOGRAPHY* DAN *SECRET SHARING* UNTUK MENJAGA KERAHASIAAN DATA PADA *CLOUD STORAGE*

Disusun dan diajukan oleh:

AZZAHRA BELADINA SHAFF
D121 20 1017

PROGRAM STUDI SARJANA TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS HASANUDDIN
GOWA
2024

LEMBAR PENGESAHAN SKRIPSI

IMPLEMENTASI *HYBRID CRYPTOGRAPHY* DAN *SECRET SHARING* UNTUK MENJAGA KERAHASIAAN DATA PADA *CLOUD STORAGE*

Disusun dan diajukan oleh

Azzahra Beladina Shaff
D121201017

Telah dipertahankan di hadapan Panitia Ujian yang dibentuk dalam rangka
Penyelesaian Studi Program Sarjana Program Studi Teknik Informatika
Fakultas Teknik Universitas Hasanuddin
Pada tanggal 28 November 2024
dan dinyatakan telah memenuhi syarat kelulusan

Menyetujui,

Pembimbing Utama,

Dr. Eng. Ady Wahyudi Paundu, S.T., M.T
NIP 19750313 200912 1 003

Ketua Program Studi,

Prof. Dr. Ir. Indrabayu, S.T., M.T., M.Bus.Sys., IPM.ASEAN.Eng.
NIP 19750716 200212 1 004

PERNYATAAN KEASLIAN

Yang bertanda tangan dibawah ini;
Nama : Azzahra Beladina Shaff
NIM : D121201017
Program Studi : Teknik Informatika
Jenjang : S1

Menyatakan dengan ini bahwa karya tulisan saya berjudul

Implementasi *Hybrid cryptography* dan *Secret Sharing* untuk Menjaga Kerahasiaan Data pada *Cloud Storage*

Adalah karya tulisan saya sendiri dan bukan merupakan pengambilan alihan tulisan orang lain dan bahwa skripsi yang saya tulis ini benar-benar merupakan hasil karya saya sendiri.

Semua informasi yang ditulis dalam skripsi yang berasal dari penulis lain telah diberi penghargaan, yakni dengan mengutip sumber dan tahun penerbitannya. Oleh karena itu semua tulisan dalam skripsi ini sepenuhnya menjadi tanggung jawab penulis. Apabila ada pihak manapun yang merasa ada kesamaan judul dan atau hasil temuan dalam skripsi ini, maka penulis siap untuk diklarifikasi dan mempertanggungjawabkan segala resiko.

Segala data dan informasi yang diperoleh selama proses pembuatan skripsi, yang akan dipublikasi oleh Penulis di masa depan harus mendapat persetujuan dari Dosen Pembimbing.

Apabila dikemudian hari terbukti atau dapat dibuktikan bahwa sebagian atau keseluruhan isi skripsi ini hasil karya orang lain, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Gowa, 29 November 2024

Yang Menyatakan

Azzanra Beladina Shaff

ABSTRAK

AZZAHRA BELADINA SHAFF. *Implementasi Hybrid cryptography dan Secret Sharing untuk Menjaga Kerahasiaan Data pada Cloud Storage* (dibimbing oleh Ady Wahyudi Paundu)

Data-data penting yang disimpan di *cloud* umumnya berisi informasi sensitif, baik itu data pribadi maupun data perusahaan yang jika bocor atau disalahgunakan dapat berdampak negatif signifikan. Tidak menutup kemungkinan, meskipun telah diamankan, data yang disimpan di *cloud* dapat dicuri, terlebih lagi apabila penyerang mampu meretas kunci enkripsi data sehingga data asli dapat jatuh ditangan penyerang. Untuk melindungi semua layanan dan manfaat yang ditawarkan oleh komputasi awan dan internet, peningkatan akan keamanan data sangatlah penting.

Tujuan penelitian ini adalah mengembangkan sistem yang mengimplementasikan *hybrid cryptography* dan *secret sharing* untuk mengamankan *file* sebelum dimasukkan ke dalam *cloud storage* dan menganalisis fungsionalitas dari penerapan *hybrid criptography* dan *secret sharing* yang diimplementasikan.

Metode yang digunakan melibatkan pengembangan sebuah sistem menggunakan algoritma kriptografi hibrida, AES-128 dan RSA serta dipadukan dengan teknik *Secret* Sharing. Data yang digunakan dalam pengujian sistem merupakan data laporan keuangan dengan format .docx, dengan ukuran bervariasi mulai dari 17 KB hingga 10000 KB. Data yang di-*split* terdiri dari 16 *byte* pertama, yang disimpan sebagai *share* di Google Drive, Dropbox, dan OneDrive.

Hasil implementasi sistem menunjukkan bahwa ukuran *file* setelah enkripsi dan dekripsi tidak berubah, memastikan efisiensi penyimpanan. Waktu untuk enkripsi dan dekripsi akan meningkat seiring peningkatan ukuran *file*. Waktu rata-rata untuk enkripsi data di bawah 100 KB adalah 0,254846 detik, 100-999 KB adalah 2,804594, dan 1000-10000 KB adalah 320,270371 detik. Sedangkan untuk dekripsi adalah 0,353689 detik, 3,760523 detik, hingga 328,910659 detik. Proses *split* untuk setiap *share* berlangsung dengan cepat, sekitar 0,0072 detik untuk semua kategori ukuran *File*, dan proses rekonstruksi meningkat seiring ukuran *file*, dengan rata-rata 0,002095 detik. Meskipun *file* terenkripsi dibagi menjadi beberapa bagian, ukuran *share* yang dihasilkan sangat kecil, sekitar 42-44 *byte*, yang tidak membebani kapasitas penyimpanan di *cloud*.

Kata Kunci: keamanan data, AES-128, RSA, *Secret Sharing*, penyimpanan *cloud*

ABSTRACT

Azzahra Beladina Shaff. *Implementation of Hybrid cryptography and Secret Sharing to Ensure Data Confidentiality in Cloud Storage* (supervised by Ady Wahyudi Paundu)

Sensitive data stored in the cloud generally includes personal or corporate information, which, if leaked or misused, could have significant negative impacts. It is possible that even though the data is secured, it could still be stolen, especially if an attacker is able to breach the encryption keys, allowing the original data to fall into the wrong hands. To protect all the services and benefits offered by cloud computing and the internet, enhancing data security is crucial.

The objective of this research is to develop a system that implements hybrid cryptography and secret sharing to secure files in cloud storage, and to analyze the functionality of the implementation of hybrid cryptography and secret sharing.

The method used involves developing a system using a hybrid cryptography algorithm, AES-128 and RSA, combined with the Secret Sharing technique. The data used in the system testing is financial report data in .docx format, with sizes ranging from 17 KB to 1,901 KB. The split data consists of the first 16 bytes, which are stored as shares in Google Drive, Dropbox, and OneDrive.

The results of the system implementation show that the File size after encryption and decryption does not change, ensuring storage efficiency. The time required for encryption and decryption increases as the File size increases. The average encryption time for data under 100 KB is 0.254846 seconds, for 100-999 KB is 2.804594 seconds, and for 1000-10000 KB is 320,270371 seconds. For decryption, the time is 0.353689 seconds, 3.760523 seconds, and up to 328.910659 seconds. The splitting process for each share is fast, around 0.0072 seconds for all File size categories, while the reconstruction process increases with File size, averaging 0,002095 seconds. Although the encrypted files are split into several parts, the size of the generated shares is very small, around 42-44 bytes, which does not burden cloud storage capacity.

Keywords: data security, AES-128, RSA, Secret Sharing, cloud storage

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR SINGKATAN DAN ARTI SIMBOL

Lambang/Singkatan	Arti dan Keterangan
AES	*Advanced Encryption Standard*
ASCII	*American Standard Code for Information Interchange*
API	*Application Programming Interface*
DOCX	*Document Extended*
GB	*GigaByte*
HTML	*HyperText Markup Language*
iOS	*iPhone Operating System*
MD4	*Message Digest Algorithm 4*
MIT	*Massachusetts Institute of Technology*
NIST	*National Institute of Standards and Technology*
OCR	*Optical Character Recognition*
PDF	*Portable Document Format*
Rcon	*Round Constant*
RSA	*Rivest-Shamir-Adleman*
SHS	*Secure Hash Standard*
SQLite	*Structured Query Language Lite*
TB	*TeraByte*
TXT	*Text*
USB	*Universal Serial Bus*
XML	*eXtensible Markup Language*
XOR	*Exclusive OR*
mod	Operasi modulus
N_r	Jumlah total putaran dalam proses AES
GF	*Galois Field*
$\varphi(n)$	*totient Euler*
e	Eksponen publik
e, n	Eksponen publik dan modulus dalam RSA (kunci publik)
d, n	Eksponen privat dan modulus dalam RSA (kunci privat)
C	*Ciphertext* (pesan hasil enkripsi)

M	*Plaintext* (pesan asli sebelum enkripsi)

DAFTAR LAMPIRAN

KATA PENGANTAR

Alhamdulillah, puji syukur kehadirat Allah SWT atas segala limpahan rahmat, hidayah, dan karunia-Nya sehingga penulis dapat menyelesaikan skripsi iniSelama proses penyusunan skripsi, penulis mendapatkan banyak bantuan, bimbingan, dan dukungan dari berbagai pihak. Oleh karena itu, penulis ingin mengucapkan terima kasih yang tulus kepada:

1. Kedua orang tua penulis, Bapak Ihwan dan Ibu Suarti, Adik-adik penulis, Mikala Muh. Izzati, dan Shafiyah Ummu Umarah, serta seluruh keluarga penulis yang telah memberikan dukungan moral maupun materi, serta menjadi sumber kekuatan dan do'a yang tiada henti.
2. Bapak Dr. Eng. Ady Wahyudi Paundu, S.T., M.T., selaku dosen pembimbing yang telah memberikan bimbingan, arahan, serta waktu dalam membantu menyelesaikan skripsi ini, serta mendampingi dalam proses penyusunan dan penelitian hingga tahap evaluasi dan penilaian akhir. Terima kasih atas kesabaran dan dedikasinya dalam membimbing penulis selama perjalanan ini.
3. Segenap Dosen dan Staf Departemen Teknik Informatika Fakultas Teknik Universitas Hasanuddin yang telah memberikan ilmu, pengetahuan, serta pengalaman berharga kepada penulis selama masa perkuliahan.
4. Zid Irsyadin Sartono Wijaogy, yang selalu menemani penulis, memberikan dukungan dan semangat, serta mendengar semua keluhan yang penulis hadapi dan memberikan bantuan akan hal itu.
5. Teman seperjuangan penulis Sitti Zakiyah Nurkhalisah, Nurza Zahira Faried, Husnul Khatimah, Atira Septiara, dan Mutia Rahman atas kebersamaan yang telah diberikan dari awal perkuliahan hingga masa pengerjaan skripsi.
6. Tri Indah Wahyuningsi, Aldilah Rezki Rhamadani Syahsir, Agunawan Ali Nur, Muh. Arfan Bahrun, dan Dimas Permana, atas dukungan serta motivasi yang diberikan, yang menjadi dorongan bagi penulis untuk terus berusaha menyelesaikan skripsi ini dengan baik.

7. Segenap keluarga Lab Ubicon yang telah menjadi tempat bagi penulis untuk berbagi pengalaman dan berkembang secara akademik. Terima kasih atas dukungan dan suasana kerja yang kondusif sehingga penulis dapat menyelesaikan skripsi ini dengan baik.
8. Seluruh teman-teman KKNT 110 Smart Village 2 Barru atas kebersamaan yang masih terjaga hingga sekarang.
9. Teman-teman “RE20LVER” atas kebersamaan yang telah dilalui selama masa perkuliahan.
10. Seluruh pihak yang tidak dapat disebutkan satu per satu yang tanpa disadari telah memberikan semangat dan dukungan dalam menyelesaikan tugas akhir ini dengan baik..

Penulis menyadari bahwa skripsi ini masih jauh dari sempurna, oleh karena itu kritik dan saran yang membangun sangat penulis harapkan demi perbaikan di masa mendatang. Akhir kata, semoga skripsi ini dapat memberikan manfaat bagi kita semua dan berkontribusi bagi kemajuan ilmu pengetahuan.

Gowa, 03 Oktober 2024

Penulis

BAB I
PENDAHULUAN

1.1 Latar Belakang

Cloud computing adalah teknologi yang mengelola sumber daya komputasi lewat jaringan internet, sehingga dapat menghubungkan antara komputer satu dengan yang lain (Berisha et al., 2022). Penggunaan *cloud computing* sendiri telah mengalami pertumbuhan yang signifikan dalam beberapa tahun terakhir. Hal ini mengubah cara penyimpanan dan akses data oleh organisasi dan individu. Dengan memanfaatkan infrastruktur internet, *cloud computing* memberikan kemudahan dalam hal skalabilitas, fleksibilitas, dan efisiensi dalam pengelolaan data. Meskipun *cloud computing* memberikan kemudahan dalam hal penyimpanan dan akses data melalui internet, hal ini juga membawa tantangan besar dalam aspek keamanan (Ramalinda et al., 2024).

GoodFirms menyurvei 648 orang di berbagai negara, dengan responden terdiri dari berbagai kelompok umur, menunjukkan bahwa sekitar 65,28% responden menggunakan penyimpanan *cloud* sebagai penyimpanan utama mereka, dengan 92,59% masyarakat menggunakannya untuk menyimpan data kantor. Lebih lanjut, 54,62% di antara mereka menggunakan setidaknya tiga layanan penyimpanan *cloud* yang berbeda dengan Google Drive, Dropbox, dan Microsoft OneDrive sebagai layanan yang paling banyak digunakan. Meskipun penyedia layanan *cloud* menawarkan jaminan keamanan, para ahli sering menyatakan bahwa penyimpanan *cloud* yang sepenuhnya aman belum ada. 78.24% orang bahkan menyatakan privasi dan keamanan data masih menjadi kekhawatiran utama terkait penggunaan penyimpanan *cloud* (Sebastian, 2024). Kekhawatiran ini diperjelas dengan laporan keamanan *cloud* Thales 2023 yang menyebutkan bahwa 39% perusahaan mengalami kebocoran data dalam lingkungan *cloud* mereka pada tahun 2022, meningkat 4% dari tahun sebelumnya (Thales, 2023).

Data-data penting yang disimpan di *cloud* umumnya berisi informasi sensitif, baik itu data pribadi maupun data perusahaan yang jika bocor atau disalahgunakan dapat berdampak negatif signifikan. Salah satu data penting perusahaan yaitu laporan keuangan. Laporan keuangan tidak hanya mencerminkan

kesehatan finansial perusahaan namun juga menjadi faktor penentu dalam berbagai keputusan. Dengan meningkatnya ancaman serangan siber, kebocoran data, pencurian informasi, dan kerusakan reputasi dapat terjadi. Hal ini membuat peningkatan keamanan data menjadi sangat penting (Hariharan, 2021).

Untuk melindungi semua layanan dan manfaat yang ditawarkan oleh komputasi awan dan Internet, keamanan data sangatlah penting. Kerahasiaan data dapat dicapai menggunakan teknik kriptografi, yaitu enkripsi dan dekripsi. Kriptografi bertujuan untuk menyediakan sekumpulan fungsi keamanan yang dapat memastikan kerahasiaan sistem (Thabit et al., 2022). Salah satu penelitian yang menerapkan ilmu kriptografi pada sistem berbasis *cloud computing* yaitu penelitian Sa'idu Sani (2022) yang berjudul *"A Comparative Analysis of Cryptographic Algorithms: AES & RSA and Hybrid Algorithm for Encryption and Decryption"*. Penelitian ini berhasil mengamankan data dan menunjukkan bahwa perpaduan antara AES dan RSA dapat menjadi solusi untuk kekurangan masing-masing algoritma. Namun, meskipun telah dienkripsi, data yang disimpan ini dapat dicuri, terlebih lagi apabila penyerang mampu meretas kunci enkripsi data sehingga data asli dapat jatuh ditangan penyerang.

Salah satu cara penyerang meretas kunci enkripsi data yaitu dengan serangan *brute force*. Serangan *brute force* adalah teknik yang digunakan untuk menebak kata sandi atau kunci enkripsi dengan mencoba semua kemungkinan kombinasi karakter secara berurutan hingga menemukan kombinasi yang tepat (Arradian, 2024). IBM melaporkan, sejumlah peretas secara konsisten menargetkan sistem yang sama setiap hari, terkadang berlangsung selama beberapa bulan atau bahkan bertahun-tahun (Swinhoe, 2020). Dari Januari hingga Juni 2021, Kaspersky mencatat total 20.847.706 percobaan serangan *brute force* yang menargetkan pengguna di Indonesia yang memiliki Microsoft RDP terpasang di desktop mereka, melonjak signifikan dibandingkan dengan 16.854.459 pada paruh pertama 2020 (Novianty, 2021). Selanjutnya, selama periode Januari hingga Desember 2023, Kaspersky mendeteksi 61.374.948 upaya serangan *Brute force* di Asia Tenggara, dengan Indonesia di urutan kedua (Arradian, 2024).

Berdasarkan masalah tersebut, penulis mengusulkan sebuah sistem yang mampu mengamankan data dengan teknik enkripsi menggunakan kriptografi

hibrida namun dipadukan dengan teknik *Shamir's Secret Sharing*. Peningkatan serangan *brute force* menegaskan perlunya pendekatan keamanan yang lebih kuat dalam melindungi data sensitif. Salah satu metode yang dapat diterapkan selain enkripsi adalah *secret sharing*, di mana data enkripsi dibagi menjadi beberapa bagian dan disimpan di berbagai lokasi terpisah. Dengan metode ini, meskipun penyerang berhasil mengetahui kunci enkripsi melalui serangan *brute force*, mereka tidak akan dapat langsung mengakses data yang dilindungi. Hal ini karena kunci yang diperlukan untuk mendekripsi data tidak tersimpan dalam satu kesatuan, melainkan tersebar, sehingga penyerang masih memerlukan beberapa bagian dari *secret sharing* untuk bisa mengakses data sepenuhnya. Teknik *secret sharing* juga dirancang sedemikian rupa sehingga, meskipun ada bagian pecahan yang hilang atau rusak, maka nilai awal dari data tetap bisa dikembalikan (Fachry et al., 2018). Sistem ini akan mengenkripsi data menggunakan algoritma AES (*Advanced Encryption Standard)*, dan kunci AES dienkripsi menggunakan algoritma RSA (*Rivest-Shamir-Adleman).* Setelah data dienkripsi, teknik *Shamir's Secret Sharing* diaplikasikan untuk membagi data terenkripsi menjadi beberapa bagian atau *'shares'*. Teknik ini akan memecah data menjadi tiga bagian dan menyimpannya ke dalam tiga *cloud service provider* yang berbeda, yaitu Google Drive, Dropbox dan Microsoft OneDrive. Penelitian ini diharapkan dapat mengurangi risiko keamanan yang terkait dengan penyimpanan data di *cloud*, memberikan perlindungan yang komprehensif, memastikan bahwa data tetap aman dan hanya dapat diakses oleh pihak yang berwenang.

1.2 Rumusan Masalah

Adapun masalah pada penelitian ini, berdasarkan latar belakang yang telah diuraikan, adalah:

1. Bagaimana mengimplementasikan *hybrid cryptography* algoritma AES dan RSA serta teknik *secret sharing* untuk mengamankan *file* sebelum dimasukkan ke dalam *cloud storage*?
2. Bagaimana fungsionalitas dari penerapan *hybrid cryptography* dan *secret sharing*?

3. Bagaimana parameter dari penerapan *hybrid cryptography* dan *secret sharing?*

1.3 Tujuan Penelitian/Perancangan

Adapun tujuan dari penelitian ini adalah sebagai berikut:

1. Mengimplementasikan *hybrid cryptography* dan *secret sharing* untuk mengamankan *file* sebelum dimasukkan ke dalam *cloud storage.*
2. Menganalisis fungsionalitas penerapan *hybrid cryptography* dan *secret sharing.*
3. Menganalisis parameter dari penerapan *hybrid cryptography* dan *secret sharing.*

1.4 Manfaat Penelitian/Perancangan

Penelitian ini diharapkan dapat memberikan manfaat sebagai berikut:

1. Memberikan solusi untuk meningkatkan keamanan data pada *cloud* dengan memanfaatkan teknik enkripsi, dekripsi, dan *secret sharing.*
2. Dapat dijadikan sumber rujukan bagi peneliti lain yang berkeinginan mengambil topik yang sejalan atau serupa.

1.5 Ruang Lingkup Penelitian

Pada penelitian ini akan dibatasi ruang lingkup pembahasannya yaitu:

1. Menggunakan bahasa pemrograman Python dalam pengembangan *website*.
2. Menggunakan Django untuk pembuatan dan pengelolaan logika *server-side* sebagai kerangka kerja utama.
3. Menggunakan db.sqlite3 sebagai *database default* untuk menyimpan dan mengolah data.
4. Penelitian menggunakan tiga layanan *cloud* yang berbeda, yaitu Google Drive, Dropbox, dan OneDrive.
5. Data yang akan digunakan pada uji coba sistem adalah dokumen laporan keuangan dengan format *file* .docx.

BAB II
TINJAUAN PUSTAKA

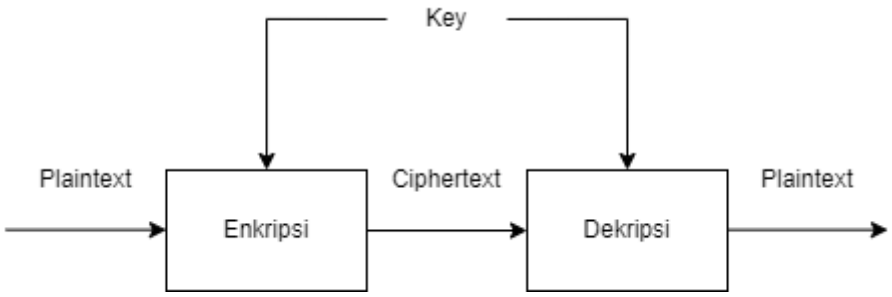
2.1 Kriptografi

Kriptografi berasal dari bahasa Yunani, yaitu kripto dan graphia, dimana kripto diartikan sebagai *secret* (rahasia) dan graphia sebagai *writing* (tulisan). Kriptografi adalah ilmu yang mempelajari teknik-teknik matematika yang berkaitan dengan aspek keamanan informasi, seperti kerahasiaan data, keabsahan data, integritas data, dan autentikasi data. Kriptografi bekerja dimana data diacak menggunakan suatu kunci enkripsi menjadi sesuatu yang sulit dibaca oleh orang yang tidak memiliki kunci dekripsi (Amin, 2017). Terdapat komponen-komponen utama dalam kriptografi, di antaranya:

1. *Plaintext* (teks biasa): data asli atau pesan yang hendak dikirim sebelum proses enkripsi, dengan kata lain informasi ini masih dalam bentuk teks yang dapat dibaca manusia.
2. *Ciphertext* (teks sandi) : hasil dari proses enkripsi, berupa data atau pesan yang tidak dapat dibaca tanpa proses dekripsi.
3. *Key* (kunci): elemen penting dalam kriptografi yang digunakan untuk enkripsi dan dekripsi. Kunci dapat berupa kunci privat ataupun kunci publik yang masing-masing memiliki peran dalam proses pengamanan data.
4. Enkripsi: proses menyembunyikan sebuah data dengan cara mengubah *plaintext* menjadi *ciphertext* sehingga sulit dimengerti oleh manusia.
5. Dekripsi: proses mengubah kembali *ciphertext* menjadi informasi semula menggunakan kunci, sehingga data dapat dibaca kembali dalam bentuk aslinya (Nanda et al., 2023).

2.1.1 Kriptografi simetris

Kriptografi simetris juga sering disebut kriptografi klasik, sebab menggunakan kunci yang sama pada proses enkripsi dan dekripsinya.

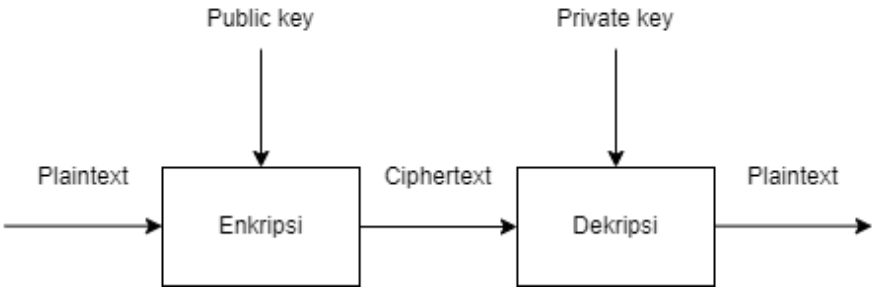


Gambar 1 Kriptografi simetris

Berdasarkan Gambar 1 *plaintext* dienkripsi menggunakan sebuah kunci sehingga menghasilkan *ciphertext.* Kunci yang sama juga digunakan untuk proses dekripsi *ciphertext* sehingga dapat menghasilkan *file* semula (G. G. Putri et al., 2018).

2.1.2 Kriptografi asimetris

Kriptograsi asimetris juga dikenal sebagai kriptografi kunci publik, dimana kunci yang digunakan untuk melakukan enkripsi dan dekripsinya berbeda. *Plaintext* dienkripsi menggunakan *public key* sehingga menghasilkan *ciphertext*. Untuk mendekripsi *ciphertext*, digunakan *private key* sehingga dapat menghasilkan *file* semula (G. G. Putri et al., 2018).

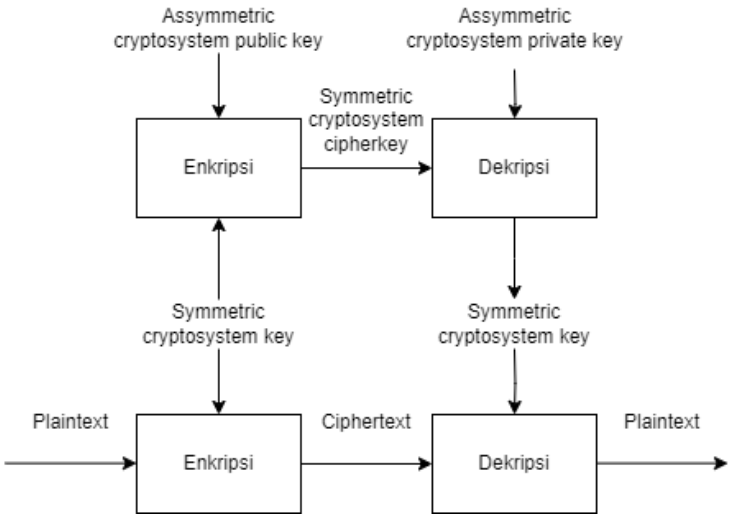


Gambar 2 Kriptografi asimetris

2.1.3 Kriptografi hibrida

Kriptografi hibrida menggunakan kombinasi algoritma enkripsi simetris dan asimetris untuk meningkatkan keamanan dalam proses enkripsi dan dekripsi data atau pesan. Algoritma simetris digunakan untuk mengenkripsi data atau pesan dengan kunci simetris yang efisien dan cepat, sedangkan algoritma asimetris digunakan untuk mengenkripsi kunci

simetris tersebut menggunakan kunci publik penerima. Dengan menggunakan kriptografi hibrida, data atau pesan dapat diamankan dengan enkripsi yang kuat selama pengiriman, dan hanya pemegang kunci privat yang dapat mengakses serta mendekripsi data atau pesan tersebut.



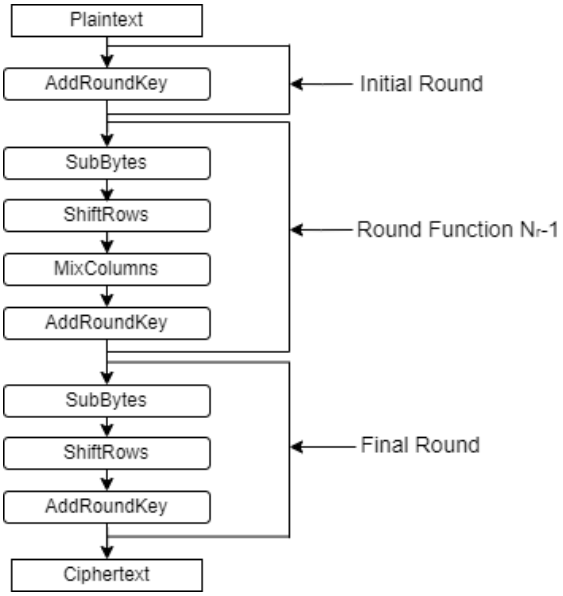
Gambar 3 Kriptografi hibrida

Pada kriptografi hibrida, algoritma simetris digunakan untuk enkripsi dan dekripsi *plaintext* dengan kunci simetris, sedangkan algoritma asimetris digunakan untuk mengenkripsi kunci simetris tersebut menggunakan kunci publik penerima, dan mendekripsi kunci simetris menggunakan kunci privat. Hasil dekripsi kunci simetris yang nantinya akan digunakan untuk melakukan dekripsi *ciphertext* sehingga menghasilkan *file* semula (Fatimatuzzahro', 2021).

2.2 *Advanced Encryption Standard* (AES)

AES adalah algoritma simetris yang dikembangkan oleh *National Institute of Standards and Technology* (NIST) di Amerika Serikat. Algoritma ini dirancang untuk melindungi informasi dengan menggunakan kunci rahasia yang sama baik untuk proses enkripsi maupun dekripsinya. Algoritma AES dirancang untuk memberikan keamanan yang tinggi dan efisiensi dalam proses enkripsi dan dekripsi, serta menjadi standar enkripsi yang digunakan secara luas dalam berbagai organisasi maupun aplikasi, seperti keamanan jaringan, perlindungan data dalam perangkat lunak, serta penyimpanan data secara aman (Oktaviani et al., 2023).

Algoritma AES juga biasa disebut dengan algoritma Rijndael. Algoritma ini menggunakan proses substitusi, permutasi, dan sejumlah putaran yang diterapkan pada setiap blok yang akan dienkripsi ataupun didekripsi. Pada setiap putaran, algoritma AES menggunakan kunci yang berbeda, yang disebut dengan *round key.* Ukuran blok untuk AES adalah 128 bit (16 *byte*). Algoritma AES mendukung panjang kunci mulai dari 128 bit hingga 256 bit dengan peningkatan 32 bit. Karena AES menetapkan panjang kunci menjadi 128, 192, dan 256 bit, maka dikenal varian AES-128, AES-192, dan AES-256 (G. G. Putri et al., 2018).

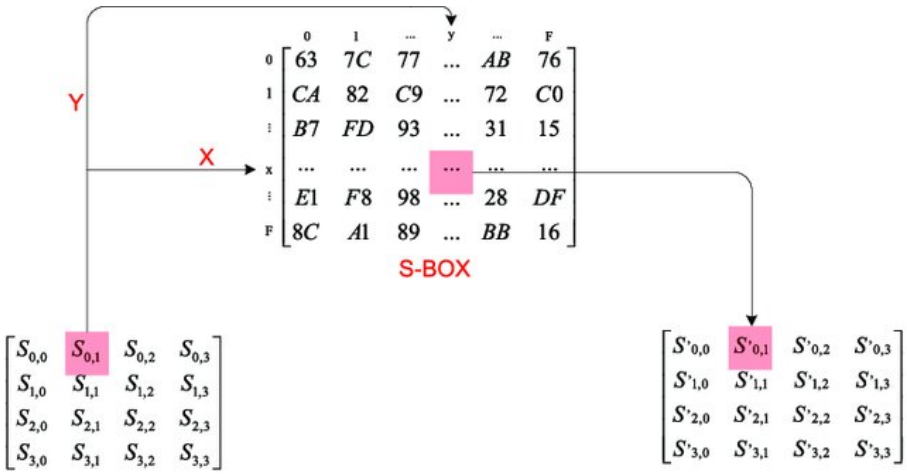


Gambar 4 Proses enkripsi AES

Proses enkripsi AES dimulai dari *initial round*, yang mana terjadi transformasi *AddRoundKey* yang melakukan operasi XOR antara *state* (matriks 4×4 dari *byte plaintext)* dengan kunci (matriks 4×4 dari *byte* kunci). Proses selanjutnya mencakup transformasi *SubBytes*, *ShiftRows*, *MixColumns,* dan *AddRoundKey*, yang dilakukan sebanyak $Nr - 1$. Nr menunjukkan jumlah total putaran yang dijalankan dalam proses enkripsi AES. Banyaknya putaran ini bergantung pada panjang kunci yang digunakan. Kunci sepanjang 128 bit membutuhkan 10 putaran, kunci 192 bit membutuhkan 12 putaran, dan kunci 256 bit membutuhkan 14 putaran. Setelah proses ini selesai, proses selanjutnya yaitu mengulangi *SubBytes*, *ShiftRows*, dan *AddRoundKey* sebanyak satu kali yang akan menghasilkan *ciphertext* (Fachry et al., 2018).

2.2.1 *SubBytes*

SubBytes merupakan operasi yang akan melakukan substitusi dengan mengganti setiap *byte* pada *state* dengan *byte* yang terdapat dalam sebuah tabel yang disebut dengan *S-box* (W. C. U. Putri et al., 2023). Kotak *S-box* adalah matriks berukuran 16×16 yang berisi nilai-nilai heksadesimal pada setiap elemennya. Untuk mengganti nilai pada elemen *state* dengan nilai yang terdapat pada kotak *S-box*, setiap nilai pada *state* dipecah menjadi dua bagian, yaitu bagian kiri dan bagian kanan. Bagian kiri direpresentasikan sebagai indeks *x* dan bagian kanan sebagai indeks *y* untuk menemukan nilai yang sesuai pada kotak *S-box* (Fachry et al., 2018).



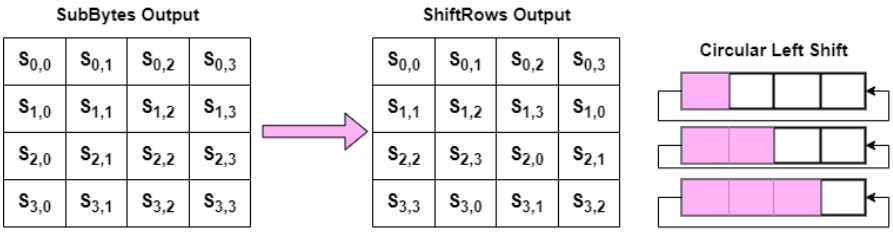
Gambar 5 Proses *SubBytes*
Sumber: (Atigani dkk, 2021)

Dalam proses *SubBytes*, setiap *byte* dari *state matrix* (matriks status) yang mewakili blok *plaintext* akan digantikan dengan nilai yang sesuai dari *S-box*. Nilai *byte* 'S[0,1]' pada *state matrix* digantikan dengan nilai 'S'[0,1]' yang diambil dari *S-box* berdasarkan koordinat *x* dan *y* yang sesuai.

2.2.2 *ShiftRows*

ShiftRows, sesuai dengan namanya, adalah sebuah proses yang melakukan *shift* (pergeseran) pada setiap elemen dalam blok atau tabel (Hts et al., 2023). Proses *ShiftRows* bekerja pada setiap baris dalam tabel *state*. Dalam proses ini, *byte* pada tiga baris terakhir (baris 1, 2, dan 3) akan digeser ke kiri dengan jumlah putaran yang berbeda. Adapun jumlah baris yang digeser, sebagai berikut (W. C. U. Putri et al., 2023):

1. Baris 0 tidak dilakukan pergeseran
2. Baris 1 dilakukan pergeseran sebanyak 1 kali
3. Baris 2 dilakukan pergeseran sebanyak 2 kali
4. Baris 3 dilakukan pergeseran sebanyak 3 kali



Gambar 6 Proses *ShiftRows*

3.2.3 *MixColumns*

Proses *MixColumns* bekerja pada setiap kolom dalam tabel *state* dengan memperlakukan setiap kolom sebagai polinomial dengan koefisien dalam $GF(2^8)$, dimana setiap kolom dikalikan dengan polinomial $a(x)$. Transformasi ini diterapkan pada sebagian besar putaran, kecuali pada putaran terakhir, dimana tahap *MixColumns* tidak dilakukan (W. C. U. Putri et al., 2023).

$$\begin{bmatrix} S'_{0,j} \\ S'_{1,j} \\ S'_{2,j} \\ S'_{3,j} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \times \begin{bmatrix} S_{0,j} \\ S_{1,j} \\ S_{2,j} \\ S_{3,j} \end{bmatrix}$$

Gambar 7 Proses *MixColumns*

Pada Gambar 7 tabel sebelah kiri diambil dari *state matrix* dimana setiap elemen dilambangkan sebagai S_i, j, dengan i menunjukkan baris dan j menunjukkan kolom. Tabel tengah merupakan matriks *MixColumns* yang bernilai konstan dan digunakan untuk memodifikasi setiap kolom dari *state matrix* melalui operasi aritmatika dalam *Galois Field.* Proses ini melibatkan perkalian antara setiap kolom dari matriks status S dengan matriks *MixColumns.* Untuk setiap kolom j pada *state matrix*, hasilnya adalah kolom baru S' yang dihasilkan dari perkalian tersebut (Sofian et al., 2019).

3.2.4 *AddRoundKey*

Pada setiap putaran *AddRoundKey* dilakukan teknik yang berbeda-beda melalui teknik yang disebut ekspansi kunci. Dalam proses ini, nilai dari kolom terakhir *state* kunci diambil untuk menjalankan *rot word,* yaitu penggeseran nilai ke atas sebanyak satu kali.

$$RotWord(B_0, B_1, B_2, B_3) = (B_1, B_2, B_3, B_0) \quad (1)$$

Hasil dari *RotWord* kemudian disubtitusi menggunakan S-Box melalui transformasi *SubBytes.* Pada operasi ini, 8 bit dari *subkey* disubstitusikan dengan nilai yang ada pada tabel S-Box.

$$SubWord(B_0, B_1, B_2, B_3) = ({B_0}', {B_1}', {B_2}', {B_3}') \quad (2)$$

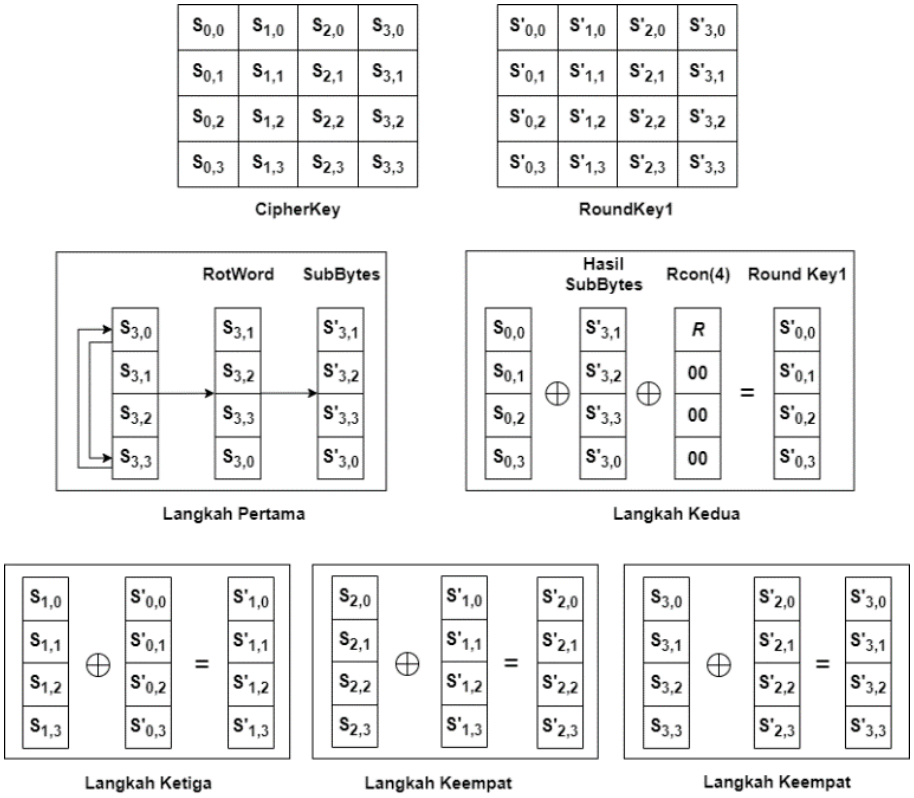
Dengan $B_i' = SubBytes(B_i), i = 0,1,2,3$. Hasil dari transformasi ini kemudian dioperasikan dengan XOR bersama kolom pertama dari matriks kunci pertama dan sebuah *round constant. Round Constant* adalah komponen tetap dari perhitungan kunci ekspansi *routine* dalam tiap putaran. Nilai dari *Rcon* ditunjukkan pada Tabel 1.

Tabel 1. Nilai *Rcon*

Round	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
***Rcon*[]**	01	02	04	08	10	20	40	80	1b	36
	00	00	00	00	00	00	00	00	00	00
	00	00	00	00	00	00	00	00	00	00
	00	00	00	00	00	00	00	00	00	00

Sumber: W. C. U. Putri et al., (2023)

Operasi XOR ini menghasilkan kolom pertama dari matriks kunci yang baru. Untuk mendapatkan tiga kolom berikutnya, kolom baru yang dihasilkan sebelumnya akan di-XOR dengan kolom yang sesuai urutannya (Fachry et al., 2018). Adapun untuk proses *AddRoundKey* dapat dilihat pada Gambar 8.



Gambar 8 Proses ekspansi kunci

Proses yang telah dijelaskan di ulang sebanyak $Nr - 1$. Untuk proses selanjutnya mencakup *SubBytes, ShiftRows,* dan *AddRoundKey* sebanyak satu kali sehingga menghasilkan *ciphertext.* Adapun proses dekripsi merupakan kebalikan dari proses enkripsi. Transformasi dibalik dan diimplementasikan ke arah yang berlawanan untuk mendapatkan *inverse cipher* (W. C. U. Putri et al., 2023).

3.2.5 Contoh Perhitungan AES-128

Perhitungan AES-128 ini dilakukan dengan melakukan enkripsi pada pesan “INIPESANENKRIPSI” dengan kunci “CONTOHKUNCIAESKU”. Langkah pertama yang dilakukan adalah mengonversi pesan dan kunci ke dalam bilangan heksadesimal. Proses ini dapat dilakukan dengan menggunakan tabel ASCII (*American Standard Code for Information Interchange*) untuk mendapatkan representasi heksadesimal dari setiap karakter.

Dec = Decimal; Hex = Hexadecimal; Char = Character

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□

Gambar 9 Tabel ASCII

Hasil dari konversi pesan dapat dilihat pada Tabel 2 dan 3.

Tabel 2. Konversi pesan ke heksadesimal

I	N	I	P	E	S	A	N	E	N	K	R	I	P	S	I
49	4E	49	50	45	53	41	4E	45	4E	4B	52	49	50	53	49

Tabel 3. Konversi kunci ke heksadesimal

C	O	N	T	O	H	K	U	N	C	I	A	E	S	K	U
43	4F	4E	54	4F	48	4B	55	4E	43	49	41	45	53	4B	55

Jika telah mendapatkan nilai heksadesimal dari pesan dan kunci, selanjutnya dilakukan pembentukan matriks *state* dan matriks kunci. AES bekerja dengan matriks 4×4 yang berisi *byte* dalam format tersebut.

49	45	45	49
4E	53	4E	50
49	41	4B	53
50	4E	52	49

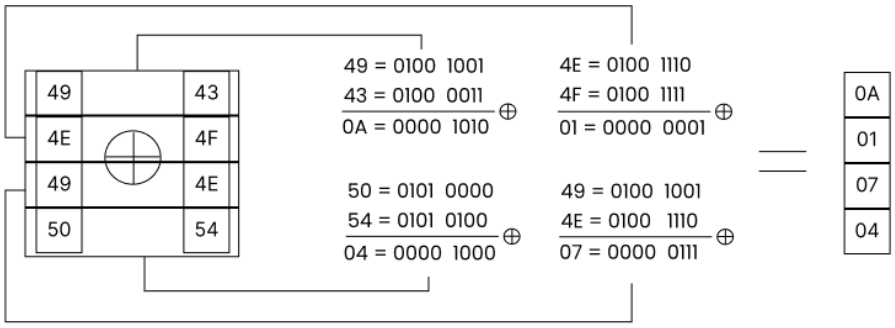
43	4F	4E	45
4F	48	43	53
4E	4B	49	4B
54	55	41	55

Gambar 10 Matriks *state* dan matriks kunci

Tabel pertama dalam Gambar 10 menunjukkan matriks *state* dari pesan dan tabel kedua menunjukkan matriks kunci. Setelah memperoleh matriks *state* dan matriks kunci, proses enkripsi menggunakan algoritma AES-128 dapat dimulai. Dalam AES-128, enkripsi dilakukan melalui 10 ronde enkripsi, di mana ronde pertama disebut *Initial Round* (menggunakan *AddRoundKey* dengan kunci awal), diikuti oleh 9 ronde utama, dan satu ronde akhir. Setiap ronde dalam AES-128 akan menggunakan kunci yang berbeda, yang dihasilkan dari ekspansi kunci awal melalui proses *Key Expansion*. *Key Expansion* ini memperluas kunci awal menjadi 10 kunci tambahan, sehingga menghasilkan total 11 kunci ronde (termasuk kunci awal).

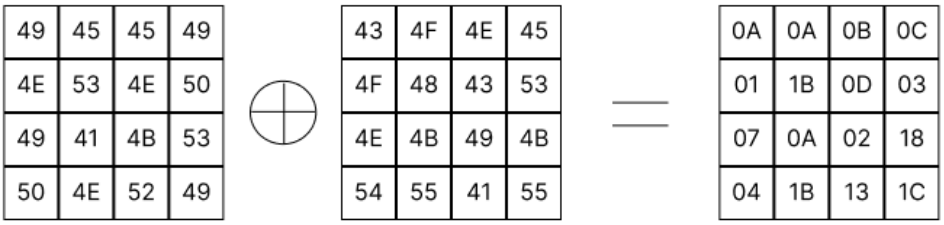
1. *Initial Round* (Ronde pertama)

   Enkripsi diawali dengan *AddRoundKey* pertama, dimana matriks *state* di-XOR dengan kunci awal. Tahap ini menyiapkan matriks untuk masuk ke ronde-ronde enkripsi.



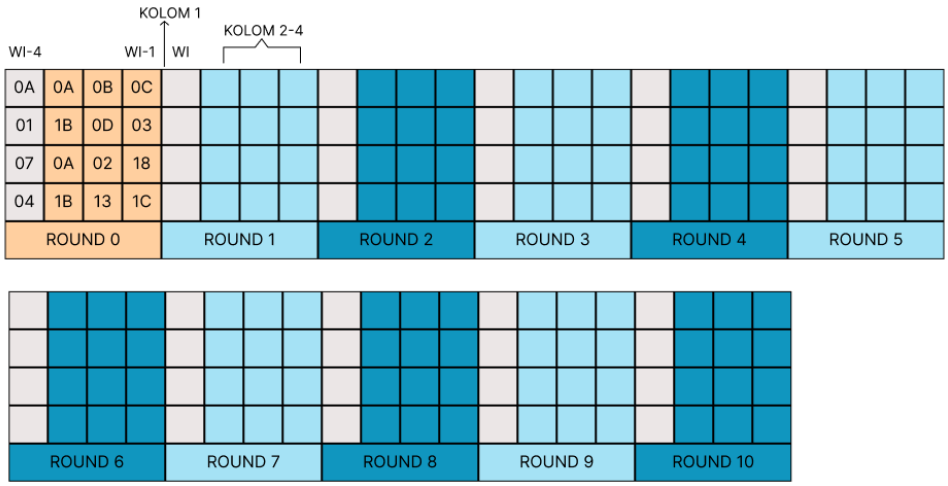
Gambar 11 Melakukan XOR dengan mengubah ke biner

Proses XOR dilakukan dengan mengubah bilangan heksadesimal menjadi biner. Proses ini dilakukan dengan XOR setiap kolom matriks *state* dan matriks kunci. Kolom pertama matriks *state* di XOR dengan kolom pertama matriks kunci. Hal yang sama dilakukan hingga kolom ke-empat. Hasil yang didapatkan dapat dilihat pada Gambar 12.



Gambar 12 XOR matriks *state* dan matriks kunci

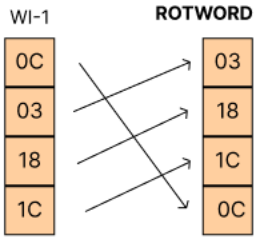
Proses selanjutnya adalah proses pembangkitan kunci/*generate key.* Pada proses ini akan dibangkitkan 10 buah kunci, yang mana akan digunakan pada setiap proses *AddRoundKey* pada 10 *round* dari *round function* AES-128.



Gambar 13 *Round* pembangkitan kunci

Pada Gambar 13, *Round 0* merupakan nilai dari *cipherkey* yang didapat dari hasil XOR antara matriks *state* dan matriks kunci. Selanjutnya, dilakukan pencarian untuk nilai dari *Round 1*. Untuk kolom yang akan dicari nilainya, dirumuskan dengan *WI*, dengan *W* berarti *"word"* dan *I* merupakan posisi atau indeks dalam urutan *key expansion* yang sedang diproses. Untuk

mendapatkan nilai pada kolom pertama di *Round 1*, langkah pertama yang dilakukan adalah menerapkan operasi *RotWord* pada kolom $WI - 1$.



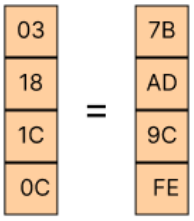
Gambar 14 Proses *Rotword*

Operasi *RotWord* pada Gambar 14 berfungsi untuk melakukan rotasi *byte* dalam satu kata dengan cara menggeser *byte* paling atas ke posisi paling bawah, sementara *byte* lainnya bergeser satu posisi ke atas. Hasil dari *RotWord* ini akan di operasikan menggunakan *SubBytes*. Dalam proses *SubBytes*, setiap *byte* yang dihasilkan dari *RotWord* digantikan dengan *byte* baru yang diambil dari tabel substitusi yang disebut S-Box (*Substitution Box)*.

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

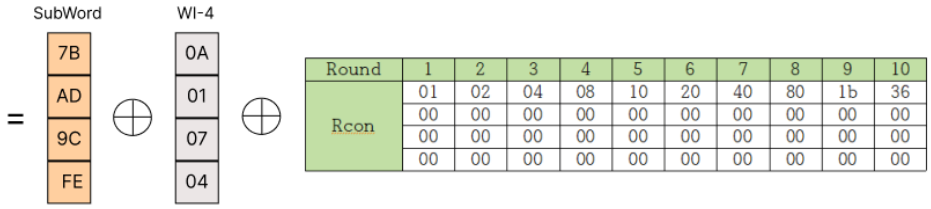
Gambar 15 Tabel S-Box
Sumber: (Selimis et al., 2007)

Untuk nilai pertama dari hasil *RotWord* yaitu 03, dimana 0 menandakan baris, dan 3 menandakan kolom. Sehingga untuk mendapatkan hasil dari *SubBytes* pada nilai pertama ini, yaitu melihat nilai yang terdapat pada baris 0 kolom 3 dalam tabel S-Box (7b).



Gambar 16 Hasil penukaran nilai dengan tabel S-Box

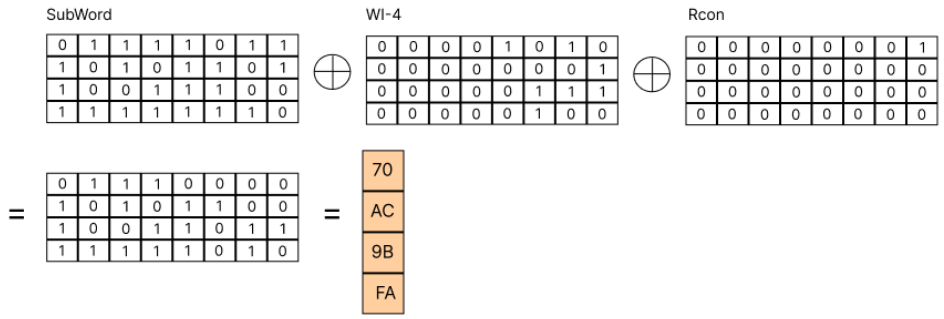
Setelah mendapatkan hasil dari *SubBytes* untuk nilai dari *RotWord*, langkah selanjutnya adalah melakukan operasi XOR antara hasil SubBytes tersebut dengan kolom $WI-4$ (Kolom $WI-4$ dapat dilihat pada Gambar 13). Hasil XOR ini kemudian dioperasikan kembali dengan konstanta Rcon.



Round	1	2	3	4	5	6	7	8	9	10
Rcon	01	02	04	08	10	20	40	80	1b	36
	00	00	00	00	00	00	00	00	00	00
	00	00	00	00	00	00	00	00	00	00
	00	00	00	00	00	00	00	00	00	00

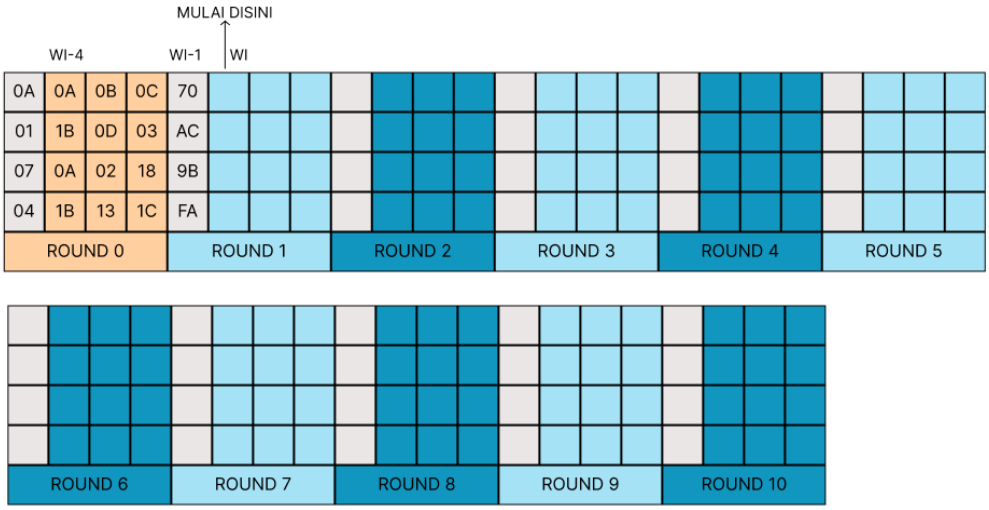
Gambar 17 Melakukan XOR hasil *SubBytes* dengan $WI-4$ dan Rcon

Operasi pada Gambar 17 dapat dilakukan dengan mengganti nilai pada kolom menjadi bilangan biner.



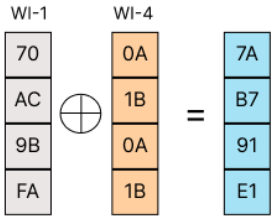
Gambar 18 Mengganti nilai ke bilangan biner dan melakukan XOR

Hasil yang didapat pada Gambar 18 merupakan nilai dari *WI* pada kolom pertama *Round 1*.



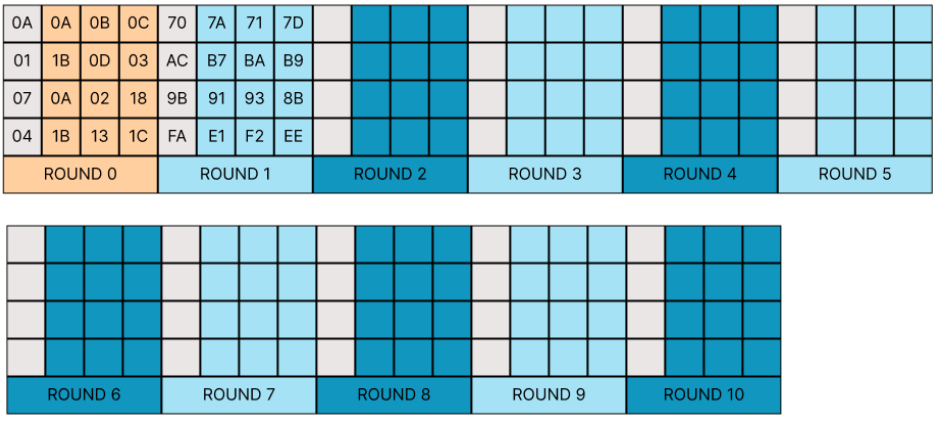
Gambar 19 Hasil *WI* kolom pertama dalam *Round 1*

Untuk mendapatkan nilai dari kolom kedua hingga keempat, cukup lakukan operasi XOR antara $WI-1$ dan $WI-4$ tanpa menggunakan Rcon.



Gambar 20 Mencari *WI* pada kolom dua

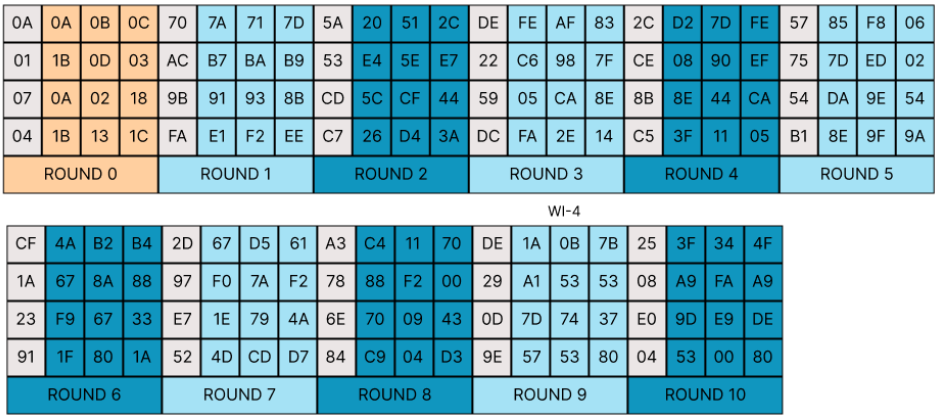
Rumus pada Gambar 20 digunakan secara berulang hingga semua nilai untuk kolom kedua, ketiga, dan keempat pada ronde pertama didapatkan, sebagaimana ditunjukkan pada Gambar 21



Gambar 21 Nilai *Round 1*

Pada ronde-ronde selanjutnya dalam proses *Key Expansion* AES, pola perhitungan nilai kolom mengikuti aturan yang sama seperti pada ronde pertama. Untuk kolom pertama di setiap ronde, nilai diperoleh dengan melakukan *RotWord, SubBytes,* lalu operasi XOR antara kolom sebelumnya ($WI-1$) dan kolom keempat dari ronde sebelumnya ($WI-4$), kemudian diikuti dengan operasi XOR dengan konstanta Rcon yang sesuai untuk ronde tersebut.

Sementara itu, untuk kolom kedua hingga keempat pada setiap ronde, nilai diperoleh hanya dengan melakukan XOR antara kolom sebelumnya ($WI-1$) dan kolom keempat dari ronde sebelumnya ($WI-4$), tanpa melibatkan Rcon. Proses ini diulang untuk setiap ronde, menghasilkan kunci ronde yang unik untuk setiap langkah enkripsi AES. Hasil dari keseluruhan ronde dapat dilihat pada Gambar 22.

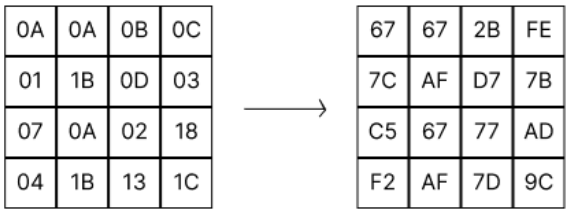


Gambar 22 Hasil *AddRoundKey* pada *InitialRound*

2. *Round Function*

   a. *SubBytes*

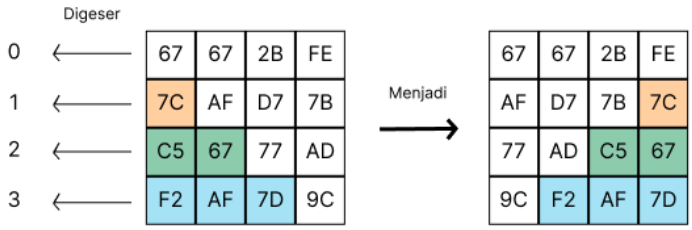
      Setelah operasi *AddRoundKey,* langkah selanjutnya adalah *SubBytes.* Pada langkah *SubBytes,* setiap *byte* dari *state* yang dihasilkan dari XOR antara *plaintext* dan *cipherkey* sebelumnya, digantikan dengan *byte* baru berdasarkan nilai pada tabel *S-Box*.



Gambar 23 Hasil *SubBytes*

b. *ShiftRows*

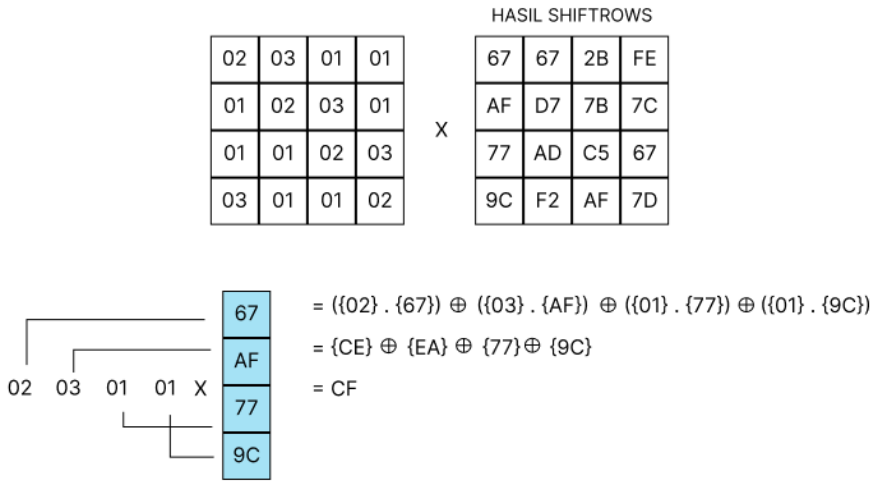
Dalam *ShiftRows*, blok dari hasil *SubBytes* akan digeser ke kiri dengan jumlah tertentu. Baris pertama tidak mengalami pergeseran, baris kedua digeser satu posisi ke kiri, baris ketiga digeser dua posisi ke kiri, dan baris keempat digeser tiga posisi ke kiri. Proses ini dapat dilihat pada Gambar 24.



Gambar 24 Proses *ShiftRows*

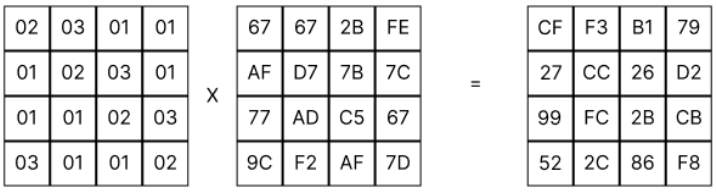
c. *MixColumns*

Setelah proses *ShiftRows*, langkah selanjutnya adalah *MixColumns.* Blok yang sudah didapat dari hasil *ShiftRows* akan dikalikan dengan matriks tetap seperti pada Gambar 25.



Gambar 25 Proses *MixColumns*

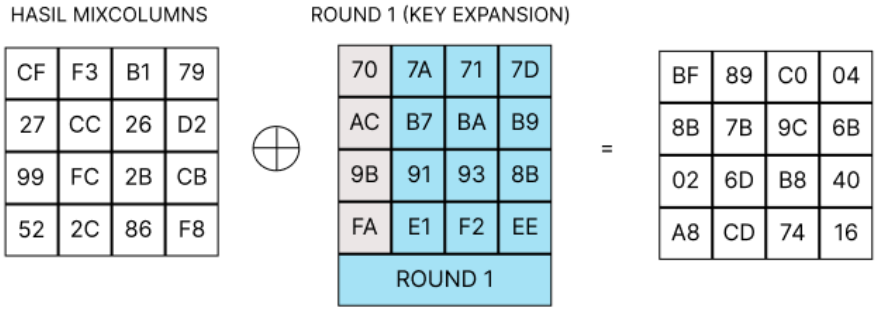
Proses *MixColumns* dilakukan dengan mengalikan kolom pertama dari matriks sebelah kiri pada Gambar 25 dengan baris pertama pada matriks hasil *ShiftRows.* Hasil yang didapat merupakan nilai dari kolom pertama baris pertama dalam matriks hasil. Selanjutnya dilakukan operasi baris kali kolom untuk mendapat nilai lainnya.



Gambar 26 Hasil *MixColumns*

d. *AddRoundKey*

Tahapan terakhir dalam *round function* merupakan *AddRoundKey*. Dalam *AddRoundKey* akan dilakukan XOR antara blok hasil *MixColumns* dengan *Round 1* yang telah dihasilkan selama proses *Key Expansion.*



Gambar 27 Proses *AddRoundKey*

Hasil dari proses ini akan digunakan dalam tahapan enkripsi ronde kedua (dimulai dari *SubBytes).*

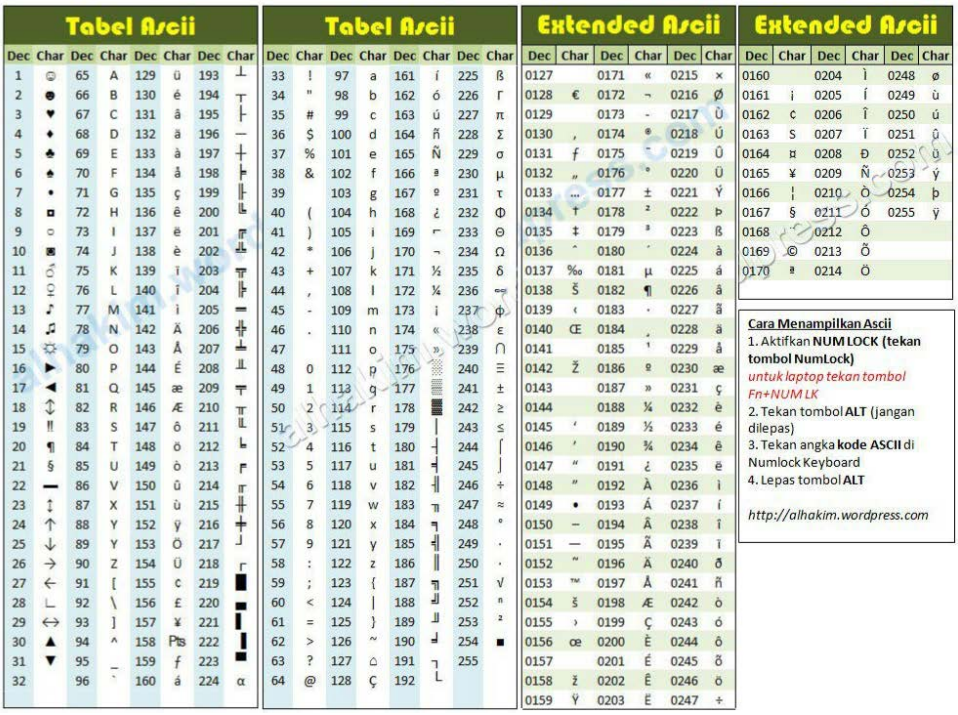
3. *Final Round*

Setelah melalui 9 ronde dalam *round function,* langkah terakhir adalah *final round,* yang mana pada ronde ini akan melewati proses *SubBytes, ShiftRows,* dan *AddRoundKey* tanpa proses *MixColumns*. Hasil akhir dari ronde ini merupakan matriks 4×4 yang berisi nilai heksadesimal yang diubah menjadi desimal dan *ciphertext.*

HEX	3F	26	C3	79	04	D1	0D	C3	09	BC	D9	55	CF	64	83	0C
DEC	63	38	195	121	4	209	13	195	9	188	217	85	207	100	131	12
CHA	?	&	├	y	♦	╤	♪	├	○	╝	┘	U	╧	d	â	♀

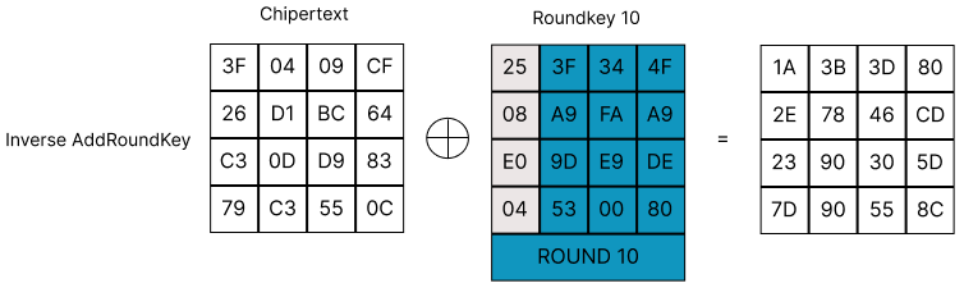
Gambar 28 Hasil akhir proses enkripsi AES-128

Berikut ASCII karakter yang dapat memudahkan untuk proses pengubahan nilai *hexadesimal* ke *decimal* ataupun ke *character.*



Gambar 29 ASCII *Characters*
Sumber: (Rismanto, 2017)

4. Proses Dekripsi



Gambar 30 Proses dekripsi *round 0*

Proses dekripsi merupakan kebalikan dari proses enkripsi. *Ciphertext* yang telah dihasilkan sebelumnya akan melalui tahap *inverse AddRoundKey*

terlebih dahulu. Pada tahap ini, *ciphertext* di-XOR dengan *round key* ke-10 untuk memperoleh blok 4×4 yang akan digunakan sebagai input pada ronde pertama dekripsi. Pada ronde pertama dekripsi, digunakan *round key* ke-9, dan seterusnya secara berurutan. Adapaun langkah-langkah dalam proses dekripsi meliputi *inverse ShiftRows*, *inverse SubBytes*, *inverse AddRoundKey*, dan *inverse MixColumns*. Pada *final round*, *inverse MixColumns* tidak disertakan, mirip seperti proses enkripsi.

3.3 *Rivest-Shamir-Adleman* (RSA)

RSA diambil dari nama penemu algoritma tersebut, yaitu Ron Rivest, Adi Shamir, dan Leonard Adleman dari *Massachusetts Institute of Technology* (MIT). RSA adalah algoritma enkripsi asimetris yang menggunakan dua kunci berbeda, yang disebut *key pair* (pasangan kunci), yang memiliki hubungan matematis dalam proses enkripsi dan dekripsi (John Swatrahadi et al., 2024).

Keamanan algoritma RSA didasarkan pada kesulitan memfaktorkan bilangan besar menjadi faktor-faktor primanya. Pemfaktoran ini diperlukan untuk mendapatkan kunci privat yang digunakan untuk mendekripsi pesan, yang membuat RSA dianggap aman. Namun, RSA juga memiliki kelemahan, dimana proses enkripsi dan dekripsinya cenderung memakan waktu lama, terutama saat menangani data berukuran besar. Oleh karena itu, RSA biasanya digunakan untuk pertukaran kunci dan pengamanan data yang relatif kecil. Properti dari algoritma RSA dapat dilihat pada Tabel 2 (W. C. U. Putri et al., 2023).

Tabel 4. Properti algoritma RSA

Properti	**Properti Algoritma**	**Kerahasiaan**
p	bilangan prima	rahasia
q	bilangan prima	rahasia
n	$n = p \times q$	tidak rahasia
$\emptyset(n)$	$\emptyset(n) = (p-1)(q-1)$	rahasia
e	kunci enkripsi, yang memenuhi $FPB(e, \emptyset(n)) = 1$	rahasia

Sumber: W. C. U. Putri et al., (2023)

RSA mempunyai cara kerja pada tahap pembangkitan kunci publik dan kunci privat, di antaranya:

1. Pilih dua bilangan prima p dan q secara acak, dengan $p \neq q$
2. Hitung

$$n = p \times q \tag{3}$$

3. Hitung

$$\emptyset(n) = (p-1)(q-1) \tag{4}$$

4. Pilih kunci publik e, yang relatif prima terhadap $\emptyset(n)$.
5. Bangkitkan d yang memenuhi

$$e.d = 1 \; mod \; \emptyset(n) \tag{5}$$

Untuk bilangan acak yang dipilih adalah p dan q, dimana kedua bilangan ini harus merupakan bilangan prima dan $p \neq q$, hal ini dilakukan sebab keamanan RSA didasarkan pada kesulitan memfaktorkan bilangan besar yang merupakan hasil perkalian dua bilangan prima yang dipilih secara acak. Selanjutnya, untuk pemilihan kunci publik e, dipilih secara acak namun harus relatif prima terhadap $\emptyset(n)$ untuk memastikan bahwa kunci privat d dapat dihitung menggunakan algoritma *Euclidean* yang diperpanjang (*extended euclidean algorithm*). Setelah melalui cara ini, maka kunci publik dan privat berhasil didapatkan. Kunci publik terdiri dari dua komponen: n (modulus yang digunakan) dan e (eksponen publik atau eksponen enkripsi). Sementara itu, kunci privat terdiri dari n (modulus yang digunakan) dan d (eksponen privat atau eksponen dekripsi) (G. G. Putri et al., 2018).

3.3.1 Enkripsi algoritma RSA

Dalam proses enkripsi data, penyandian dilakukan dengan menggunakan kunci publik (e, n). Mula-mula, data dalam bentuk *plaintext* dimasukkan dan diubah menjadi format ASCII untuk setiap karakternya. Setelah itu, *plaintext* akan dienkripsi menggunakan kunci publik melalui perhitungan dengan rumus berikut:

$$C = M^e \mod n \quad (6)$$

Setelah proses enkripsi selesai, data yang sebelumnya berupa *plaintext* akan berubah menjadi *ciphertext*. Hasil enkripsi yang berupa *ciphertext* tidak lagi dapat dibaca atau dimengerti secara langsung oleh pihak yang tidak memiliki kunci privat untuk mendekripsinya (Calvin Christian et al., 2023).

3.3.2 Dekripsi algoritma RSA

Dalam proses dekripsi, data yang sebelumnya telah dienkripsi (*ciphertext*) menggunakan kunci publik (e, n) akan dikembalikan ke bentuk pesan asli (*plaintext*) dengan menggunakan kunci privat (d, n). Langkah pertama yang dilakukan adalah memasukkan *ciphertext*, lalu data tersebut didekripsi menjadi angka-angka ASCII untuk setiap karakternya melalui rumus perhitungan berikut:

$$M = C^d \mod n \quad (7)$$

Setelah *ciphertext* berhasil diubah ke dalam bentuk bilangan ASCII, langkah berikutnya adalah mengonversi data ASCII tersebut kembali ke bentuk aslinya, yaitu *plaintext*. Pada tahap ini, setiap bilangan ASCII yang telah didekripsi akan diidentifikasi sebagai karakter-karakter yang sesuai dengan nilai ASCII-nya. Kemudian, seluruh karakter yang telah dihasilkan dari proses tersebut akan disusun ulang menjadi pesan atau teks awal, seperti bentuk semula sebelum proses enkripsi dilakukan (Calvin Christian et al., 2023).

3.3.3 Contoh Perhitungan Algoritma RSA

Berikut adalah contoh untuk pembangkitan kunci, proses enkripsi, dan dekripsi RSA:

1. Pembangkitan kunci
   a. Misalkan Zahra memilih $p = 53$ dan $q = 71$ (keduanya harus prima), maka dapat dihitung:

$$n = p \times q = 53 \times 71 = 3763 \quad (8)$$

$$ø(n) = (p-1)(q-1) \quad (9)$$

$$ø(n) = (53-1)(71-1)$$

$$ø(n) = 52 \times 70 = 3640$$

b. Zahra memilih kunci publik $e = 79$ (harus relatif prima dengan $ø(n)$, yaitu 3640).
c. Nilai e dan n bukan merupakan rahasia sehingga dapat dipublikasikan ke umum.
d. Zahra menghitung kunci privat d dengan memenuhi kongruensi pada persamaan (5), dimana d adalah invers modular dari e dalam modulus $ø(n)$. Nilai d dapat dihitung menggunakan algoritma *Euclidean* yang diperluas (*extended Euclidean algorithm*) atau dengan rumus berikut:

$$d = \frac{1 + k\, ø(n)}{e} \quad (10)$$

dengan k adalah bilangan bulat positif (dimulai dari 1) yang dicoba hingga memperoleh d bilangan bulat. Dalam contoh ini, dengan mencoba $k = 1,2,3, \ldots$, ditemukan untuk $k = 13$, diperoleh $d = 599$, yang merupakan bilangan bulat.

$$d = \frac{1 + 13 \times 3640}{79}$$

$$d = \frac{47321}{79} = 599$$

Nilai $d = 599$ ini menjadi kunci privat Zahra yang akan digunakan untuk proses dekripsi.

2. Proses Enkripsi RSA
   a. Dimisalkan Bela akan mengirim pesan $M =$ "$HELLO\ ZAHRA$" kepada Zahra. Setiap huruf ini dikodekan sebagai angka dengan aturan $A =$

$00, B = 01, \dots, Z = 25$, dengan spasi diabaikan. Dengan aturan ini, pesan M akan dikonversi menjadi angka integer, sebagai berikut:

$$M = 07041111142500071700$$

Nilai M yang diperoleh dipecah menjadi blok 4 digit, diantaranya:

$m_1 = 0704$

$m_2 = 1111$

$m_3 = 1425$

$m_4 = 0007$

$m_5 = 1700$

Setiap blok m_i berada dalam rentang $[0, 3763 - 1]$.

b. Bela mengenkripsi setiap blok menggunakan kunci publik Zahra ($e = 79$), dengan menggunakan persamaan (6) :

$$C_1 = 704^{79}\ mod\ 3763 = 1393$$

$$C_2 = 1111^{79}\ \text{mod}\ 3763 = 585$$

$$C_3 = 1425^{79}\ mod\ 3763 = 1690$$

$$C_4 = 7^{79}\ \text{mod}\ 3763 = 2816$$

$$C_5 = 1700^{79}\ mod\ 3763 = 3396$$

Sehingga diperoleh *ciphertext* $C = 13930585169028163396$

c. Bela mengirim *ciphertext* C kepada Zahra

3. Proses dekripsi RSA

a. Zahra menerima *ciphertext* dari Bela dan mendekripsi setiap blok dengan menggunakan kunci privatnya, yaitu $d = 599$. Dekripsi ini dilakukan dengan memasukkan nilai pada persamaan (7)

$$M_1 = 1393^{599}\ \text{mod}\ 3763 = 704$$

$$M_2 = 585^{599} \mod 3763 = 1111$$

$$M_3 = 1690^{599} \mod 3763 = 1425$$

$$M_4 = 2816^{599} \mod 3763 = 7$$

$$M_5 = 3396^{599} \mod 3763 = 1700$$

b. Setelah semua blok didekripsi, Zahra mendapat kembali pesan dalam bentuk integer $M = 07041111142500071700$, dan kemudian mengonversi angka ini menjadi huruf sehingga menghasilkan pesan $M = "HELLO\ ZAHRA"$.

3.4 *Shamir's Secret Sharing*

Secret Sharing (pembagian rahasia) adalah metode yang ditemukan oleh Adi Shamir dan George Blakley untuk memecahkan tugas kriptografi dengan cara membagi sebuah rahasia menjadi beberapa bagian di antara beberapa *shareholder* (pemegang bagian). Rahasia ini hanya bisa diungkapkan kembali jika sejumlah bagian tertentu digabungkan. Jika hanya satu bagian yang bocor atau jatuh ke tangan yang salah, itu tidak akan cukup untuk mengungkap rahasia (Georgi et al., 2021).

Shamir (1979) mendeskripsikan *Secret Sharing* dengan memecah data (D) dalam beberapa bagian (n) menjadi $D_i \ldots, D_n$. Untuk mendapatkan kembali nilai D_i, diperlukan kumpulan nilai D_i sebanyak k, dimana $k > 1$ dan $n \geq k$. Dengan kata lain, data akan lebih mudah kembali dipulihkan jika terdapat setidaknya k nilai D_i, sementara pihak yang tidak diinginkan akan kesulitan mendapatkan data jika hanya memiliki $k - 1$ nilai D_i atau kurang (Ivanov et al., 2020).

Algoritma ini menggunakan notasi (k, n) dalam memecah data, yang dikenal sebagai *threshold scheme* (skema *threshold*). Cara *split* (pemecahan data) dilakukan dengan memilih $k - 1$ bilangan bulat acak, yang masing-masing dilambangkan sebagai $a_1, a_2, \ldots, a_{k-1}$. Bilangan bulat ini dan data akan dinyatakan dalam bentuk fungsi polinomial $f(x)$ berderajat $k - 1$. Untuk mendapatkan nilai

share yang ideal, algoritma ini menggunakan aritmatika modular dengan modulus yang dilambangkan sebagai p. Nilai p adalah bilangan prima acak yang dipilih, dimana $p > D$. Fungsi ini dinotasikan dengan persamaan berikut:

$$f(x) = D + a_1 x + a_2 x^2 \ldots + a_{k-1} x^{x-1} \, mod \, p \tag{11}$$

Fungsi tersebut akan digunakan untuk menghasilkan nilai *share* dari kata yang ingin dipecah. Setiap nilai *share* didefinisikan dalam bentuk $(i, f(i))$, dimana $i = 1,2, \ldots, n$ dan $f(i)$ merupakan nilai masing-masing *share* yang dihasilkan dari fungsi $f(x)$ (Fachry et al., 2018).

Proses rekonstruksi (penggabungan) dari beberapa kumpulan *share* menggunakan metode *Lagrange Polynomial*, yang bekerja dengan mendefinisikan pasangan bentuk (x_i, y_i) yang diperoleh dari nilai $(i, f(i))$. Untuk mendapatkkan nilai *D* pada persamaan (8), nilai x dalam $f(x)$ harus bernilai 0, yang dapat dihitung menggunakan persamaan berikut (Fachry et al., 2018):

$$f(0) = D = \sum_{i=0}^{k} yi \prod_{1 \le j \le k, i \ne j} \frac{x_j}{x_j - x_i} \tag{12}$$

Berikut persyaratan setiap komponen untuk dapat membangun skema *Shamir's Secret Sharing:*

Tabel 5. Komponen *Shamir's Secret Sharing*

Komponen	**Tipe**	**Deskripsi**	**Persyaratan**
D	Rahasia	Data yang ingin disembunyikan atau dibagi.	Harus lebih kecil dari p ($D < p$), nilai rahasia.
p	Bilangan prima	Bilangan prima yang digunakan untuk operasi modulus dalam perhitungan polinomial.	Lebih besar dari D dan koefisien, harus bilangan prima.

Komponen	Tipe	Deskripsi	Persyaratan
n	Jumlah *shares*	Jumlah *share* yang diinginkan untuk membagi rahasia.	Harus lebih besar atau sama denga k ($n \geq k$)
k	Jumlah minimum rekonstruksi	Jumlah minimum *shares* yang diperlukan untuk merekonstruksi rahasia. Menentukan derajat polynomial yang akan digunakan.	Menentukan derajat polynomial, $k < n$.
a	Bilangan bulat acak	Koefisiean acak dalam polinomial, seperti $a_1, a_2, \dots, a_{k-1}$.	Bilangan bulat, dipilih secara acak, berbeda dalam tiap pembagian, berada dalam rentang 0 hingga $p - 1$
x	Indeks	Nilai yang digunakan sebagai input untuk menghitung *share*. Biasanya berupa angka urut, misalnya 1, 2, 3, dst.	Tidak acak namun berbeda untuk setiap *share*.
y	Nilai *share*	Hasil perhitungan polinomial pada nilai x tertentu, yaitu $y = f(x)\ mod\ p$.	Hasil dari polinomial mod p.

Sebagai contoh, data dengan nilai 98765 ($D = 98765$) akan diamankan. Data akan dipecah menjadi lima bagian ($n = 5$) dengan nilai *threshold* sama dengan 3 ($k = 3$). Proses selanjutnya yaitu memilih bilangan bulat acak sebanyak $k - 1$ dan bilangan prima. Karena $k = 3$, maka polinomial yang kita bentuk akan berderajat 2 ($k - 3 = 2$). Persamaan polinomialnya menjadi:

$$f(x) = D + a_1 x + a_2 x^2 \; mod \; p \tag{13}$$

Bilangan bulat yang akan digunakan adalah 5432 dan 3210 sedangkan bilangan prima yang dipilih adalah 104729 ($a_1 = 5432,\ a_2 = 3210,\ p = 104729$). Adapun langkah-langkah penyelesaiannya:

1. Menghitung nilai untuk setiap *share* menggunakan polinomial $f(x)$ untuk $x = 1,2,3,4,$ dan 5

   a. Untuk $x = 1$:

$$f(1) = 98765 + 5432 \times 1 + 3210 \times 1^2 \; mod \; 104729$$

$$= 98765 + 5432 + 3210 \; mod \; 104729$$

$$= 107407 \; mod \; 10729 = 2678$$

   b. Untuk $x = 2$:

$$f(2) = 98765 + 5432 \times 2 + 3210 \times 2^2 \; mod \; 104729$$

$$= 98765 + 10864 + 12840 \; mod \; 104729$$

$$= 122469 \; mod \; 10729 = 17740$$

   c. Untuk $x = 3$:

$$f(3) = 98765 + 5432 \times 3 + 3210 \times 3^2 \; mod \; 104729$$

$$= 98765 + 16296 + 28980 \; mod \; 104729$$

$$= 144041 \; mod \; 10729 = 39222$$

   d. Untuk $x = 4$:

$$f(4) = 98765 + 5432 \times 4 + 3210 \times 4^2 \; mod \; 104729$$

$$= 98765 + 21728 + 51456 \, mod \, 104729$$

$$= 171949 \, mod \, 10729 = 67124$$

e. Untuk $x = 5$:

$$f(5) = 98765 + 5432 \times 5 + 3210 \times 5^2 \, mod \, 104729$$

$$= 98765 + 27160 + 80250 \, mod \, 104729$$

$$= 206175 \, mod \, 10729 = 101446$$

Dari persamaan ini akan menghasilkan lima nilai yaitu $D_1 = 2678, D_2 = 17740, D_3 = 39222, D_4 = 67124, D_5 = 101446$. Berdasarkan nilai *threshold*, maka proses rekonstruksi hanya akan menggunakan tiga nilai dari hasil *share* yang telah terbentuk. Hasil *share* yang dipilih akan dinotasikan dalam (x_i, y_i). Nilai-nilai hasil *split* yang didapatkan adalah $(x_0, y_0) = (1,2678), (x_1, y_1) = (2,17740), (x_2, y_2) = (3,39222), (x_3, y_3) = (4,67124)$, dan $(x_4, y_4) = (5,101446)$.

Adapun untuk proses rekonstruksi, cara mendapatkan kembali data awal yaitu memasukkan nilai yang didapat ke dalam persamaan (12), sehingga menghasilkan:

$$D = \left(y_0 \times \frac{x_1 \times x_2}{(x_0 - x_1)(x_0 - x_2)} + y_1 \times \frac{x_0 \times x_2}{(x_1 - x_0)(x_1 - x_2)} + y_2 \times \frac{x_0 \times x_1}{(x_2 - x_0)(x_2 - x_1)} \right) mod \, p \quad (14)$$

Langkah-langkah perhitungannya adalah sebagai berikut:

1. Menghitung setiap bagian secara terpisah
   a. Bagian pertama:

$$y_0 \times \frac{x_1 \times x_2}{(x_0 - x_1)(x_0 - x_2)} \, mod \, p \quad (15)$$

$$= 2678 \times \frac{2 \times 3}{(1-2)(1-3)} \; mod \; 104729$$

$$= 2678 \times \frac{2 \times 3}{(-1)(-2)} \; mod \; 104729$$

$$= 2678 \times \frac{6}{2} \; mod \; 104729$$

$$= 2678 \times 3 \; mod \; 104729 = 8034$$

b. Bagian kedua:

$$y_1 \times \frac{x_0 \times x_2}{(x_1 - x_0)(x_1 - x_2)} \; mod \; p \qquad (16)$$

$$= 17740 \times \frac{1 \times 3}{(2-1)(2-3)} \; mod \; 104729$$

$$= 17740 \times \frac{3}{(1)(-1)} \; mod \; 104729$$

$$= 17740 \times \frac{3}{-1} \; mod \; 104729$$

$$= 17740 \times (-3) \; mod \; 104729$$

$$= -53220 \; mod \; 104729 = 51509$$

c. Bagian ketiga:

$$y_2 \times \frac{x_0 \times x_1}{(x_2 - x_0)(x_2 - x_1)} \; mod \; p \qquad (17)$$

$$= 39222 \times \frac{1 \times 2}{(3-1)(3-2)} \; mod \; 104729$$

$$= 39222 \times \frac{2}{2 \times 1} \; mod \; 104729$$

$$= 39222 \times 1 \; mod \; 104729$$

$$= 39222 \; mod \; 104729 = 39222$$

2. Gabungkan semua bagian sesuai rumus pada persamaan (14).

$$D = (8034 + 51509 + 39222) = 98765$$

Diperoleh $D = 98765$, yang mana nilai ini merupakan nilai awal dari data yang dirahasiakan. Sehingga, proses ini menunjukkan keberhasilan dalam merekonstruksi kembali nilai yang telah dibagi sebelumnya.

3.5 *Cloud Storage*

Cloud storage (penyimpanan berbasis awan) adalah layanan penyimpanan *file* berbasis jaringan internet, dimana *file* yang disimpan dapat diakses dan dikelola dari berbagai lokasi selama pengguna terhubung dengan *cloud* melalui internet. Cara kerja *cloud storage* adalah dengan memanfaatkan infrastruktur penyimpanan yang dikelola oleh penyedia layanan *cloud,* dan penggunaannya dapat berupa layanan penyimpanan dokumen atau arsip yang dapat diakses melalui internet. Sebelum *cloud computing* menjadi populer seperti sekarang, media penyimpanan berbasis *cloud* atau *cloud storage* ini lebih dikenal sebagai *virtual drive.* Dengan menggunakan *cloud storage,* pengguna tidak perlu membawa media penyimpanan fisik seperti *hard drive* (Tantowi & Wijayanti, 2023).

Media penyimpanan berbasis *cloud* tidak memerlukan pengaturan yang rumit, karena semua pengaturan telah dikelola secara otomatis oleh penyedia layanan melalui *platform* internet. Ketika pengguna ingin menyimpan *file* atau arsip, yang dibutuhkan hanyalah koneksi internet serta akun yang telah diverifikasi oleh penyedia layanan. Terdapat empat jenis penyimpanan *cloud storage,* di antaranya (Yang et al., 2020):

1. *Public cloud storage*, merupakan bentuk *cloud storage* dimana pengguna menyerahkan urusan penyimpanan data mereka kepada penyedia layanan *cloud* tanpa harus membangun infrastruktur atau memelihara server sendiri. Data yang disimpan hanya dapat diakses oleh pengguna yang memiliki izin. *Public cloud storage* menarik bagi banyak perusahaan kecil dan menengah karena fleksibilitas, kemampuan untuk berkembang, dan penghematan biaya yang ditawarkannya.
2. *Personal cloud storage*, dikenal juga sebagai *mobile cloud storage*, pada dasarnya merupakan bagian dari *cloud public*, namun lebih fokus pada penyimpanan *cloud* untuk individu.
3. *Private cloud storage*, merupakan bentuk *cloud* yang diciptakan untuk organisasi pengguna, dimana perusahaan harus membangun infrastruktur *cloud* sendiri dan memiliki tim profesional untuk mengelola dan menjaga server. Ini memberikan tingkat keamanan yang lebih tinggi dibandingkan dengan *public cloud*, serta kontrol data sepenuhnya ada di tangan perusahaan meskipun biaya yang harus dikeluarkan jadi lebih tinggi. Model penyimpanan ini lebih cocok untuk perusahaan besar yang memiliki data yang sangat penting dan sensitif.
4. *Hybrid cloud storage*, menggabungkan *public cloud* dan *private cloud*, sehingga mendapat manfaat dari kedua jenis tersebut. Pengguna dapat menyimpan data yang sensitif dan berharga di *private cloud*, sementara data lain dapat disimpan di *public cloud.*

3.6 Google Drive

Gambar 31 Logo google drive
Sumber: (logos-world, 2024)

Google drive adalah layanan penyimpanan data berbasis *cloud* yang diluncurkan pada 24 April 2012. Layanan ini memberikan pengguna ruang penyimpanan gratis sebesar 15 *Gigabyte* (GB). Dengan Google Drive, pengguna dapat dengan mudah menyimpan berbagai jenis data seperti gambar, video, dokumen teks, *spreadsheet*, dan presentasi, serta mengaksesnya dengan mudah dari berbagai perangkat. Google Drive juga terintegrasi dengan produk Google lainnya seperti Gmail, Google Plus, dan Google Search. Salah satu fitur utama Google Drive adalah kemampuannya untuk secara otomatis mengenali objek, baik itu orang maupun tempat, dalam berbagai jenis *file* yang berbeda (.txt, .html, .xml, dll.) selama proses pengindeksan. *Platform* ini juga menggunakan teknologi *Optical Character Recognition* (OCR) untuk menemukan teks dalam gambar atau pdf (Safitri & Nasution, 2023).

3.7 Dropbox

Gambar 32 Logo dropbox
Sumber: (Logosmarcas, 2022)

Dropbox adalah layanan berbasis *cloud* yang menyediakan penyimpanan data digital dan memungkinkan sinkronisasi data dengan perangkat lain yang juga menggunakan Dropbox. Dropbox pertama kali didirikan oleh Drew Houston dan Arash Ferdowsi, yang keduanya adalah lulusan *Massachusetts Institute of Technology* (MIT), dengan dana awal dari *Y Combinator*. Ide untuk membuat Dropbox muncul ketika Drew Houston, yang saat itu masih menjadi mahasiswa MIT, sering lupa membawa *USB Drive*. *USB Drive* dianggap tidak efisien karena sering mengalami masalah *error, corrupt*, dan kesalahan lainnya, sehingga Houston memutuskan untuk menciptakan solusi penyimpanan *cloud* yang lebih mudah. Dropbox secara resmi duluncurkan pada acara *TechCrunch*, sebuah konferensi teknologi, pada tahun 2008 (Tulim & Helman, 2022).

Dropbox banyak digunakan dalam layanan pertukaran *file* elektronik dan memiliki dokumentasi API *developer* yang cukup lengkap, yang bersifat *open source* dan dapat diimplementasikan oleh pengembang mana pun. Pengguna

Dropbox dapat mengunggah *file* secara otomatis sehingga memungkinkan pencadangan data pada layanan *cloud computing.* Dropbox menawarkan kapasitas penyimpanan sebesar 2GB secara gratis kepada pengguna saat pertama kali mendaftar. Pengguna dan pengembang memiliki opsi untuk menambah kapasitas penyimpanan pada akun mereka, baik secara gratis maupun berbayar. Selain itu, Dropbox menyediakan fitur berbagi data yang mudah digunakan oleh klien di internet, menyediakan opsi untuk menyimpan data secara *offline* dan sinkronisasi dengan *file online* (Bagus et al., 2022).

3.8 Microsoft OneDrive

Gambar 33 Logo onedrive
Sumber: (loghi-famosi, 2022)

OneDrive merupakan layanan penyimpanan *cloud* yang dimiliki Microsoft. Dengan berlangganan, pengguna mendapatkan ruang penyimpanan sebesar 1TB atau 5GB untuk paket gratis. Seperti layanan penyimpanan *cloud* lainnya, OneDrive menyediakan tempat untuk dapat menyimpan *file* di internet. Untuk mengaksesnya, pengguna perlu masuk ke OneDrive dengan akun Microsoft mereka (atau secara otomatis masuk ke Windows menggunakan akun Microsoft tersebut) untuk mengakses data (Gamnis et al., 2022). Aplikasi OneDrive dapat diakses melalui *file explorer* dan muncul secara *default* dalam sistem operasi tersebut. Namun, siapa pun dapat menggunakan OneDrive di web, dengan mengunduh aplikasi *dekstop* untuk Mac atau versi Windows sebelumnya, serta melalui aplikasi OneDrive di Android, iOS, Windows Phone dan X*box* (Agus et al., 2019).

Kekuatan utama OneDrive terletak pada kemampuannya untuk bekerja dengan aplikasi Microsoft Office seperti Word atau PowerPoint, sehingga ketika pengguna meluncurkan salah satu aplikasi tersebut, daftar dokumen terbaru yang disimpan di OneDrive akan muncul. Jika pengguna memiliki langganan Office 365 dan membuka dokumen yang disimpan di OneDrive, mereka dapat berkolaborasi

secara *real-time* dengan pengguna lain, bahkan bisa melihat perubahan yang dibuat oleh orang lain saat perubahan tersebut terjadi. OneDrive mengatur *file* berdasarkan jenisnya, sehingga memudahkan pengguna untuk menemukan apa yang mereka butuhkan. Aplikasi OneDrive di Android, iOS, dan *Windows Phone* juga memiliki fitur unggahan foto otomatis, yang secara otomatis menyimpan foto atau gambar yang diambil dengan telepon ke direktori yang telah diatur sebelumnya (Agus et al., 2019).

3.9 Application Programming Interface (API)

Application Programming Interface (API) adalah teknologi yang memungkinkan pertukaran informasi dan/atau data antara dua atau lebih perangkat lunak (*software*). API memungkinkan pengembang untuk mengintegrasikan dua bagian dari aplikasi yang berbeda secara bersamaan, dengan tujuan mempercepat proses pengembangan. API menyediakan berbagai fungsi secara terpisah sehingga pengembang tidak perlu membuat fitur serupa dari awal (Ariesta et al., 2021).

Manfaat penerapan API menjadi sangat jelas terutama ketika fitur yang diinginkan sudah sangat kompleks. API berfungsi sebagai antarmuka virtual antara dua fitur perangkat lunak yang bekerja secara bersamaan, seperti antara program pengolah kata dan *spreadsheet*. API dapat diterapkan pada level sistem operasi, sistem berkas, jaringan, dan sistem basis data. API yang beroperasi pada level sistem operasi bertujuan untuk membantu aplikasi berinteraksi dengan lapisan dasar sesuai dengan spesifikasi dan protokol yang ditetapkan (Bagus et al., 2022).

3.10 Django

Django adalah kerangka kerja web berbasis python yang dirancang untuk membantu pengembangan cepat dan menciptakan desain yang sederhana serta praktis. Django dikembangkan oleh para pengembang berpengalaman untuk menangani banyak masalah dalam pengembangan web, sehingga kita bisa lebih fokus pada pembuatan aplikasi tanpa harus memulai semuanya dari awal. Kerangka kerja ini bersifat *open source* dan gratis. Tujuan utama Django adalah untuk mempermudah pembuatan situs web kompleks yang menggunakan *database*. Django mendukung penggunaan kembali komponen dan kemudahan penambahan

fitur (*pluggability*), serta mendorong penggunaan kode yang lebih sedikit, keterkaitan antar bagian yang rendah, pengembangan yang cepat, dan prinsip "jangan mengulang pekerjaan yang sama". Python digunakan di semua bagian, termasuk dalam *file* konfigurasi dan model data (Srivastava, 2022).

3.11 *Structured Query Language Lite* (SQLite)

Gambar 34 Logo sqlite
Sumber: (sqlite.org, 2024)

SQLite pertama kali dirilis pada Agustus 2000 sebagai sebuah pustaka kecil yang menyediakan berbagai fungsi untuk manajemen data (Srivastava, 2022). Berbeda dengan sistem *database client-server* tradisional, yang biasanya berjalan pada proses terpisah dan berkomunikasi dengan aplikasi melalui mekanisme memori bersama, SQLite justru tertanam langsung di dalam proses aplikasi yang menggunakannya. Ini berarti bahwa alih-alih berkomunikasi dengan server *database* yang berada di luar proses aplikasi, aplikasi tersebut dapat langsung mengelola *database* SQLite dengan memanggil fungsi-fungsi yang disediakan oleh pustaka SQLite. Dengan demikian, SQLite menawarkan pendekatan yang lebih sederhana dan efisien untuk manajemen data dalam aplikasi (Gaffney et al., 2022).

SQLite tertanam di berbagai peramban web utama, komputer pribadi, televisi pintar, sistem media otomotif, serta dalam bahasa pemrograman PHP dan Python. Selain itu, SQLite juga ditemukan di setiap perangkat iOS dan Android, yang saat ini jumlahnya mencapai miliaran. Diperkirakan ada lebih dari satu triliun *database* SQLite yang aktif digunakan. SQLite diyakini sebagai salah satu pustaka perangkat lunak yang paling banyak digunakan dalam berbagai jenis aplikasi (Gaffney et al., 2022).

3.12 *Black Box Testing*

Black box testing adalah metode pengujian perangkat lunak yang bersifat fungsionalitas dan diterapkan pada sistem informasi yang sedang dikembangkan

(Turrahmi K et al., 2023). *Black box testing* berfokus pada pengujian fungsi-fungsi atau pengembangan modul, termasuk struktur data, akses data dalam *database*, dan potensi kesalahan performa yang terjadi. Metode ini dilakukan tanpa memerlukan pengetahuan tentang *source code* aplikasi. Dalam *black box testing*, salah satu teknik yang sering digunakan adalah menguji setiap menu dengan membagi masukan ke dalam kelompok berdasarkan fungsinya untuk memastikan setiap bagian berfungsi dengan benar sesuai dengan input yang diberikan (Herdianto, 2023).

3.13 Algoritma Hash-256

Algoritma Hash SHA-256 adalah salah satu anggota dari keluarga SHA-2 yang menghasilkan *hash* atau *digest* sepanjang 256 bit. Algoritma ini dikembangkan berdasarkan MD4 oleh Ronald L. Rivest dari MIT dan menggunakan enam operasi logika dasar, yaitu AND, OR, XOR, pergeseran bit ke kanan, dan rotasi bit ke kanan [11][12]. Dalam prosesnya, SHA-256 memproses pesan melalui *message schedule* yang terdiri dari 64 elemen 32-bit, delapan variabel 32-bit, serta delapan variabel penyimpan nilai *hash* 32-bit. Proses ini menghasilkan *message digest* dengan panjang 256 bit. Algoritma ini mengubah pesan masukan menjadi *hash* sepanjang 256 bit. Menurut *Secure Hash Standard* (SHS) yang diterbitkan oleh *National Institute of Standards and Technology* (NIST) [14], pesan yang panjangnya kurang dari 2^{64} bit dipecah menjadi blok-blok 512 bit sebelum diproses. Hasil akhir dari proses ini adalah *message digest* sepanjang 256 bit, yang memberikan tingkat keamanan tinggi dalam aplikasi kriptografi dan keamanan informasi (Ahmad Halimi et al., 2024).