EVALUASI WAKTU KOMPUTASI ALGORITMA *ADVANCED ENCRYPTION STANDARD* (AES) DENGAN *RIVEST-SHAMIR-ADLEMAN* (RSA) UNTUK KEAMANAN PESAN TEKS

SKRIPSI

OLEH
QOTRUNNADA SHOBAHA ZAHIRAH
NIM. 220601110017

PROGRAM STUDI MATEMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM MALANG
2026

EVALUASI WAKTU KOMPUTASI ALGORITMA *ADVANCED ENCRYPTION STANDARD* (AES) DENGAN *RIVEST-SHAMIR-ADLEMAN* (RSA) UNTUK KEAMANAN PESAN TEKS

SKRIPSI

Diajukan Kepada
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang
untuk Memenuhi Salah Satu Persyaratan dalam
Memperoleh Gelar Sarjana Matematika (S.Mat)

Oleh
Qotrunnada Shobaha Zahirah
NIM. 220601110017

PROGRAM STUDI MATEMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM MALANG
2026

EVALUASI WAKTU KOMPUTASI ALGORITMA *ADVANCED ENCRYPTION STANDARD* (AES) DENGAN *RIVEST-SHAMIR-ADLEMAN* (RSA) UNTUK KEAMANAN PESAN TEKS

SKRIPSI

Oleh
Qotrunnada Shobaha Zahirah
NIM. 220601110017

Telah Disetujui Untuk Diuji

Malang, 19 Februari 2026

Dosen Pembimbing I	Dosen Pembimbing II
Muhammad Khudzaifah, M. Si	Dr. Fachrur Rozi, M. Si
NIPPPK. 19900511 202321 1 029	NIP. 19800527 200801 1 012

Mengetahui,
Ketua Program Studi Matematika

Dr. Fachrur Rozi, M. Si
NIP. 19800527 200801 1 012

EVALUASI WAKTU KOMPUTASI ALGORITMA *ADVANCED ENCRYPTION STANDARD* (AES) DENGAN *RIVEST-SHAMIR-ADLEMAN* (RSA) UNTUK KEAMANAN PESAN TEKS

SKRIPSI

Oleh
Qotrunnada Shobaha Zahirah
NIM. 220601110017

Telah Dipertahankan di Depan Dewan Penguji Skripsi
dan Dinyatakan Diterima Sebagai Salah Satu Persyaratan
untuk Memperoleh Gelar Sarjana Matematika (S.Mat)

Tanggal, 7 April 2026

Ketua Penguji	: Hisyam Fahmi, M.Kom.	.....
Anggota Penguji 1	: Mohammad Nafie Jauhari, M.Si.	.....
Anggota Penguji 2	: Muhammad Khudzaifah, M.Si.	.....
Anggota Penguji 3	: Dr. Fachrur Rozi, M.Si	.....

Mengetahui,
Ketua Program Studi Matematika

Dr. Fachrur Rozi, M.Si.
NIP. 19800527 200801 1 012

PERNYATAAN KEASLIAN TULISAN

Saya bertanda tangan di bawah ini

Nama : Qotrunnada Shobaha Zahirah

NIM : 220601110017

Program Studi : Matematika

Fakultas : Sains dan Teknologi

Judul Skripsi : Evaluasi Waktu Komputasi Algoritma Advanced Encryption Standard (AES) dengan Rivest-Shamir-Adleman (RSA) untuk Keamanan Pesan Teks

Menyatakan bahwa skripsi yang saya tulis ini merupakan hasil karya sendiri, bukan mengambil alih tulisan atau pemikiran orang lain. Saya menyatakan skripsi ini sebagai pemikiran saya, kecuali dengan mencantumkan sumber kutipan pada daftar rujukan di halaman terakhir. Apabila dikemudian hari terbukti atau dapat dibuktikan skripsi ini hasil plagiasi atau tiruan orang lain, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Malang, 7 April 2026

Qotrunnada Shobaha Zahirah

Nim. 220601110017

MOTO

"Sesungguhnya bersama kesulitan ada kemudahan. Maka apabila engkau telah selesai (dari sesuatu urusan), tetaplah bekerja keras (untuk urusan yang lain), dan hanya kepada Tuhanmulah engkau Berharap"
(Q.S Al-Insyirah: 6-8)

"Perang telah usai, aku bisa pulang
Kubaringkan panah dan berteriak MENANGGG!!!"
(Nadin Amizah)

"Tidak ada mimpi yang terlalu tinggi dan tidak ada mimpi yang patut diremehkan. Lambungkan setinggi yang kau inginkan dan gapailah dengan selayaknya yang kau harapkan"
(Maudy Ayunda)

"Semua jatuh bangunmu hal yang biasa, angan dan pertanyaan waktu yang menjawabnya, berikan tenggat waktu bersedihlah secukupnya, rayakan perasaanmu sebagai manusia"
(Baskara Putra-Hindia)

HALAMAN PERSEMBAHAN

Segala puji dan ucapan rasa syukur panjatkan kepada Allah SWT, berkat rahmat serta hidayah-Nya sehingga penulis mampu menyelesaikan skripsi ini. Dengan segenap hati, rasa syukur dan rasa bahagia karena telah sampai pada titik ini, tentunya bukan suatu hal yang mudah, tetapi dengan niat, dukungan dan juga doa dari orang-orang baik disekitar saya, pada akhirnya tugas akhir ini terselesaikan dengan baik. Saya persembahkan Skripsi ini untuk:

1. Teruntuk cinta pertama dan pintu surgaku, Ayah Yusman, S.Pd dan Ibu Eni Mufarrida, S.E, yang dalam diamnya menyimpan doa paling tulus, yang dalam lelahnya tetap tersenyum demi masa depanku, dan yang dalam setiap langkahku selalu menjadi alasan untuk tidak menyerah. Jika hari ini ada satu pencapaian yang bisa kubanggakan, sesungguhnya itu adalah pantulan dari cinta, pengorbanan, dan keteguhan hati kalian yang tak pernah meminta balasan.
2. Teruntuk kedua adik laki-lakiku tersayang, yang menjadi penguat dalam setiap lelah dan pengingat bahwa aku harus terus melangkah lebih jauh. Semoga setiap capaian ini bukan hanya menjadi kebanggaanku, tetapi juga jejak kecil yang menginspirasi langkah kalian menuju masa depan yang lebih gemilang.
3. Sahabat terbaikku yakni Diah, Faizah, Afifah, dan Maulany yang telah membersamai sejak hari pertama perkuliahan hingga titik ini. Terima kasih telah menjadi tempat pulang, tawa, dan penguat di setiap lelah.
4. Khoridatul, Farin, Izza, dan Vebhi keluarga keduaku di perantauan. Terima kasih atas kebersamaan sederhana yang penuh arti dan dukungan yang tak pernah berhenti.
5. Teruntuk Zuhruful teman seperjuangan sejak PKL hingga bimbingan, terima kasih telah bertumbuh dan berproses bersama hingga setiap langkah menemukan akhirnya.

Semoga setiap doa, dukungan, dan kebaikan yang kalian berikan dibalas oleh Allah dengan limpahan keberkahan, kebahagiaan, dan segala hal terbaik dalam kehidupan.

KATA PENGANTAR

Assalamu'alaikum Warahmatullahi Wabarakatuh

Segala puji dan syukur atas kehadirat Allah SWT, yang telah melimpahkan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan skripsi ini dengan lancar. Shalawat serta salam semoga tetap tercurahkan kepada junjungan kita Nabi Muhammad SAW yang telah menuntun manusia dari zaman kegelapan ke zaman yang terang benderang yakni *Addinul Islam wal Iman*.

Skripsi ini disusun oleh penulis sebagai tugas akhir untuk mendapatkan gelar sarjana di Program Studi Matematika Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang. Sebagai penulis, perkenankan saya menyampaikan terima kasih dan penghargaaan yang setinggi-tingginya kepada:

1. Prof. Dr. Hj. Ilfi Nur Diana, M.Si., CHARM., CRMP, selaku Rektor Universitas Islam Negeri Maulana Malik Ibrahim Malang.
2. Dr. Agus Mulyono, M.Kes, selaku Dekan Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang.
3. Dr. Fachrur Rozi, M.Si, selaku Ketua Program Studi sekaligus Dosen Pembimbing II yang telah memberikan arahan dan masukan yang sangat bermanfaat dalam penulisan skripsi ini.
4. Muhammad Khudzaifah, M.Si, selaku Dosen Pembimbing I yang senantiasa memberikan arahan serta ilmu yang sangat berharga dalam penyusunan skripsi ini.
5. Ari Kusumastuti, M.Pd, M.Si, selaku Dosen Wali yang telah memberikan arahan dan dukungan selama masa studi.
6. Seluruh dosen Program Studi Matematika, Fakultas Sains dan Teknologi, yang telah memberikan ilmu serta wawasan selama perkuliahan.
7. Teristimewa penulis ucapkan kepada cinta pertama yakni Ayahanda Yusman, S.Pd dan pintu surgaku Ibunda Eni Mufarrida, S.E, terima kasih selalu mengusahakan segala kebutuhan penulis, mendidik, membimbing, dan selalu memberikan kasih sayang yang tulus, motivasi, serta dukungan dan mendoakan penulis dalam keadaan apapun agar penulis mampu bertahan untuk melangkah dalam meraih mimpi di masa depan.

8. Teruntuk kedua adik laki-laki saya, M. Alif Azmi R dan M. Haikal R yang selalu membuat penulis termotivasi untuk bisa belajar menjadi sosok kakak yang dapat memberikan pengaruh positif, baik dalam bidang akademik maupun non-akademik.
9. Teruntuk sahabat seperjuangan yakni Diah, Maulany, Faizah, dan Afifah yang selalu memberikan dukungan dan motivasi selama perkuliahan. Meskipun setelah ini akan menjalani kehidupan masing-masing yang berbeda dan mungkin berada di kota atau negara yang berbeda, semoga pertemanan ini selalu terjaga selamanya.
10. Seluruh teman-teman mahasiswa Program Studi Matematika angkatan 2022.
11. Semua pihak yang terlibat secara langsung maupun tidak langsung yang tidak bisa penulis tuliskan satu per satu.
12. Dan yang terakhir untuk diriku sendiri, Qotrunnada Shobaha Zahirah. Terima kasih sudah bertahan dan melangkah sejauh ini. Di tengah tekanan, keraguan, dan berbagai ujian yang hadir termasuk luka yang sempat menguji keteguhan hati. Aku memilih untuk tetap berdiri dan menyelesaikan apa yang telah dimulai. Pencapaian ini bukan hanya tentang menyelesaikan sebuah karya, tetapi tentang keberanian untuk terus berjalan meski keadaan tidak selalu mudah.

Penulis menyadari dalam menyusun skripsi ini masih terdapat kekurangan yang perlu diperbaiki. Meskipun demikian, harapannya adalah agar skripsi ini dapat memberikan manfaat terutama bagi penulis dan pembaca.

Wassalamu'alaikum Warahmatullahi Wabarakatuh.

Malang, 7 April 2026

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR SIMBOL

$a \equiv b \ (mod\ m)$	: Kongruensi modulo
C	: *Ciphertext*
$C \equiv M^e \ (mod\ n)$	: Persamaan enkripsi RSA
C_{key}	: Kunci AES yang dienkripsi RSA
C_{text}	: *Ciphertext* hasil enkripsi AES
C_{total}	: Gabungan *ciphertext* dan kunci terenkripsi
D	: Fungsi dekripsi
$D_K(C)$	: Dekripsi *ciphertext* dengan kunci K
$D_{d,n}$	: Dekripsi RSA dengan kunci privat (d, n)
E	: Fungsi enkripsi
$E_K(P)$	: Enkripsi *plaintext* dengan kunci K
$E_{e,n}$	: Enkripsi RSA dengan kunci publik (e, n)
gcd (a, b)	: *Greatest Common Divisor* (FPB)
$GF(2^8)$	: Medan hingga pada operasi *MixColumns*
K	: Kunci
K_{AES}	: Kunci simetris AES
M	: Representasi numerik *plaintext* pada RSA
$M \equiv C^d \ (mod\ n)$	: Persamaan dekripsi RSA
mod	: Operasi modulo
m	: Modulus
n	: Modulus RSA $(n = p \times q)$
p	: Bilangan prima pertama pada RSA
q	: Bilangan prima kedua pada RSA
S	: *State* matrix AES
S'	: *State* hasil transformasi
$S_{i,j}$	: Elemen *byte* baris ke-i kolom ke-j
d	: Eksponen privat RSA
e	: Eksponen publik RSA
p	: *Plaintext*

$\varphi(n)$	: Fungsi Totien Euler
$\oplus$	: Operasi XOR

DAFTAR LAMPIRAN

ABSTRAK

Zahirah, Qotrunnada Shobaha. 2026. Evaluasi Waktu Komputasi Algoritma *Advanced Encryption Standard* (AES) dengan *Rivest-Shamir-Adleman* (RSA) untuk Keamanan Pesan Teks. Skripsi. Program Studi Matematika, Fakultas Sains dan Teknologi, Universitas Islam Negeri Maulana Malik Ibrahim Malang. Pembimbing: (1) Muhammad Khudzaifah, M.Si. (2) Dr. Fachrur Rozi, M.Si.

Kata Kunci: AES-128, RSA, *Hybrid* AES–RSA, waktu komputasi, pesan teks.

Perkembangan teknologi informasi yang pesat menyebabkan meningkatnya kebutuhan terhadap keamanan data, khususnya dalam pengolahan dan pengiriman pesan teks. Berbagai algoritma kriptografi, seperti *Advanced Encryption Standard* (AES) dan *Rivest-Shamir-Adleman* (RSA), telah banyak digunakan untuk menjaga kerahasiaan informasi. Namun, masing-masing algoritma memiliki karakteristik kinerja yang berbeda, terutama dari aspek waktu komputasi. Oleh karena itu, diperlukan evaluasi untuk mengetahui efisiensi waktu proses dari masing-masing algoritma serta kombinasi keduanya dalam metode *hybrid*. Penelitian ini bertujuan untuk mengevaluasi dan membandingkan waktu komputasi proses enkripsi dan dekripsi pada algoritma AES, RSA, serta metode *hybrid* AES-RSA dalam pengolahan pesan teks. Implementasi dilakukan melalui simulasi sistem dengan menggunakan AES-128 untuk enkripsi data dan RSA untuk enkripsi kunci simetris. Pengukuran waktu komputasi dilakukan sebanyak 50 kali pengulangan untuk setiap metode dengan variasi panjang pesan. Data hasil pengujian dianalisis menggunakan uji normalitas dan uji *Kruskal-Wallis* untuk mengidentifikasi perbedaan performa waktu secara statistik. Hasil penelitian menunjukkan bahwa AES memiliki waktu komputasi paling efisien, RSA membutuhkan waktu komputasi paling besar, sedangkan metode *hybrid* AES-RSA berada di antara keduanya. Temuan ini memberikan dasar empiris dalam pemilihan metode kriptografi berdasarkan aspek kinerja waktu proses pada sistem pengolahan pesan teks.

ABSTRACT

Zahirah, Qotrunnada Shobaha. 2026. Evaluation of the Computational Time of the Advanced Encryption Standard (AES) and Rivest-Shamir-Adleman (RSA) Algorithms for Text Message Security. Thesis. Mathematics Program, Faculty of Science and Technology, Universitas Islam Negeri Maulana Malik Ibrahim Malang. Advisors: (1) Muhammad Khudzaifah, M.Si. (2) Dr. Fachrur Rozi, M.Si.

Keywords: AES-128, RSA, Hybrid AES–RSA, computational time, text messages.

Rapid advancements in information technology have led to an increased need for data security, particularly in the processing and transmission of text messages. Various cryptographic algorithms, such as the Advanced Encryption Standard (AES) and Rivest-Shamir-Adleman (RSA), have been widely used to maintain information confidentiality. However, each algorithm has distinct performance characteristics, particularly in terms of computational time. Therefore, an evaluation is necessary to determine the computational efficiency of each algorithm as well as their combination in a hybrid method. This study aims to evaluate and compare the computational time of the encryption and decryption processes of the AES, RSA, and the AES-RSA hybrid method in text message processing. The implementation was carried out through system simulation using AES-128 for data encryption and RSA for symmetric key encryption. Computational time measurements were conducted 50 times for each method with varying message lengths. The test data were analyzed using normality tests and the Kruskal-Wallis test to identify statistically significant differences in computational performance. The results show that AES has the most efficient computational time, RSA requires the longest computational time, while the AES-RSA hybrid method lies between the two. These findings provide an empirical basis for selecting cryptographic methods based on computational performance in text message processing systems.

مسخلص البحث

زاهر, قطر الندى, صبحا. ٢٠٢٦. **تقييم الوقت الحاسوبي *لخوارزميات معيار التشفير المتقدم (AES)* مع *ريفيست–شامير–أدلمان (RSA)*, لأمان الرسائل النصية**. الجامعى. قسم الرياضيات، كلية العلوم والتكنولوجيا، جامعة مولانامالك إبراهيم الإسلاميةالحكوميةمالانج. المشرف: (1) محمدخديفة,الما جستير فيالعلوم. (2) الدكتور فخر الرازى،الماجستير فيالعلوم.

الكلمات الأساسية: *AES*، *RSA*، *AES-128* هجين مع *RSA*، وقت الحساب، رسالة نصية.

أدى التطور السريع لتقنية المعلومات إلى زيادة الحاجة إلى أمن البيانات، خاصة في معالجة وإرسال الرسائل النصية. تم استخدام خوارزميات تشفير متنوعة، مثل *معيار التشفير المتقدم (AES)* ومعيار *ريفيست–شامير–أدلمان (RSA)*، على نطاق واسع للحفاظ على سرية المعلومات. ومع ذلك، لكل خوارزمية خصائص أداء مختلفة، خاصة من ناحية الوقت الحسابي. لذلك، هناك حاجة إلى تقييم لتحديد كفاءة زمن معالجة كل خوارزمية بالإضافة إلى الجمع بين الاثنين في الطريقة *الهجينة*. هدفت هذه الدراسة إلى تقييم ومقارنة الوقت الحسابي لعملية التشفير وفك التشفير في *AES*، *وخوارزميات RSA*، بالإضافة إلى طريقة *RSA-AES* الهجينة في معالجة الرسائل النصية. يتم تنفيذ التنفيذ من خلال محاكاة النظام باستخدام *AES-128* لتشفير البيانات و *RSA* لتشفير المفتاح المتماثل. تجرى قياسات الوقت الحسابي 50 مرة لكل طريقة مع اختلافات في طول الرسالة. تم تحليل بيانات نتائج الاختبار باستخدام اختبارات الطبيعية *واختبارات كروسكال–واليس* لتحديد الفروق في الأداء الزمني إحصائيا. ظهرت النتائج أن *AES* لديه الوقت الحسابي الأكثر كفاءة، بينما يتطلب *RSA* أكبر وقت حسابي، بينما طريقة *RSA-AES* الهجينة تقع في مكان ما بينهما. توفر هذه النتائج أساسا تجريبيا لاختيار طرق التشفير بناء على جوانب الأداء في وقت معالجة نظام معالجة الرسائل النصية.

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Di era digital yang serba terhubung seperti saat ini, komunikasi melalui pesan teks telah menjadi sarana utama dalam pertukaran informasi, baik antar individu maupun antar instansi (Akter dkk., 2023). Komunikasi digital modern kini banyak dilakukan melalui pesan teks, baik dalam aplikasi perpesanan, sistem notifikasi, maupun komunikasi antarinstansi. Pesan teks sering kali mengandung data sensitif seperti informasi pribadi, hasil transaksi, dan dokumen penting, sehingga keamanan pesan menjadi hal yang krusial dalam menjaga kerahasiaan data pengguna (Chang dkk., 2025). Menurut Stallings (2017), sistem komunikasi yang berjalan di atas jaringan publik menghadapi risiko besar, seperti penyadapan (*eavesdropping*), pengubahan isi pesan (*modification*), dan penyamaran identitas (*masquerade*). Risiko tersebut dapat menyebabkan gangguan terhadap privasi pengguna serta menurunkan keandalan sistem komunikasi (Walle, 2024). Oleh karena itu, dibutuhkan suatu mekanisme pengamanan yang dapat memastikan bahwa pesan teks hanya dapat diakses oleh pihak yang berwenang serta tetap utuh selama proses transmisi berlangsung (Kominfo, 2022).

Dalam perspektif Islam, urgensi menjaga keamanan dan privasi pesan digital sejalan dengan nilai-nilai yang diajarkan dalam Al-Qur'an. Allah Swt. berfirman dalam Surah Al-Mu'minun ayat 8:

وَالَّذِينَ هُمْ لِأَمَانَاتِهِمْ وَعَهْدِهِمْ رَاعُونَ

"Dan (sungguh beruntung) orang yang memelihara amanah-amanah dan janjinya." (NU Online, 2025).

Ayat ini menegaskan bahwa amanah mencakup seluruh bentuk tanggung jawab yang dipercayakan kepada individu, tidak terbatas pada aspek pernikahan, tetapi meliputi janji dan perjanjian dalam berbagai konteks sosial. Menjaga amanah merupakan karakter utama orang beriman dan menjadi indikator integritas moral dalam interaksi sosial.

Dalam konteks kemajuan teknologi informasi dan komunikasi, konsep amanah meluas mencakup tanggung jawab atas data pribadi, pesan digital, dan informasi elektronik lainnya. Menjaga kerahasiaan dan mencegah akses tidak sah terhadap informasi tersebut merupakan bentuk amanah digital. Kelalaian atau penyalahgunaan dalam menjaga informasi digital bukan hanya berdampak hukum dan sosial, tetapi juga bertentangan dengan prinsip etika Islam. Dengan demikian, ayat ini menjadi landasan etis dan spiritual bagi pengembangan sistem keamanan digital yang menjunjung tinggi perlindungan data dan privasi sebagai wujud dari nilai keimanan dan tanggung jawab dalam Islam.

Salah satu pendekatan yang banyak digunakan dalam menjaga keamanan pesan digital adalah kriptografi, yaitu ilmu dan seni menyandikan pesan (*plaintext*) ke dalam bentuk tidak bermakna (*ciphertext*) agar tidak dapat dibaca oleh pihak yang tidak berwenang (Munir, 2006). Kriptografi berperan penting dalam menyediakan layanan keamanan seperti kerahasiaan *(confidentiality)*, integritas *(integrity)*, dan keaslian *(authenticity)* data, terutama dalam komunikasi melalui jaringan terbuka. Setiap algoritma kriptografi memiliki keunggulan dan keterbatasan masing-masing. Algoritma simetris seperti *Advanced Encryption Standard* (AES) sangat efisien untuk proses enkripsi-dekripsi karena menggunakan satu kunci yang sama, namun memiliki kelemahan dalam distribusi kunci secara

aman (Stallings, 2017). Sebaliknya, algoritma asimetris seperti *Rivest–Shamir–Adleman* (RSA) menggunakan pasangan kunci publik dan privat sehingga memudahkan manajemen kunci, tetapi kurang efisien untuk mengamankan data berukuran besar (Munir, 2006).

Untuk mengatasi keterbatasan dari masing-masing algoritma kriptografi, dikembangkan metode kriptografi *Hybrid*, yaitu kombinasi antara algoritma simetris dan asimetris dalam satu sistem yang saling melengkapi (Stallings, 2017). Dalam model enkripsi *Hybrid*, data utama (*plaintext*) terlebih dahulu dienkripsi menggunakan algoritma *Advanced Encryption Standard* (AES) karena AES sangat cepat dalam memproses data besar menggunakan satu kunci simetris. Setelah itu, kunci AES yang dihasilkan dienkripsi dengan algoritma *Rivest–Shamir–Adleman* (RSA). Urutan ini dipilih karena RSA memiliki proses komputasi lebih berat dan hanya efisien untuk data berukuran kecil, seperti kunci AES. Dengan demikian, sistem memperoleh kombinasi antara efisiensi dari AES dan keamanan distribusi kunci dari RSA. Pendekatan ini memungkinkan keamanan distribusi kunci tetap terjaga dan mencegah akses tidak sah selama proses transmisi data, sekaligus mempertahankan efisiensi proses enkripsi dan dekripsi pesan (Akter dkk., 2023; Chang dkk., 2025).

Beberapa penelitian terdahulu memberikan kontribusi penting terhadap pengembangan model *Hybrid* AES–RSA, meskipun dengan konteks dan batasan yang berbeda. Penelitian oleh Akter dkk. (2023) berperan dalam memperkenalkan model *Hybrid cryptography* AES–RSA untuk meningkatkan keamanan data digital. Penelitian tersebut menunjukkan bahwa kombinasi kedua algoritma mampu memperkuat perlindungan terhadap serangan seperti *key exposure* dan *man-in-the-*

middle attack. Namun, penelitian ini masih terbatas pada konteks *cloud computing*, belum secara spesifik diterapkan pada sistem pesan teks yang membutuhkan waktu pemrosesan lebih cepat dan efisiensi tinggi.

Penelitian oleh Walle (2024) menyoroti efektivitas integrasi AES dan RSA dalam meningkatkan keamanan proses transmisi data. Hasilnya menunjukkan peningkatan signifikan terhadap ketahanan enkripsi dari serangan penyadapan dan modifikasi pesan. Akan tetapi, penelitian tersebut lebih menekankan pada aspek keamanan jaringan dan tidak mengukur performa waktu enkripsi–dekripsi serta efisiensi sistem pada pesan teks yang bersifat *real-time*.

Penelitian oleh Chang dkk. (2025) menunjukkan bahwa mekanisme *Hybrid* AES–RSA yang dimodifikasi (MRA *mode*) dapat meningkatkan efisiensi enkripsi–dekripsi tanpa menurunkan tingkt keamanan. Fokus penelitian diarahkan pada komunikasi IoT berdaya rendah, bukan pesan teks, sehingga penerapan pada komunikasi berbasis teks masih memerlukan penyesuaian lebih lanjut.

Dari ketiga penelitian tersebut dapat disimpulkan bahwa pendekatan *Hybrid cryptography* AES–RSA telah terbukti efektif dalam meningkatkan keamanan data digital. Namun, masih terdapat kesenjangan penelitian (*research gap*) dalam penerapannya pada sistem komunikasi pesan teks yang membutuhkan efisiensi waktu tinggi serta kecepatan proses enkripsi dan dekripsi secara *real-time*. Penelitian sebelumnya juga belum banyak menganalisis performa algoritma *Hybrid* AES–RSA secara empiris terhadap waktu pemrosesan pesan teks yang dikirim melalui jaringan terbuka.

Selain alasan teknis, urgensi penelitian ini juga diperkuat oleh aspek regulasi. Pemerintah Indonesia melalui Undang-Undang Nomor 27 Tahun 2022

tentang Perlindungan Data Pribadi (UU PDP) menegaskan pentingnya perlindungan data pribadi dalam semua bentuk komunikasi digital, termasuk pesan teks (Republik Indonesia, 2022). Hal ini menunjukkan bahwa penerapan metode kriptografi tidak hanya merupakan kebutuhan teknologi, tetapi juga kewajiban hukum dalam menjaga keamanan informasi pengguna.

Pesan teks dipilih sebagai fokus penelitian karena mewakili bentuk komunikasi digital yang sederhana namun paling sering digunakan dalam aplikasi pesan singkat, autentikasi pengguna, dan sistem notifikasi. Oleh karena itu, penelitian ini bertujuan untuk menguji penerapan algoritma kriptografi *Hybrid* AES-RSA, serta menganalisis waktu proses enkripsi dan dekripsi pesan teks secara empiris.

1.2 Rumusan Masalah

Berdasarkan latar belakang tersebut, rumusan masalah dalam penelitian ini sebagai berikut:

1. Bagaimanakah hasil simulasi kriptografi *Hybrid* AES-RSA dalam sistem keamanan pesan teks?
2. Bagaimana evaluasi waktu yang dibutuhkan oleh sistem dalam proses enkripsi dan dekripsi pesan teks menggunakan pendekatan kriptografi *Hybrid* AES-RSA?

1.3 Tujuan Penelitian

Berdasarkan rumusan masalah, maka tujuan dari penelitian ini sebagai berikut:

1. Untuk mensimulasikan algoritma kriptografi *Hybrid* AES-RSA dalam sistem keamanan pesan teks, sehingga diperoleh *ciphertext*, kunci AES terenkripsi RSA, dan *output* dari proses enkripsi-dekripsi.
2. Untuk mengukur dan mengevaluasi waktu proses enkripsi dan dekripsi pada algoritma *Hybrid* AES–RSA menggunakan data pesan teks dengan variasi panjang karakter.

1.4 Manfaat Penelitian

Adapun manfaat yang diharapkan dari penelitian ini adalah sebagai berikut:

1. Manfaat Teoritis

   Menambah literatur di bidang kriptografi terkait penerapan algoritma *Hybrid* AES–RSA dalam pengamanan pesan teks, serta memperkuat kajian empiris mengenai kecepatan proses enkripsi–dekripsi pada komunikasi digital.

2. Manfaat Praktis
   a. Memberikan solusi praktis dalam pengamanan pesan teks melalui penerapan sistem kriptografi *Hybrid* yang cepat dan aman.
   b. Menyediakan data dan informasi yang berguna bagi pengembang sistem komunikasi digital dalam memilih dan menerapkan algoritma kriptografi yang sesuai.
   c. Mendukung implementasi kebijakan perlindungan data pribadi sesuai dengan peraturan perundang-undangan yang berlaku di Indonesia.

1.5 Batasan Masalah

Berikut batasan masalah dalam penelitian ini adalah sebagai berikut:

1. Implementasi algoritma dilakukan pada lingkungan simulasi perangkat lunak.
2. Fokus pengujian diarahkan pada analisis waktu komputasi yang mencakup proses enkripsi dan dekripsi sebagai dua parameter utama yang diukur dalam penelitian ini.

1.6 Definisi Istilah

Berikut istilah yang digunakan dalam penelitian:

1. Kriptografi: Ilmu yang mempelajari teknik penyandian data agar informasi tidak dapat diakses oleh pihak yang tidak berwenang.
2. *Plaintext*: Pesan asli sebelum dienkripsi, dapat dibaca oleh manusia atau sistem.
3. *Ciphertext*: Hasil dari proses enkripsi, berupa data acak yang tidak dapat dibaca tanpa proses dekripsi.
4. Kunci (*key*): Nilai rahasia yang digunakan untuk proses enkripsi dan dekripsi pesan.
5. Enkripsi: Proses mengubah *plaintext* menjadi *ciphertext* menggunakan algoritma dan kunci tertentu.
6. Dekripsi: Proses mengubah *ciphertext* kembali menjadi *plaintext* menggunakan kunci yang sesuai.
7. *Advanced Encryption Standard* (AES): Algoritma kriptografi simetris berbasis blok yang mengenkripsi data menggunakan satu kunci rahasia.
8. *Rivest-Shamir-Adleman* (RSA): Algoritma kriptografi asimetris yang menggunakan sepasang kunci publik dan privat untuk enkripsi dan dekripsi.

9. *Hybrid cryptography*: Kombinasi antara algoritma simetris dan asimetris di mana AES digunakan untuk mengenkripsi data, dan RSA untuk mengenkripsi kunci AES.
10. Kunci Publik: Kunci yang digunakan untuk enkripsi dan dapat diketahui umum.
11. Kunci Privat: Kunci rahasia yang digunakan untuk dekripsi dan hanya diketahui oleh penerima pesan.
12. *Streamlit: Framework Streamlit* untuk membuat aplikasi web interaktif secara cepat, digunakan sebagai platform implementasi sistem kriptografi dalam penelitian ini.
13. Pesan Teks: Data digital berupa karakter atau string yang dikirim dalam bentuk pesan melalui jaringan komunikasi.
14. Operasi *Exclusive OR* (XOR): Operasi logika biner yang menghasilkan nilai 1 jika *input*nya berbeda, dan menghasilkan nilai 0 jika *input*nya sama.

BAB II
KAJIAN TEORI

2.1 Teori Pendukung

2.1.1 Dasar Teori Bilangan dalam Kriptografi

Kriptografi modern khususnya algoritma RSA, banyak bergantung pada konsep dalam teori bilangan. Beberapa konsep utama yang digunakan antara lain keterbagian, kongruensi modulo, bilangan prima, faktor persekutuan terbesar (FPB) atau (GCD), dan fungsi totien Euler (Rosen, 2021; Stein, 2017). Konsep-konsep ini digunakan untuk membentuk kunci publik dan privat dalam sistem kriptografi asimetris RSA (Stallings, 2017).

1. Keterbagian dan Kongruensi Modulo

   Suatu bilangan bulat a dikatakan membagi bilangan bulat b apabila terdapat bilangan bulat k sehingga $b = a \times k$ (Irawan dkk., 2013). Konsep keterbagian ini menjadi dasar operasi aritmetika modular yang digunakan pada sistem kriptografi (Rosen, 2021).

   Dua bilangan a dan b dikatakan kongruen modulo m jika m membagi selisish keduanya, dapat ditulis sebagai:

   $$a \equiv b \pmod{m} \text{ jika dan hanya jika } m|(a-b).$$

   (Stein, 2017).

   Sifat kongruensi ini memungkinkan operasi penjumlahan dan perkalian tetap berlaku dalam sistem modulo, yang menjadi dasar operasi eksponensial modular dalam RSA (Rosen, 2021).

2. Bilangan Prima dan Relatif Prima

Bilangan prima p adalah bilangan bulat positif yang lebih besar dari satu dan hanya memiliki dua faktor positif, yaitu 1 dan p itu sendiri (Irawan dkk., 2013; Rosen, 2012). Teorema dasar aritmetika menyatakan bahwa setiap bilangan bulat $n > 1$ dapat dinyatakan secara unik sebagai hasil kali dari faktor-faktor prima (Rosen, 2012).

Dua bilangan bulat a dan b disebut relatif prima jika tidak memiliki faktor prima yang sama, atau secara formal:

$$\gcd(a, b) = 1.$$

(Irawan dkk., 2013).

Sifat relatif prima ini menjadi syarat utama dalam pemilihan bilangan publik e pada algoritma RSA, yaitu e harus relatif prima terhadap $\varphi(n)$ agar *Inv*ers modulonya dapat dihitung (Stallings, 2017).

3. Faktor Persekutuan Terbesar (FPB)

Faktor persekutuan terbesar (FPB) atau *greatest common divisor* (GCD) dari dua bilangan bulat a dan b adalah bilangan bulat positif terbesar yang dapat membagi keduanya (Irawan dkk., 2013).

Secara matematis dapat dituliskan:

$$\gcd(a, b) = d \Leftrightarrow (d|a) \text{ dan } (d|b).$$

GCD dapat dihitung menggunakan algoritma Euclidean melalui proses rekursif:

$$\gcd(a, b) = \gcd(b, a \bmod b).$$

(Rosen, 2012).

Konsep ini digunakan pada algoritma RSA untuk menentukan nilai d, yaitu *Inv*ers dari e terhadap $\varphi(n)$ dengan memenuhi hubungan $d \equiv e^{-1} \pmod{\varphi(n)}$ (Stallings, 2017).

4. Fungsi Totien Euler $(\varphi(n))$

   Fungsi totien Euler dilambangkan dengan $\varphi(n)$, menyatakan bahwa banyaknya bilangan bulat positif yang lebih kecil atau sama dengan n dan relatif prima terhadap n (Stein, 2017).

   Rumus umum fungsi totien Euler dinyatakan sebagai:

$$\varphi(n) = n \prod_{p|n} \left(1 - \frac{1}{p}\right).$$

   di mana p adalah bilangan prima pembagi n (Rosen, 2012). Jika n merupakan hasil kali dua bilangan prima p dan q, maka:

$$\varphi(n) = (p-1)(q-1).$$

   (Stein, 2017).

   Fungsi totien Euler inilah yang digunakan pada RSA untuk menghitung kunci privat d setelah memilih bilangan publik e (Stallings, 2017).

   Teorema Euler juga menyatakan bahwa jika $\gcd(a, n) = 1$, maka:

$$a^{\varphi(n)} \equiv 1 \pmod{n}.$$

   yang menjadi dasar keamanan algoritma RSA (Rosen, 2012; Stallings, 2017).

5. Keterkaitan Teori Bilangan dengan Algortima RSA

   Semua konsep di atas menjadi fondasi matematis bagi pembentukan kunci pada algoritma *Rivest–Shamir–Adleman* (RSA). Proses pembangkitan kunci RSA dilakukan dengan memilih dua bilangan prima p dan q,

menghitung nilai $n = p \times q$, menentukan fungsi totien Euler $\varphi(n) = (p-1)(q-1)$, memilih bilangan publik e yang relatif prima terhadap $\varphi(n)$, dan menghitung *Inv*ers d sebagai kunci privat yang memenuhi:

$$d \equiv e^{-1} \left(\mathrm{mod}\ \varphi(n)\right).$$

(Stallings, 2017; Putri dkk., 2025).

Dengan demikian, teori bilangan berperan penting dalam mendukung keamanan matematis algoritma RSA yang digunakan pada sistem kriptografi *Hybrid* AES–RSA dalam penelitian ini (Walle, 2024).

2.1.2 Kriptografi: Definisi dan Konsep Dasar

Kriptografi merupakan bidang ilmu dalam matematika terapan dan ilmu komputer yang berfokus pada pengamanan informasi melalui transformasi data agar tidak dapat dibaca oleh pihak yang tidak berwenang (Munir, 2006). Istilah *cryptography* berasal dari bahasa Yunani *kryptos* (tersembunyi) dan *graphein* (menulis), yang berarti "tulisan tersembunyi" (Stallings, 2017). Tujuan utama kriptografi adalah untuk menjamin kerahasiaan (*confidentiality*), integritas (*integrity*), autentikasi (*authentication*), dan *non-repudiation* dalam komunikasi digital (Stallings, 2017).

Secara umum, sitem kriptografi terdiri atas beberapa elemen utama (Munir, 2006):

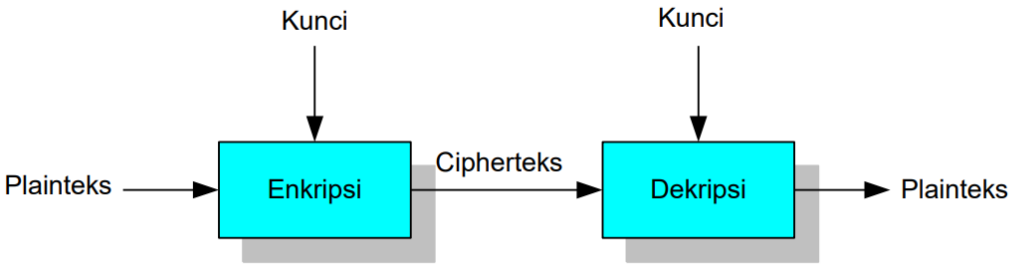
1. *Plaintext*: pesan asli yang dapat dibaca manusia atau diproses oleh sistem.
2. *Ciphertext*: hasil proses enkripsi yang tidak dapat dipahami tanpa proses dekripsi.
3. *Key* atau kunci: nilai rahasia yang digunakan pada proses enkripsi dan dekripsi.

4. Algoritma (E, D): fungsi matematis yang mengubah *plaintext* menjadi *ciphertext* atau mengubah *ciphertext* menjadi *plaintext*.

   Proses kriptografi dapat dinyatakan sebagai berikut:

$$C = E_K(P), \qquad P = D_K(C)$$

   dengan P adalah *plaintext* (pesan teks), C adalah *ciphertext*, K adalah kunci, E adalah fungsi enkripsi, dan D adalah fungsi dekripsi (Stallings, 2017).



Gambar 2.1 Skema Enkripsi dan Dekripsi

Berdasarkan struktur kuncinya, algoritma kriptografi dibagi menjadi dua kategori utama (Stallings, 2017):

1. Kriptografi Simetris: menggunakan satu kunci yang sama untuk proses enkripsi dan dekripsi. Algoritma ini efisien dan cepat untuk data berukuran besar, salah satu contoh yang paling banyak digunakan adalah *Advanced Encryption Standard (AES)* yang bekerja pada blok data.
2. Kriptografi Asimetris: menggunakan sepasang kunci, yaitu kunci publik dan privat. Contohnya adalah algoritma *Rivest–Shamir–Adleman* (RSA) yang memanfaatkan prinsip bilangan prima besar dan eksponen modular untuk menyediakan mekanisme distribusi kunci secara aman.

Kelebihan kriptografi simetris terletak pada kecepatan proses enkripsi dan dekripsi, sedangkan kelemahannya adalah pada manajemen distribusi kunci

rahasia. Sebaliknya, kriptografi asimetris memberikan keamanan tinggi dalam distribusi kunci, tetapi memiliki performa komputasi yang lebih lambat dibandingkan algoritma simetris seperti AES (Farisi dkk., 2023).

2.1.3 Kriptografi Simetris: *Advanced Encryption Standard* (AES)

Advanced Encryption Standard (AES) merupakan algoritma kriptografi simetris berbasis blok yang digunakan secara luas untuk pengamanan data digital (Stallings, 2017). Algoritma ini dikembangkan sebagai pengganti *Data Encryption Standard* (DES) karena kelemahan DES dalam menghadapi serangan modern seperti kriptanalisis dan *brute-force-attack* berbasis teknologi komputasi (Mabruri, 2020). AES secara resmi diadopsi oleh *National Institute of Standards and Technology* (NIST) pada tahun 2001 sebagai standar enkripsi global karena memiliki efisiensi tinggi dan ketahanan terhadap berbagai bentuk serangan kriptanalisis (Farisi dkk., 2023).

AES menggunakan ukuran blok tetap 128 bit, dengan panjang kunci 128, 192, atau 256 bit tergantung tingkat keamanan yang diinginkan (Stallings, 2017). Proses enkripsi AES terdiri atas beberapa putaran transformasi, yakni *SubBytes*, *ShiftRows*, *MixColumns*, dan *AddRoundKey*, yang menghasilkan difusi dan non-linearitas tinggi sehingga *ciphertext* sangat sulit direkonstruksi tanpa kunci yang benar (Mabruri, 2020; Farisi dkk., 2023).

Secara matematis, algoritma *Advanced Encryption Standard* (AES) bekerja dalam bentuk *state matrix* 4×4 *byte*, dengan mengenkripsi data dalam 10 putaran (*round*) untuk AES-128. *Round* 1-9: *SubBytes* → *ShiftRows* → *MixColumns* → *AddRoundKey*. *Round* 10 (final): *SubBytes* → *ShiftRows* → *AddRoundKey* (tanpa *MixColumns*). Bentuk *state matrix* 4×4 :

$$S = \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} \quad (2.1)$$

dengan $s_{i,j}$ adalah *byte* ke-(i, j) dalam *state* (Mabruri, 2020; Stallings, 2017).

Berikut ini adalah tahapan utama dalam proses enkripsi AES:

1. *SubBytes*

   SubBytes merupakan proses substitusi non-linierterhadap setiap elemen *byte* dalam *state* berdasarkan tabel substitusi (*S-box*):

   $$S'(x) = S[x] \quad (2.2)$$

   yaitu setiap *byte* x diganti dengan nilai dari tabel *S-box.*

   Contoh:

   $$S\text{-}box\ (32) = 23$$

   $$0 \times 32 \rightarrow S - box[3][2] = 23$$

   (Mabruri, 2020).

L(xy)	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
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	12	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	d5	b3	29	e3	2f	84
5	53	dl	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	00c	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	1v	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 2.2 Tabel *S-box* AES

2. *ShiftRows*

Transformasi ini bertujuan untuk menyebarkan *byte* dalam *state* ke posisi yang berbeda melalui rotasi baris ke kiri. Skema rotasi adalah sebagai berikut:

a. Baris ke-0: tidak digeser

b. Baris ke-1: digeser 1 *byte* ke kiri

c. Baris ke-2: digeser 2 *byte* ke kiri

d. Baris ke-3: digeser 3 *byte* ke kiri

Secara matematis, rotasi baris dapat dinyatakan sebagai:

$$Row_i = ShiftLeft(Row_i i) \tag{2.3}$$

Transformasi ini tidak mengubah nilai *byte* tetapi memindahkannya untuk mengacaukan korelasi antar kolom (Farisi dkk., 2023).

3. *MixColumns*

Transformasi *MixColumns* berfungsi mencampur *byte* dalam setiap kolom *state* melalui operasi linear di medan hingga $\mathrm{GF}(2^8)$.

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \times \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix}$$

Multiplikasi dalam $\mathrm{GF}(2^8)$ dilakukan menggunakan operasi XOR dan pergeseran bit. Sebagai contoh:

$$02 \cdot x = \begin{cases} x \ll 1, & jika\ bit\ tertinggi = 0 \\ (x \ll 1) \oplus 0x1B, & jika\ bit\ tertinggi = 1 \end{cases}$$

$$Kolom = \begin{bmatrix} d4 \\ bf \\ 5d \\ 30 \end{bmatrix} \Rightarrow \begin{bmatrix} 2 \times d4 & 3 \times bf & 1 \times 5d & 1 \times 30 \\ 1 \times d4 & 2 \times bf & 3 \times 5d & 1 \times 30 \\ 1 \times d4 & 1 \times bf & 2 \times 5d & 3 \times 30 \\ 3 \times d4 & 1 \times bf & 1 \times 5d & 2 \times 30 \end{bmatrix} Kolom' = \begin{bmatrix} 04 \\ e0 \\ 48 \\ 28 \end{bmatrix}$$

Langkah perhitungan untuk baris pertama:

a. $2 \times d4$:

   i. $d4\ (11010100) \rightarrow$ geser ke kiri $\rightarrow 10101000 = \text{a8}$

   ii. Karena bit kiri keluar, XOR dengan $0 \times 1B$: $a8 \oplus 1b = b3$

b. $3 \times bf$:

   i. $bf \times 2 = (1011111 \ll 1) = (0111110) = 7e$

   ii. Karena bit kiri keluar, XOR dengan $0 \times 1B$: $7e \oplus 1b = 65$

   iii. Lalu $(\times 3) = (\times 2 \text{ hasil}) \oplus \text{nilai asli} = 65 \oplus \text{bf} = \text{da}$

c. $1 \times 5d = 5d$

d. $1 \times 30 = 30$

Gabungkan hasil dengan XOR:

$$b3 \oplus da \oplus 5d \oplus 30$$

Hitung satu per satu (menggunakan kalkulator heksadesima XOR atau tabel biner):

a. $b3 \oplus da = 69$

b. $69 \oplus 5d = 34$

c. $34 \oplus 30 = 04$

$$s_0' = 04$$

Transformasi ini memastikan *avalanche effect* di mana perubahan satu bit pada *plaintext* dapat memengaruhi seluruh kolom *ciphertext* (Mabruri, 2020).

4. *AddRoundKey*

Menurut Stallings (2017), tahap *AddRoundKey* merupakan proses penggabungan antara *state matrix* dengan *round key* menggunakan operasi

XOR pada setiap ronde enkripsi AES. Tahapan ini berfungsi untuk menambahkan efek kunci terhadap data sehingga setiap bit hasil enkripsi selalu bergantung pada nilai kunci rahasia.

$$State' = State \oplus RoundKey \quad (2.4)$$

Contoh:

$$32 \oplus 2B = 19$$

(Mabruri, 2020) Farisi dkk., 2023).

Proses Dekripsi AES dilakukan dengan urutan operasi yang berlawanan arah dari proses enkripsi. Setiap blok *ciphertext* diproses menggunakan fungsi *Invers* dari setiap tahap, yaitu *InvShiftRows, InvSubBytes, InvMixColumns,* dan *AddRoundKey.*

Berikut ini adalah tahapan utama dalam proses dekripsi AES:

1. *InvShiftRows*

   InvShiftRows merupakan proses *invers* dari *ShiftRows* yang bertujuan mengembalikan posisi *byte* dalam *state matrix* melalui rotasi baris ke kanan.

   Skema rotasi adalah sebagai berikut:

   a. Baris ke-0: tidak digeser

   b. Baris ke-1: digeser 1 *byte* ke kanan

   c. Baris ke-2: digeser 2 *byte* ke kanan

   d. Baris ke-3: digeser 3 *byte* ke kanan

   Secara matematis, rotasi baris dapat dinyatakan sebagai:

   $$Row_i = ShiftRight\ (Row_i) \quad (2.5)$$

   Transformasi ini tidak mengubah nilai *byte*, tetapi mengembalikan posisi *byte* seperti sebelum proses enkripsi.

2. *InvSubBytes*

InvSubBytes merupakan proses substitusi invers terhadap setiap elemen byte dalam state menggunakan tabel Inverse S-box.

$$S'(x) = S^{-1}[x] \tag{2.6}$$

yaitu setiap byte x diganti dengan nilai pada tabel *Inverse S-box*.

Contoh:

$$S^{-1}(23) = 32$$

Transformasi ini membalik proses substitusi non-linear pada tahap *SubBytes* (Mabruri, 2020).

L(xy)	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	79
9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
a	47	f1	1a	71	1f	29	c5	89	6f	b7	62	0e	aa	18	be	1b
b	fc	56	3e	4b	v6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
d	60	51	7f	a9	19	b5	4a	0d	2d	e4	7a	9f	93	c9	9c	ef
e	a0	e0	3b	4d	sr	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
f	17	2b	04	7e	bs	77	d6	26	e1	69	14	63	55	21	0c	7d

Gambar 2.3 Tabel *Invers S-Box* AES

3. *AddRoundKey*

Menurut Stallings (2017), tahap *AddRoundKey* merupakan proses penggabungan antara *state matrix* dengan *round key* menggunakan operasi XOR pada setiap ronde dekripsi AES.

$$State' = State \oplus RoundKey \quad (2.7)$$

Contoh:

$$19 \oplus 2B = 32$$

Transformasi ini bersifat reversibel karena menggunakan operasi XOR, sehingga dapat mengembalikan nilai *state* sebelumnya (Mabruri, 2020; Farisi dkk., 2023).

4. *InvMixColumns*

Transformasi *InvMixColumns* berfungsi mengembalikan hasil pencampuran kolom pada proses enkripsi melalui operasi linear pada medan hingga $GF(2^8)$.

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \times \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix}$$

Multiplikasi dilakukan menggunakan operasi XOR dan pergeseran bit dalam $GF(2^8)$. Transformasi ini mengembalikan efek difusi dari proses *MixColumns* sehingga struktur data kembali seperti sebelum pencampuran kolom.

Proses dekripsi AES dilakukan dengan urutan sebagai berikut:

$$P = D_K(C)$$

$$= InvShiftRows\left(InvSubBytes\left(AddRoundKey\left(InvMixColumns(C, K)\right)\right)\right)$$

di mana P adalah *plaintext*, C adalah *ciphertext*, dan K adalah kunci simetris yang sama dengan kunci enkripsi (Stallings, 2017; NIST, 2001). Transformasi tersebut dilakukan secara berulang pada setiap ronde dengan menggunakan *round key* dalam urutan terbalik hingga diperoleh kembali *plaintext*.

AES memiliki keunggulan utama berupa efisiensi tinggi dan ketahanan yang kuat terhadap serangan *brute-force* karena panjang kunci yang fleksibel dan struktur blok tetap. AES juga menunjukkan stabilitas performa yang baik di berbagai platform, termasuk aplikasi mobile dan sistem pesan teks, dengan waktu enkripsi rata-rata di bawah lima detik untuk file berukuran 200 KB (Farisi dkk., 2023). Namun, kelemahan utama AES terletak pada proses distribusi kunci. Karena algoritma ini simetris, kedua pihak harus menyimpan kunci yang sama (Khaing Oo & Soe, 2019). Apabila kunci berhasil dicegat oleh pihak tidak berwenang, maka keamanan sistem terganggu karena AES bergantung sepenuhnya pada kerahasiaan kuncinya (Chang dkk., 2025).

Pada penelitian ini, AES yang digunakan adalah AES-128 dengan ukuran blok 128 bit dan diimplementasikan dalam mode operasi *Cipher Block Chaining* (CBC). Setiap *plaintext* dibagi ke dalam blok 16 *byte* dan dilakukan penambahan *padding* (misalnya skema PKCS#7) sehingga panjang *ciphertext* selalu menjadi kelipatan 16 *byte*. Proses enkripsi pada AES-CBC juga memanfaatkan vektor inisialisasi (*Initialization Vector*, IV) berukuran 128 bit yang dibangkitkan secara acak untuk setiap sesi enkripsi, sehingga *ciphertext* yang dihasilkan berbeda walaupun *plaintext* dan kunci yang digunakan sama.

2.1.4 Kriptografi Asimetris: *Rivest-Shamir-Adleman* (RSA)

Algoritma *Rivest–Shamir–Adleman* (RSA) merupakan sistem kriptografi asimetris yang menggunakan sepasang kunci berbeda, yaitu kunci publik untuk enkripsi dan kunci privat untuk dekripsi (Stallings, 2017). Keamanan algoritma RSA bergantung pada kompleksitas faktorisasi bilangan prima besar. Chang dkk. (2025) menjelaskan bahwa belum ada algoritma efisien yang mampu

menyelesaikan faktorisasi tersebut dengan cepat, sehingga RSA tetap menjadi metode yang aman untuk sistem *Hybrid*. Prinsip utama RSA didasarkan pada *trapdoor one-way function*, yaitu fungsi matematis yang mudah dihitung ke satu arah namun sangat sulit dibalik tanpa informasi tertentu (Putri dkk., 2025).

Proses pembangkitan pasangan kunci RSA dilakukan melalui beberapa tahap matematis. Pertama, pilih dua bilangan prima besar p dan q, dengan $p \neq q$. Kemudian hitung nilai modulus:

$$n = p \times q \tag{2.8}$$

Selanjutnya, hitung fungsi totien euler:

$$\varphi(n) = (p-1)(q-1) \tag{2.9}$$

Kemudian pilih bilangan bulat e sedemikian sehingga:

$$1 < e < \varphi(n), \qquad \gcd\big(e, \varphi(n)\big) = 1 \tag{2.10}$$

Langkah berikutnya adalah menghitung d sebagai *Inv*ers dari e terhadap modulo $\big(\varphi(n)\big)$:

$$d \equiv e^{-1} \left(mod\ \varphi(n)\right) \tag{2.11}$$

Pasangan kunci RSA kemudian dinyatakan sebagai:

1. Kunci Publik: (e, n)
2. Kunci Privat: (d, n)

(Fatonah dkk., 2022; Stallings, 2017).

Proses enkripsi dan dekripsi pada RSA melibatkan operasi perpangkatan modular. Enkripsi dilakukan dengan kunci publik menggunakan persamaan:

$$C \equiv M^e\ (mod\ n) \tag{2.12}$$

sedangkan dekripsi menggunakan kunci privat:

$$M \equiv C^d \ (mod\ n) \tag{2.13}$$

dengan M adalah *plaintext* dan C adalah *ciphertext* (Menezes dkk., 1996; Putri dkk., 2025).

Contoh:

1. Pilih dua bilangan prima berbeda, sebagai contoh perhitungan sederhana:

   $$p = 61,\ \ q = 53$$

   Dalam implementasi nyata, RSA menggunakan bilangan prima besar untuk menjamin keamanan, tetapi contoh ini hanya digunakan untuk ilustrasi konsep perhitungan manual.

2. Modulus:

   $$n = p \times q = 61 \times 53 = 3233$$

3. Totien Euler:

   $$\varphi(n) = (p-1)(q-1) = 60 \times 52 = 3120$$

4. Pilih eksponen publik:

   $$e = 17, \qquad \gcd(17{,}3120) = 1$$

5. Hitung eksponen privat:

   $$d \equiv e^{-1} \left(mod\ \varphi(n)\right)$$

   $$d \equiv 17^{-1} \ (mod\ 3120)$$

   Hasil dengan algoritma Euclid diperluas:

   $$d = 2753 \ (sebab\ 17 \times 2753 = 46801 \equiv 1 \ (mod\ 3120)$$

6. Pasangan kunci RSA:

   a. Kunci Publik: $(e, n) = (17, 3233)$

   b. Kunci Privat: $(d, n) = (2753, 3233)$

7. Proses enkripsi pesan:

   Ambil pesan (dalam bilangan):

$$M = 42, \ \ 0 < M < n$$

   Enkripsi dengan kunci publik:

$$C \equiv M^e \ (mod\ n)$$

$$C \equiv 42^{17} \ (mod\ 3233)$$

$$C = 2557$$

8. Contoh Dekripsi

   Dekripsi dengan kunci privat:

$$M \equiv C^d \ (mod\ n)$$

$$M \equiv 2557^{2753} \ (mod\ 3233)$$

$$M = 42$$

(Stallings, 2017).

RSA memiliki keunggulan karena mendukung distribusi kunci publik tanpa perlu pertukaran kunci rahasia secara manual, serta dapat digunakan untuk autentikasi data melalui tanda tangan digital (Chang dkk., 2025). Dalam sistem *Hybrid* AES–RSA, RSA sering digunakan untuk mengenkripsi kunci simetris seperti AES, sehingga menjaga keamanan sekaligus mempertahankan efisiensi proses enkripsi data (Putri dkk., 2025). RSA juga diterapkan secara luas pada protokol *Transport Layer Security* (TLS), di mana kunci publik digunakan untuk mengenkripsi *session key* sehingga hanya penerima dengan kunci privat yang dapat mendekripsinya (Walle, 2024).

2.1.5 *Hybrid Cryptography* (AES-RSA)

Hybrid cryptography merupakan pendekatan gabungan antara algoritma simetris dan asimetris untuk memperoleh keamanan dan efisiensi secara bersamaan (Mitali dkk., 2021). Dalam skema ini, AES digunakan untuk mengenkripsi pesan teks karena kecepatan dan efisiensinya dalam memproses data, sedangkan RSA digunakan untuk mengenkripsi kunci AES agar dapat dikirim secara aman melalui jaringan publik (Walle, 2024). Kombinasi keduanya menciptakan sistem enkripsi berlapis yang mampu melindungi data sekaligus menjaga efisiensi pemrosesan (Rasheed & Kumar, 2025).

Secara matematis, proses *Hybrid* AES–RSA dapat digambarkan melalui tahapan:

1. Enkripsi:

$$K_{AES} \leftarrow GenKey_{AES(128\ bit)}$$

$$C_{text} = E_{K_{AES}}(P)$$

$$C_{key} = E_{e,n}(K_{AES})$$

$$C_{Total} = (C_{text}, C_{key})$$

2. Dekripsi:

$$K_{AES} = D_{d,n}(C_{key})$$

$$P = D_{K_{AES}}(C_{text})$$

$$P \rightarrow Plaintext$$

dengan P adalah *plaintext*, C_{text} adalah *ciphertext*, C_{key} adalah kunci AES yan di enkripsi menggunakan RSA, K_{AES} adalah kunci simetris AES, serta (e, n) dan (d, n) masing-masing adalah kunci publik RSA dan kunci privat RSA (Chang dkk., 2025).

Tahapan utamanya meliputi:

1. Pengirim mengenkripsi pesan teks menggunakan AES dengan kunci K_{AES}.
2. Kunci K_{AES} tersebut dienkripsi menggunakan RSA dengan kunci publik penerima.
3. Penerima mengembalikan kunci AES menggunakan kunci privat RSA.
4. Penerima menggunakan K_{AES} yang telah didekripsi untuk mendekripsi pesan asli.

(Khaing Oo & Soe, 2019).

Pendekatan ini efektif karena RSA hanya digunakan untuk mengenkripsi kunci kecil (128–256 bit), sehingga tidak membebani komputasi, sedangkan AES menangani data besar dengan kecepatan tinggi (Mitali dkk., 2021). Chang dkk (2025) menunjukkan bahwa modifikasi algoritma *Hybrid* AES–RSA dalam mode MRA mampu meningkatkan efisiensi proses enkripsi dan dekripsi secara signifikan dibanding RSA konvensional, terutama dalam pengujian data besar pada perangkat IoT, sementara penelitian oleh Walle (2024) membuktikan peningkatan *throughput* sebesar 0,4% dan penurunan latensi 3,6%. Selain itu, Rasheed & Kumar (2025) menyatakan bahwa model *Hybrid* dapat menjaga *confidentiality*, *integrity*, dan *non-repudiation* secara simultan dalm sistem pesan teks dan cloud.

Dengan demikian, *Hybrid* AES–RSA menjadi solusi yang sangat efektif karena memadukan kecepatan enkripsi AES dengan keamanan distribusi kunci RSA, sehingga sangat sesuai digunakan dalam pengamanan pesan teks dan komunikasi digital modern (Rasheed & Kumar, 2025).

2.1.6 *Framework Streamlit* untuk Implementasi Kriptografi

Streamlit adalah kerangka kerja *Python* yang memungkinkan pembuatan aplikasi web interaktif dengan cepat tanpa menulis HTML, CSS, atau *JavaScript* (Santhoshi dkk., 2024). Keunggulan utama *Streamlit* adalah kemampuannya memperbarui tampilan secara *real-time*, sehingga sangat cocok untuk eksperimen yang mengukur performa algoritma secara langsung (Lee dkk., 2022). Dalam konteks kriptografi, *Streamlit* memfasilitasi prototipe enkripsi dan dekripsi pesan teks antarmuka pengguna dengan instan, sehingga pengguna dapat mengamati hasil *ciphertext* langsung dari masukan teks (Santhoshi dkk., 2024).

Integrasi *Streamlit* dengan pustaka kriptografi *Python*, seperti *PyCryptodome*, memungkinkan implementasi AES dan RSA secara langsung dalam aplikasi web ringan (Santhoshi dkk., 2024). Untuk mengukur kecepatan, modul seperti `time.perf_counter()` dapat digunakan di *backend Streamlit* untuk menghitung lama enkripsi (T_{enc}) dan dekripsi (T_{dec}) secara *real-time* (Santhoshi dkk., 2024). Hasil waktu kemudian dapat ditampilkan di dashboard *Streamlit* dalam bentuk metrik atau grafik, memudahkan analisis langsung tanpa perlu beralih ke alat eksternal.

Studi *"Crypto Analysis and Visualization using Python and Streamlit"* menunjukkan bahwa aplikasi berbasis *Streamlit* dapat digunakan untuk visualisasi kriptografi, meskipun penelitian tersebut fokus pada aspek analisis data, tetapi menunjukkan bahwa penggunaan *Streamlit* dalam domain kripto cukup aplikatif (Santhoshi dkk., 2024). Oleh karena itu, dalam penelitian ini *Streamlit* dipilih sebagai platform utama untuk menguji kecepatan enkripsi–dekripsi AES–RSA pada pesan teks digital.

2.1.7 Pengamanan Pesan Teks

Pesan teks merupakan salah satu bentuk komunikasi digital yang paling sering digunakan, baik melalui aplikasi chat, email, maupun form *input* pada sistem web. Karena karakteristiknya berupa *string* sederhana yang mudah dibaca manusia, pesan teks sangat rentan terhadap serangan seperti *sniffing* dan *man-in-the-middle attack*, di mana penyerang dapat menyadap atau memodifikasi isi pesan selama transmisi (Walle, 2024). Selain itu, serangan *database breach* juga dapat terjadi ketika penyimpanan pesan di server berhasil ditembus sehingga data asli dapat diakses pihak yang tidak berwenang (Akter dkk., 2023).

Untuk mengatasi kelemahan masing-masing sistem enkripsi, pendekatan *Hybrid cryptography* yang menggabungkan algoritma simetris AES dan algoritma asimetris RSA diterapkan sebagai solusi yang efisien dalam menjamin keamanan data (Chang dkk., 2025). Dalam kombinasi ini, AES berfungsi untuk mengenkripsi isi pesan teks (*plaintext*) menjadi *ciphertext* sehingga isi pesan tidak dapat dibaca tanpa kunci yang sesuai, sedangkan RSA digunakan untuk mengenkripsi kunci AES (*session key*) agar distribusi kunci tetap aman selama proses komunikasi (Hermawan & Ujianto, 2021).

Secara matematis, proses enkripsi pesan teks P dapat dituliskan sebagai:

$$C_D = E_{K_s}(P), \quad C_K = RSA\_Enc(K_s)$$

di mana:

1. P adalah *plaintext* pesan teks.
2. C_D adalah *ciphertext* hasil enkripsi AES.
3. C_K adalah kunci AES yang dienkripsi RSA.

(Khaing Oo & Soe, 2019).

Selain menjaga kerahasiaan (*confidentiality*), sistem *Hybrid* AES–RSA juga meningkatkan integritas pesan, sebab perubahan satu bit saja pada *ciphertext* akan menghasilkan hasil dekripsi yang tidak valid (Rasheed & Kumar, 2025). Prinsip autentikasi dapat pula dicapai dengan menambahkan tanda tangan digital (*digital signature*) sehingga penerima dapat memverifikasi identitas pengirim dengan aman (Akter dkk., 2023).

Hasil penelitian menunjukkan bahwa kombinasi AES–RSA lebih efisien dan aman dibanding penggunaan algoritma tunggal. Misalnya, Walle (2024) membuktikan bahwa skema AES–RSA mampu meningkatkan efisiensi dekripsi hingga 61 kali dibanding RSA murni ketika ukuran pesan mencapai 6 KB, sedangkan Penelitian oleh Chang dkk. (2025) memperlihatkan peningkatan signifikan baik dari sisi keamanan maupun efisiensi energi pada komunikasi IoT berdaya rendah menggunakan enkripsi *Hybrid* AES–RSA dalam mode MRA. Dengan demikian, penerapan *Hybrid* AES–RSA dalam pengamanan pesan teks terbukti efektif dalam menjamin kecepatan, kerahasiaan, integritas secara empiris pada proses enkripsi dan dekripsi pesan teks.

2.2 Integrasi Nilai-Nilai Al-Qur’an dalam Menjaga Keamanan

Perkembangan teknologi informasi telah memberikan kemudahan besar dalam komunikasi digital, khususnya melalui pertukaran pesan teks yang cepat dan efisien. Namun, kemajuan ini juga membawa risiko serius, seperti kebocoran data, manipulasi pesan, dan pelanggaran privasi. Dalam Islam, keamanan informasi bukan hanya sekadar aspek teknis, tetapi juga bagian dari tanggung jawab moral yang melekat pada nilai amanah dan keadilan.

Nilai amanah merupakan salah satu prinsip fundamental dalam ajaran Islam yang menuntut setiap individu untuk menjaga apa yang telah dipercayakan kepadanya. Allah SWT berfirman dalam Surah An-Nisa ayat 58:

اِنَّ اللّٰهَ يَأْمُرُكُمْ اَنْ تُؤَدُّوا الْاَمٰنٰتِ اِلٰٓى اَهْلِهَاۙ وَاِذَا حَكَمْتُمْ بَيْنَ النَّاسِ اَنْ تَحْكُمُوْا بِالْعَدْلِۗ اِنَّ اللّٰهَ نِعِمَّا يَعِظُكُمْ بِهٖۗ اِنَّ اللّٰهَ كَانَ سَمِيْعًاۢ بَصِيْرًا ﴿٥٨﴾

"Sesungguhnya Allah menyuruh kamu menyampaikan amanat kepada yang berhak menerimanya, dan (menyuruh kamu) apabila menetapkan hukum di antara manusia supaya kamu menetapkan dengan adil. Sesungguhnya Allah memberi pengajaran yang sebaik-baiknya kepadamu. Sesungguhnya Allah adalah Maha Mendengar lagi Maha Melihat." (NU Online, 2025).

Tafsir QS. An-Nisa ayat 58 menjelaskan bahwa ayat ini mengandung prinsip moral yang fundamental, yakni kewajiban untuk menyampaikan amanah kepada pihak yang berhak serta menegakkan keadilan dalam setiap keputusan. Konsep amanah mencakup seluruh bentuk tanggung jawab, baik dalam ranah sosial, hukum, maupun profesional. Dalam konteks komunikasi digital, amanah dimanifestasikan melalui perlindungan terhadap integritas, kerahasiaan, dan penggunaan informasi secara etis. Menjaga data dan informasi agar tidak bocor, rusak, atau disalahgunakan merupakan implementasi nyata dari perintah tersebut. Melaksanakan amanah secara tepat waktu dan akurat dalam dunia digital merupakan wujud penghayatan nilai-nilai Islam di era teknologi informasi.

Islam juga menekankan pentingnya usaha aktif dan peningkatan kualitas diri dalam mencapai kebaikan. Allah Swt. berfirman dalam surah Ar-Ra'd ayat 11:

اِنَّ اللّٰهَ لَا يُغَيِّرُ مَا بِقَوْمٍ حَتّٰى يُغَيِّرُوْا مَا بِاَنْفُسِهِمْۗ

"Sesungguhnya Allah tidak mengubah keadaan suatu kaum hingga mereka mengubah apa yang ada pada diri mereka" (NU Online, 2025).

Tafsir Al-Wajiz menjelaskan perubahan kondisi suatu kaum bergantung pada kemauan mereka untuk berubah dari dalam, terutama dalam sikap, mentalitas,

dan usaha yang dilakukan. Prinsip ini relevan dalam konteks pengelolaan keamanan informasi, yang menuntut peningkatan kualitas dan inovasi berkelanjutan. Dalam penelitian ini, transformasi dari penggunaan satu algoritma menuju penggabungan dua algoritma merupakan upaya menuju sistem keamanan digital yang lebih efektif dan andal. Pendekatan ini tidak hanya bersifat teknis, tetapi juga mencerminkan nilai-nilai amanah dan semangat perubahan positif sebagaimana diajarkan dalam Al-Qur'an.

Dengan demikian, keamanan informasi tidak semata-mata dilihat dari perspektif ilmiah dan teknologis, melainkan juga mengandung dimensi etika dan spiritual. Hal ini mencerminkan pengamalan nilai-nilai *ta'dib* (pendidikan moral) dalam era digital, di mana teknologi dimanfaatkan bukan hanya untuk alat efisiensi, tetapi juga sebagai sarana menjaga keadilan, kepercayaan, dan tanggung jawab sosial dalam komunikasi virtual.

2.3 Kajian Topik dengan Teori Pendukung

Perkembangan teknologi digital telah mengubah cara manusia bertukar informasi, terutama melalui pesan teks yang menjadi media komunikasi utama pada berbagai platform. Namun, meningkatnya aktivitas digital juga menimbulkan risiko kebocoran data dan serangan siber seperti *phishing*, *sniffing*, dan *man-in-the-middle attack*. Untuk itu, dibutuhkan mekanisme pengamanan yang mampu menjaga kerahasiaan dan keaslian pesan dengan proses yang cepat dan terukur.

Penelitian ini berlandaskan teori dasar kriptografi dan matematika, seperti konsep keterbagian, kongruensi modulo, bilangan prima, faktor persekutuan terbesar (GCD), serta fungsi totien Euler yang digunakan dalam pembentukan kunci algoritma RSA. RSA merupakan algoritma asimetris yang mengandalkan kesulitan

faktorisasi bilangan prima besar untuk menjaga keamanan kunci. Sementara itu, *Advanced Encryption Standard* (AES) merupakan algoritma simetris berbasis blok yang unggul dalam kecepatan enkripsi–dekripsi melalui transformasi *substitution–permutation network* dalam medan hingga $GF(2^8)$.

Kombinasi keduanya membentuk *Hybrid cryptography* AES–RSA, di mana AES mengenkripsi pesan teks dengan cepat, sedangkan RSA mengenkripsi kunci AES untuk menjamin keamanan distribusi kunci. Pendekatan ini menghasilkan sistem kriptografi yang kuat dan efisien terhadap waktu proses, karena masing-masing algoritma saling menutupi kelemahan satu sama lain.

Dalam implementasinya, penelitian ini memanfaatkan *framework Streamlit* sebagai antarmuka interaktif untuk menjalankan proses enkripsi dan dekripsi secara langsung serta mencatat waktu eksekusinya. Dengan dukungan teori matematika, kriptografi, dan pemrograman *Python*, penelitian ini berfokus pada pengukuran waktu proses enkripsi–dekripsi pesan teks menggunakan metode *Hybrid* AES–RSA sebagai indikator performa sistem keamanan digital.

Berdasarkan seluruh kajian teori pendukung, dapat disimpulkan bahwa kombinasi konsep RSA, AES, *Hybrid cryptography* AES–RSA, teori bilangan, dan implementasi berbasis *Streamlit* menjadi dasar ilmiah yang kuat untuk mendukung penelitian ini, khususnya dalam mengukur kecepatan proses enkripsi dan dekripsi pesan teks secara empiris.

BAB III
METODE PENELITIAN

3.1 Jenis Penelitian

Penelitian ini termasuk penelitian kuantitatif eksperimental bersifat terapan (*applied research*). Penelitian terapan dipilih karena bertujuan menganalisis kinerja waktu komputasi algoritma *Hybrid* AES–RSA dalam mengamankan pesan teks serta menganalisis waktu proses enkripsi dan dekripsi secara empiris. Waktu proses enkripsi (T_{enc}) dan dekripsi (T_{dec}) yang dihitung dalam satuan detik menggunakan `time.perf_counter()` fungsi pada *Python*. Nilai waktu rata-rata digunakan sebagai indikator performa algoritma *Hybrid* AES–RSA.

Pendekatan kuantitatif digunakan untuk memperoleh data numerik yang dapat diukur, seperti waktu enkripsi (T_{enc}) dan waktu dekripsi (T_{dec}), yang kemudian dianalisis untuk mengetahui performa sistem. Hasil pengujian dilakukan menggunakan perangkat lunak berbasis *Python* dengan *framework Streamlit* untuk memperoleh hasil pengukuran secara *real-time*.

3.2 Pra Penelitian

Tahapan pra-penelitian meliputi kegiatan sebagai berikut:

1. Studi Literatur

   Mengumpulkan referensi dari jurnal, buku, dan artikel ilmiah terbaru (2020–2025) yang membahas algoritma RSA, AES, dan model *Hybrid cryptography* AES–RSA. Kajian difokuskan pada aspek matematis dan performa waktu proses.

2. *Input* Data Penelitian

   Input data dalam penelitian ini berupa:

a. Pesan teks (*plaintext*) dengan panjang karakter berbeda, yaitu: 50, 75, 100, 150, dan 200 karakter.

b. Bentuk *input* berupa *string*, digunakan untuk mensimulasikan pesan digital yang akan diamankan.

3. Desain Awal Sistem

Sistem dirancang dengan dua tahap enkripsi:

a. AES untuk mengenkripsi pesan teks.

b. RSA untuk mengenkripsi kunci AES.

Alur *input* yang digunakan → enkripsi → dekripsi → *output* pada platform *Streamlit*.

4. Validasi Teori dan Algoritma

Melakukan uji coba kecil untuk memastikan fungsi dasar enkripsi RSA dan AES berjalan sesuai algoritma matematis sebelum digabungkan ke dalam sistem *Hybrid*.

3.3 Tahapan Penelitian

Tahapan yang perlu dilakukan untuk melakukan penelitian ini yaitu sebagai berikut:

3.3.1 Proses Enkripsi *Hybrid* AES-RSA

Alur proses enkripsi ditunjukkan pada Gambar 3.1. Gambar tersebut menjelaskan rangkaian langkah mulai dari *input* pesan teks, pembangkitan kunci AES, proses enkripsi menggunakan algoritma AES, hingga tahapan enkripsi kunci AES menggunakan RSA. Gambar tersebut membantu memahami bagaimana kedua algoritma bekerja bersama untuk menghasilkan *ciphertext* dan kunci yang telah diamankan.

Langkah-Langkah Proses Enkripsi:

1. *Input* pesan teks (*plaintext*)

   Pengguna memasukkan pesan teks dengan panjang tertentu.

2. Generate kunci AES (K_{AES})

   Sistem membangkitkan kunci AES 128-bit secara acak.

3. Enkripsi pesan menggunakan AES

   Plaintext dienkripsi menggunakan AES-128 dalam mode CBC dengan vektor inisialisasi (IV) acak. Secara matematis dituliskan:

   $$C_{text} = E_{AES-CBC}(K_{AES}, IV, P)$$

   dengan P adalah *plaintext*, K_{AES} adalah kunci AES 128-bit, IV adalah vektor inisialisasi 128 bit, E_{AES} adalah fungsi enkripsi AES, C adalah *ciphertext*. Karena menggunakan *padding* dan blok 128-bit, panjang *output* C_{text} selalu berupa kelipatan 16 *byte*.

4. Enkripsi kunci AES menggunakan RSA

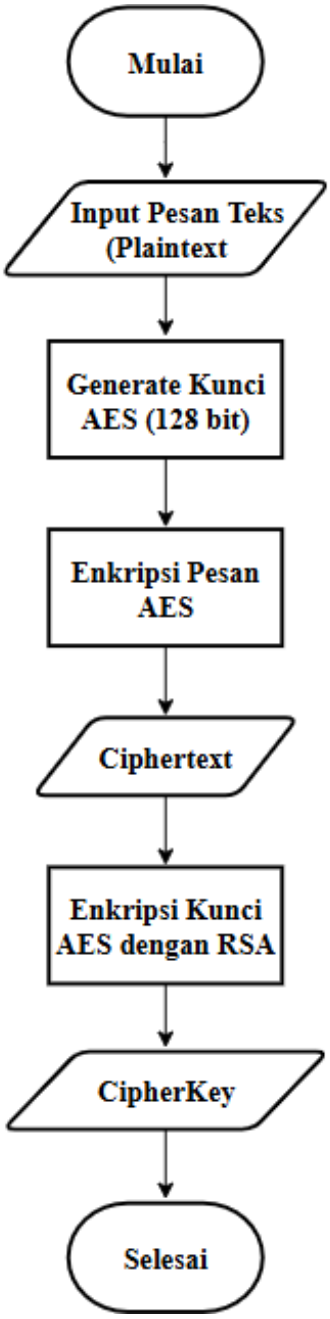
   K_{AES} dienkripsi dengan kunci publik RSA:

   $$C_{key} = E_{RSA}(K_{pub}, K_{AES})$$

   dengan K_{pub} adalah kunci publik RSA, C_{key} adalah kunci AES yang sudah dienkripsi RSA.

5. Hasil akhir simulasi enkripsi

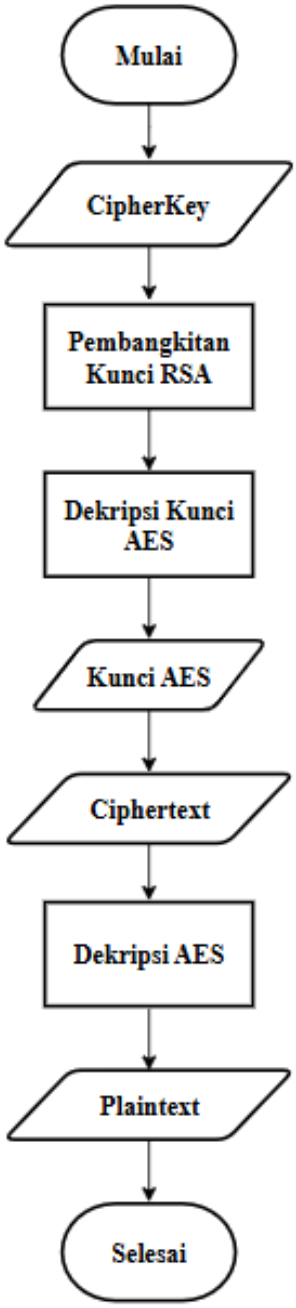
   Output yang dihasilkan adalah *ciphertext* dan *cipherkey*.



Gambar 3.1 *Flowchart* Proses Enkripsi

3.3.2 Proses Dekripsi *Hybrid* AES-RSA

Alur proses dekripsi ditunjukkan pada Gambar 3.2. Gambar ini menunjukkan tahapan mulai dari penerimaan *ciphertext* dan *cipherkey*, proses dekripsi RSA untuk mengembalikan kunci AES, hingga penggunaan kunci tersebut untuk mendekripsi *ciphertext* menjadi *plaintext* semula.



Gambar 3.2 *Flowchart* Proses Dekripsi

Langkah-Langkah Proses Dekripsi:

1. *Input ciphertex* dan *cipherkey*

   Sistem menerima kedua *output* dari proses enkripsi.

2. Dekripsi RSA untuk mendapatkan K_{AES}

$$K_{AES} = D_{RSA}(K_{priv}, C_{key}),$$

dengan K_{priv} adalah kunci privat RSA, D_{RSA} adalah fungsi dekripsi RSA.

3. Dekripsi AES untuk mengembalikan *plaintext*

$$P' = D_{AES}(K_{AES}, C_{text}),$$

dengan P' adalah *plaintext* hasil dekripsi.

4. Validasi hasil dekripsi

Proses berhasil apabila:

$$P' = P.$$

Artinya *plaintext* setelah dekripsi sama persis dengan *plaintext* awal.

3.3.3 Tahapan Evaluasi Waktu Komputasi

Tahapan ini bertujuan mengukur waktu komputasi dari tiga skenario: AES saja, RSA saja, dan *Hybrid* AES–RSA.

Langkah-Langkah Pengujian Waktu:

1. Menentukan variasi panjang pesan

   50, 75, 100, 150, dan 200 karakter.

2. Menjalankan tiga skenario pengujian
   a. AES saja (enkripsi-dekripsi AES)
   b. RSA saja (enkripsi-dekripsi RSA)
   c. *Hybrid* AES-RSA (AES + RSA untuk enkripsi, RSA + AES untuk dekripsi)
3. Melakukan pengulangan sebanyak 50 kali
4. Mengukur waktu enkripsi dan dekripsi menggunakan fungsi `perf_counter()`

5. Menghitung waktu rata-rata

   a. Enkripsi:

$$\bar{T}_{enc} = \frac{1}{N}\sum T_{enc}$$

   b. Dekripsi:

$$\bar{T}_{dec} = \frac{1}{N}\sum T_{dec}$$

6. Menyusun hasil pengujian dalam tabel

   Setiap skenario memiliki tabel waktu rata-rata untuk setiap panjang pesan.

7. Analisis perbandingan

   a. Skenario dengan kinerja tercepat.

   b. Skenario dengan kinerja terendah (paling lambat).

   c. Efisiensi skema *Hybrid* AES-RSA.

   d. Pengaruh panjang pesan terhadap waktu komputasi.

BAB IV
HASIL DAN PEMBAHASAN

4.1 Hasil Simulasi Proses Enkripsi

4.1.1 Pembangkitan Kunci AES

Pada simulasi penelitian ini, kunci yang diperoleh dinyatakan dalam bentuk string heksadesimal:

$$K_{AES} = d9adf7aab4e3f811ac7330e0e2c31bb8$$

Perlu ditegaskan bahwa perhitungan manual di bawah ini hanya contoh ilustratif untuk menunjukkan cara kerja perluasan kunci (*key expansion*). Dalam implementasi sebenarnya, seluruh proses ini dilakukan otomatis oleh pustaka kriptografi (misalnya *PyCryptodome*).

1. Penyusunan *Word* Awal dari Kunci 128-bit

   Kunci 128-bit terdiri atas 16 *byte*. Dalam bentuk heksadesimal:

   $$d9\ ad\ f7\ aa\ b4\ e3\ f8\ 11\ ac\ 73\ 30\ e0\ e2\ c3\ 1b\ b8$$

   Kunci ini kemudian dipecah menjadi empat *word* awal (w_0, w_1, w_2, w_3), masing-masing 4 *byte*:

   $$w_0 = d9\ ad\ f7\ aa$$

   $$w_1 = b4\ e3\ f8\ 11$$

   $$w_2 = ac\ 73\ 30\ e0$$

   $$w_3 = e2\ c3\ 1b\ b8$$

   Keempat *word* tersebut merupakan *round key* ke-0 yang digunakan pada operasi *AddRoundKey* sebelum putaran pertama enkripsi AES.

2. Contoh Perhitungan Manual *Key Expansion* (Menuju *Round key* Pertama)

   Untuk AES-128, *word* berikutnya w_4 dihitung dari w_0 dan w_3 dengan rumus:

$$w_4 = w_0 \oplus T(w_3),\ \ T(x) = SubWord\big(RotWord(x)\big) \oplus Rcon[1]$$

   a. *RotWord* pada w_3

$$w_3 = e2\ c3\ 1b\ b8$$

   RotWord menggeser *byte* ke kiri satu posisi:

$$RotWord(w_3) = c3\ 1b\ b8\ e2$$

   b. *SubWord* (menggunakan tabel S-box)

   Setiap *byte* diganti melalui S-box AES:

   i. $c3 \rightarrow 2e$

   ii. $1b \rightarrow af$

   iii. $b8 \rightarrow 6c$

   iv. $e2 \rightarrow 98$

   Sehingga:

$$SubWord\big(RotWord(w_3)\big) = 2e\ af\ 6c\ 98$$

   c. XOR dengan $Rcon[1] = 01\ 00\ 00\ 00$

   XOR per *byte*:

$$2e \oplus 01 = 2f$$

$$af \oplus 00 = af$$

$$6c \oplus 00 = 6c$$

$$98 \oplus 00 = 98$$

   Maka diperoleh:

$$T(w_3) = 2f\ af\ 6c\ 98$$

d. Menghitung $w_4 = w_0 \oplus T(w_3)$

$$w_0 = d9\ ad\ f7\ aa$$

XOR per *byte*:

$$d9 \oplus 2f = f6$$

$$ad \oplus af = 02$$

$$f7 \oplus 6c = 9b$$

$$aa \oplus 98 = 32$$

Sehingga:

$$w_4 = f6\ 02\ 9b\ 32$$

e. Menghitung $w_5 = w_1 \oplus w_4$

$$w_1 = b4\ e3\ f8\ 11,\ \ w_4 = f6\ 02\ 9b\ 32$$

XOR per *byte*:

$$b4 \oplus f6 = 42$$

$$e3 \oplus 02 = e1$$

$$f8 \oplus 9b = 63$$

$$11 \oplus 32 = 23$$

Sehingga:

$$w_5 = 42\ e1\ 63\ 23$$

f. Menghitung $w_6 = w_2 \oplus w_5$

$$w_2 = ac\ 73\ 30\ e0,\ \ w_5 = 42\ e1\ 63\ 23$$

XOR per *byte*:

$$ac \oplus 42 = ee$$

$$73 \oplus e1 = 92$$

$$30 \oplus 63 = 53$$

$$e0 \oplus 23 = c3$$

Sehingga:

$$w_6 = ee\ 92\ 53\ c3$$

g. Menghitung $w_7 = w_3 \oplus w_6$

$$w_3 = e2\ c3\ 1b\ b8,\ \ w_6 = ee\ 92\ 53\ c3$$

XOR per *byte*:

$$e2 \oplus ee = 0c$$

$$c3 \oplus 92 = 51$$

$$1b \oplus 53 = 48$$

$$b8 \oplus c3 = 7b$$

Sehingga:

$$w_7 = 0c\ 51\ 48\ 7b$$

Berdasarkan hasil perhitungan manual *key expansion* AES-128 diperoleh empat *word* baru yaitu w_4, w_5, w_6, dan w_7 yang membentuk *Round Key* 1 sepanjang 128 bit. Secara linear, *round key* tersebut dituliskan sebagai:

$$Round\ Key\ 1 = f6\ 02\ 9b\ 32\ 42\ e1\ 63\ 23\ ee\ 92\ 53\ c3\ 0c\ 51\ 48\ 7b$$

Penulisan ini merupakan representasi deret *byte* hasil langsung proses ekspansi kunci. Namun dalam algoritma AES, kunci putaran digunakan dalam bentuk matriks 4×4 karena harus menyesuaikan struktur *state matrix* pada proses enkripsi, khususnya tahap *AddRoundKey*. Setiap *word* menempati satu kolom matriks, sehingga *Round Key* 1 dapat disusun menjadi:

$$\begin{bmatrix} f6 & 42 & ee & 0c \\ 02 & e1 & 92 & 51 \\ 9b & 63 & 53 & 48 \\ 32 & 23 & c3 & 7b \end{bmatrix}$$

Dengan demikian, bentuk deret *byte* dan matriks merepresentasikan nilai *round key* yang sama. Perbedaannya hanya pada format penyajian: bentuk linear untuk perhitungan *key expansion*, sedangkan bentuk matriks untuk implementasi proses enkripsi AES.

3. *State* Matrix 4 × 4

   a. Matrix Kunci Awal (*Round key* 0)

$$\begin{bmatrix} d9 & ad & f7 & aa \\ b4 & e3 & f8 & 11 \\ ac & 73 & 30 & e0 \\ e2 & c3 & 1b & b8 \end{bmatrix}$$

   b. Matrix *Round key* Pertama $(w_4 - w_7)$

$$\begin{bmatrix} f6 & 42 & ee & 0c \\ 02 & e1 & 92 & 51 \\ 9b & 63 & 53 & 48 \\ 32 & 23 & c3 & 7b \end{bmatrix}$$

Proses *key expansion* yang ditunjukkan pada contoh di atas hanya menggambarkan pembentukan *round key* pertama yang berasal dari empat *word* awal (w_0, w_1, w_2, w_3). Pada algoritma AES-128, proses perluasan kunci tidak berhenti sampai *word* w_7, tetapi berlanjut hingga menghasilkan total 44 *word*, yang kemudian disusun menjadi 11 *round key* lengkap, yaitu:

1. 1 *round key* awal (*Round key* 0)
2. 10 *round key* berikutnya (*Round key* 1 sampai *Round key* 10)

Setiap *round key* terdiri atas empat *word* $(4 \times 4\ byte = 16\ byte)$.

Dengan demikian, *key expansion* terus menerus membentuk *word* baru:

$$w_8, w_9, w_{10}, \dots, w_{43}$$

mengikuti pola perhitungan yang sama seperti pada ilustrasi:

1. Setiap 4 *word* akan membentuk 1 *round key*.

2. Setiap *word* indeks kelipatan 4 menerapkan fungsi khusus $T(x)$ (Rot*Word* + Sub*Word* + Rcon).
3. *Word* lainnya dihitung dengan operasi XOR sederhana.

4.1.2 Enkripsi Menggunakan AES Tunggal

AES digunakan untuk mengenkripsi pesan secara simetris. Tabel 4.1, menunjukkan hasil enkripsi beberapa sampel pesan.

Tabel 4.1 Hasil Enkripsi AES Tunggal

No	Panjang Pesan	*Plaintext*	*Ciphertext* (hex)	*Initialization Vector* (IV)	Panjang *Output*
1	50 karakter	Pesan ini aman dan TERENKRIPSI dengan sangat baik.	ca08b0626cb72cbeafa8ee191d6e6f9bfb17124e73e243c33ff7d6f2dc6023de4c027c1110fccc3dc8f5465c8225801e2f3f5766d42b402a6194d6e28435cb23	77cef22d61e56be28ea958ef41f77b7c	64 *byte*
2	75 karakter	*Hybrid* AES dan RSA mengamankan pesan teks digital secara efisien aman baik	f75add990296417005611d3b61487858a5f1fddf953cb4bdbb2c4f76bb97fed28e10a3bdd1d61e08318e144e102c1341865559ac91366082784c8263f7a6831189f864eb	5aac7f1606d6b465d963eb4333f6ba97	80 *byte*

No	Panjang Pesan	*Plaintext*	*Ciphertext* (hex)	*Initialization Vector* (IV)	Panjang *Output*
			c037f1e43be7ad3b1329e0bd		
3	100 karakter	Keamanan pesan teks penting agar data tetap aman, dan sistem enkripsi membantu menjaga privasi anda.	636708ffaae6d0d3192d462c4aa488b74429e44676d25de58b29e1628f79889ce4e3b23d6889682de837cbba94705252eb68187ffb05e43f7515bdaaf8074d077b867782c5b8de206a0aa8410f1129155e622176060eb02cc0d7b47aa5133edab62ebd2e7b95f294bc89788d8c239a77	1c00b3c23a927afd8fbfd7933d97cb9e	112 *byte*
4	150 karakter	Sistem *Hybrid* AES dan RSA melindungi pesan teks digital secara aman dan efisien dengan menggabungkan kecepatan AES serta keamanan	0f79b98c22413fb8f2b04acca8d0d6b136cdac59c39d59b17b87908145aad991ef94d69691e2fe0310d80c3c50e3c26c02ee1f2d16c9b021643360df6c96b4add925696b8f49a80507180225	1b09d9f3afccdd55089c60c6d42f7f3e	176 *byte*

No	Panjang Pesan	*Plaintext*	*Ciphertext* (hex)	*Initialization Vector* (IV)	Panjang *Output*
		distribusi kunci RSA	ecc1a7a3f5049f8817f9058c742a1e47d98f0b66be5a6aedd063fec8a4e0efba7de8f68cc202e2846c45eb23f6f84505c73849b0c36dae8b1a89f252a05390daeada2bb923e1c1b20b67a3fe3b57485834ae8e70		
5	200 karakter	*Hybrid* AES dan RSA meningkatkan keamanan pesan teks dengan memadukan kecepatan enkripsi AES dan kekuatan RSA sehingga distribusi kunci tetap aman serta proses komunikasi berlangsung lebih terlindungi.	b08e32ccf7feb44033e60a56531593d8d21f5a1d5b674bc934c1f040f602f5c54e056b0055c1db6f26d777dd4621d8b8d39a69813dcef95f4b14f814a5239f77ce916938a91d4b9a4df1952d92d16e85dadc240d9181cfdf82d098fc5bdb7bb7e1bdec2febeaf8c9294bd8f2cce7e4c7d131d1ae27	01c3d16e205c322e80a62df259e5b2fe	208 *byte*

No	Panjang Pesan	*Plaintext*	*Ciphertext* (hex)	*Initialization Vector* (IV)	Panjang *Output*
			9bd8811fd74b3 4b56d4db1eb16 657c00583b1d6 84b220b9eeb12 10784f4f627b70 025acf95db83fe 059c458f44bbd5 1c3cad27915cb6 d3290295368e6 8e40b113a8b3c 063bc828a9a2e3 0e204128b36b4 e22724597ff480 3b6f7f4		

Berdasarkan Tabel 4.1, panjang *ciphertext* AES tunggal meningkat seiring bertambahnya panjang pesan, yaitu dari 64 *byte* hingga 208 *byte*. Hal ini disebabkan oleh mekanisme pemrosesan blok AES berukuran tetap 128 bit. Tabel juga menunjukkan bahwa setiap proses enkripsi menghasilkan *Initialization Vector* (IV) yang berbeda, sehingga *ciphertext* yang dihasilkan bersifat unik pada setiap sesi.

4.1.3 Pembangkitan Kunci RSA

Pembangkitan kunci RSA pada penelitian ini dilakukan secara otomatis oleh sistem menggunakan pustaka kriptografi pada bahasa pemrograman *Python*. Proses ini menghasilkan sepasang kunci RSA, yaitu kunci publik dan kunci privat, yang ditampilkan langsung pada antarmuka aplikasi sebagai keluaran program. Penambahan gambar hasil program pada subbab ini bertujuan untuk menunjukkan

bahwa pembangkitan kunci dilakukan secara nyata dan bukan sekadar konsep teoretis.

Secara matematis, pembangkitan kunci RSA mengikuti tahapan standar algoritma RSA. Sistem terlebih dahulu memilih dua bilangan prima besar yang berbeda, kemudian menghitung nilai modulus $n = p \times q$. Selanjutnya dihitung fungsi totien Euler $\varphi(n) = (p-1)(q-1)$. Dari nilai tersebut, sistem memilih eksponen publik e yang relatif prima terhadap $\varphi(n)$, kemudian menghitung eksponen privat d sebagai *Inv*ers modulo dari $e$terhadap $\varphi(n)$, sehingga memenuhi hubungan:

$$d \equiv e^{-1} \left(mod\ \varphi(n)\right)$$

Hasil akhir dari proses ini adalah pasangan kunci RSA yang dinyatakan sebagai:

1. Kunci Publik (e, n), yang digunakan untuk mengenkripsi kunci AES.
2. Kunci Privat (d, n), yang digunakan untuk mendekripsi kembali kunci AES pada sisi penerima.

Gambar 4.1 Tampilan Kunci Publik pada Program

Gambar 4.2 Tampilan Kunci Privat pada Program

Berdasarkan keluaran program yang ditampilkan pada Gambar 4.1 dan Gambar 4.2, terlihat bahwa sistem berhasil membangkitkan pasangan kunci RSA dengan panjang kunci 2048 bit. Kunci publik digunakan pada proses enkripsi kunci AES, sedangkan kunci privat disimpan secara aman dan hanya digunakan pada proses dekripsi. Pemisahan peran ini menunjukkan karakteristik kriptografi asimetris RSA, di mana keamanan sistem tidak bergantung pada kerahasiaan kunci publik, melainkan pada kunci privat.

4.1.4 Enkripsi Menggunakan RSA Tunggal

RSA diuji untuk melihat karakteristik *output ciphertext* ketika mengenkripsi nilai pendek.

Tabel 4.2 Hasil Enkripsi RSA Tunggal

No	*Plaintext*	*Ciphertext* (hex)	Panjang *Output*
1	Pesan ini aman dan TERENKRIPSI dengan sangat baik.	cf0ac4bae7ff5aa731f459a241d647c9f9d a4b00fb93dd437d277ff04bc06443232b 7144f5e41f39e45c068dee1b5f5fbfbc81 8e09ca3f9dfc9f337fb70f1bb0afb13ffeb 6d2e27ac3ef3a883593b993c5e2aba7abf ed741347f011b4a4d757507b3e8023afc 1347eee37c6badbbc2da325c74b89dfd8 cbf02a188c6985745cf67353f2e014e259 fb99944d3c947dad9c9ac986f40cb9139 a3311828bb3bf19bcee36feaa9ac0e9f83 44b4eceed1b62b1d4262fd9408b701e0f 186515668472a8809f8d6a28d7f9bfabb e9a5f804add9b154f7e195005c9d66877 065634e407bfd888cbfa197931b137f8fd f8893bb68b5d18fbdd516a0544acd136d 647ddcef	512 *byte*
2	*Hybrid* AES dan RSA mengamankan pesan teks digital secara efisien aman baik	8f0418cdecb7b17af19fd7fd28cdaf192d 60b0be1e3ec2a4ef11e602f7ef6fd46969 5f9223b6620d4a09e1385d3fd1fea7618 bfef51dab6657e3a53e96548f48ff7d1c4 3d2086b6019d0d57ea8a038b8c48cf010 b31cd518039fe4dd6427b407aab8b758c 98693f6f4252e710195c8954165f2854d 1a05a6afc1ff9f7730536e1a810fdd8318 ad5909461253972808184 5c00c18ffb38 1aecb5c8776d31a16f8a01736fde7160a5 a1586d2a1f9732278aaf0312a6406b392 e035894d8df76a47d5267e9acb82bb0c7 cd87d07d7005994ce98326ad9887ef9cb aaf663b4f490c0d87fb4e13fe06a390b9c	512 *byte*

No	*Plaintext*	*Ciphertext* (hex)	Panjang *Output*
		6f0c35f31f9f1dbc9e0cf7b874f21b3ad5d578a57f90	
3	Keamanan pesan teks penting agar data tetap aman, dan sistem enkripsi membantu menjaga privasi anda.	25d410038619e5b006597b808c4eed54b4e0bc139585bbf04cfc7572469bd943b6aa96c5e07202be017b45b5512a92ca430cf2ff4577afcdfcf935554c0d9b99d43fe13514cd8bc4b971ec09e664361bc99bb6f4a868b88a78c41cd7a9209c26613a3e50ac273013fa0e12304f8d5fc1afdd1aedb880b7ea61d745aceea737d98eda0a2487e71c6a9ab31250886640d63c97b8ff884e09dc5a12333fcef36e09135a7bd152c46716858f3c82e260ddb46196513f74f0de23448fd6478df0d887401ee34f71a564bbd139f6a879e19a03e27803e51c9bf0c7e12ae6ccfbfa1c74b9f9d3ea3c0650b78b2429279d9d60adb303e0ec53b9b3a05d1fa3c4ff08af82	512 *byte*
4	Sistem *Hybrid* AES dan RSA melindungi pesan teks digital secara aman dan efisien dengan menggabungkan kecepatan AES serta keamanan distribusi kunci RSA	bc64d767e15f241dc194942abd2c60a987dd8f417d239545ef4351d597f75da430307e0565f4fc31df182d7cb96207eb93e6a4c8fce8b58f298d944ceca6a5c1adc74a85c03797ec7a6e054bd53b87b804d80788164160e2e03a1e46facbc07262b4e954623b210885c55bcd4e9fa27b409d5647bd9c00e08743d2a756c6f3b00e1bf3195377c0516847bb2874263928127e08dfb296601f1c7b641baf0d49180d06500eea56786200e7e4999b1a28c48a9864347e7f1f7e6138362ed44e4bc4a97a2f034ddd193573ec94c8aa5b005c0f35c5318e7618e	512 *byte*

No	*Plaintext*	*Ciphertext* (hex)	Panjang *Output*
		ae96f557d86608987324f9acd3d5d7b93 f4fe7aec9b8db5496afba7a0a77083ea6d 52fa43b8e88d00	
5	*Hybrid* AES dan RSA meningkatkan keamanan pesan teks dengan memadukan kecepatan enkripsi AES dan kekuatan RSA sehingga distribusi kunci tetap aman serta proses komunikasi berlangsung lebih terlindungi.	Tidak Berhasil Dienkripsi (melebihi batas ukuran *input* RSA)	-

Berdasarkan Tabel 4.2, seluruh *plaintext* yang dienkripsi menggunakan RSA menghasilkan *ciphertext* dengan panjang *output* yang sama, yaitu 512 *byte*. Proses enkripsi pada tahap ini dilakukan menggunakan kunci publik RSA (e, n), di mana e merupakan eksponen publik dan n adalah modulus hasil perkalian dua bilangan prima besar. Hal ini menunjukkan bahwa ukuran *ciphertext* RSA ditentukan oleh panjang modulus (n) pada pasangan kunci RSA yang digunakan, bukan oleh panjang *plaintext* yang dienkripsi. Karena nilai modulus n sama pada kunci publik maupun kunci privat, maka panjang *ciphertext* akan tetap selama ukuran kunci tidak berubah.

Namun, pada *plaintext* dengan panjang yang lebih besar (±200 karakter), proses enkripsi tidak berhasil dilakukan. Hal ini disebabkan oleh keterbatasan

RSA yang hanya dapat mengenkripsi data dengan ukuran lebih kecil dari modulus n setelah mempertimbangkan *padding*.

4.1.5 Enkripsi *Hybrid* AES-RSA dengan Perhitungan Manual

Berikut contoh langkah-langkah perhitungan manual *Hybrid* AES-RSA:

1. Pesan teks asli (*plaintext*) yang akan dienkripsi

   Pada tahap awal, disiapkan teks asli (*plaintext*) yang akan digunakan dalam proses perhitungan enkripsi manual. Teks diambil dari sebuah kalimat pesan yaitu: “Pesan ini aman dan terenkripsi dengan sangat baik.”. Namun, karena proses perhitungan dilakukan secara manual untuk mempermudah analisis, maka tidak seluruh kalimat digunakan. Penelitian ini hanya mengambil satu kata sebagai sampel perhitungan, yaitu: TERENKRIPSI.

2. Menambahkan *padding* agar sesuai dengan ukuran blok

   Algoritma AES-128 bekerja dengan ukuran blok tetap sebesar 16 *byte*. Oleh karena itu, panjang *plaintext* harus merupakan kelipatan 16 *byte* agar dapat diproses dalam satu blok enkripsi secara utuh. Jika jumlah karakter tidak memenuhi ukuran tersebut, maka diperlukan penambahan *padding.* Pada penelitian ini, *plaintext* yang digunakan dalam perhitungan manual adalah: TERENKRIPSI. Kata tersebut terdiri dari 11 karakter, sehingga memiliki panjang 11 *byte* dalam representasi ASCII. Karena belum memenuhi 16 *byte*, maka perlu dilakukan penambahan *padding* sebanyak 5 *byte* (16 − 11 = 5). Metode *padding* yang digunakan adalah PKCS#7, di mana setiap *byte* tambahan diisi dengan nilai heksadesimal yang menunjukkan jumlah *byte padding* yang ditambahkan.

   Dengan demikian, karena jumlah *padding* adalah 5 *byte*, maka nilai setiap

byte padding adalah 05. Setelah penambahan *padding*, blok *plaintext* menjadi:

TERENKRIPSI + 05 05 05 05 05

Jika dikonversikan ke dalam bentuk heksadesimal ASCII, maka diperoleh:

54 45 52 45 4E 4B 52 49 50 53 49 05 05 05 05 05

Dengan penambahan tersebut, panjang data kini telah menjadi 16 *byte* sehingga memenuhi ukuran blok yang dipersyaratkan dan dapat diproses pada tahap enkripsi AES-128 selanjutnya.

3. Pemisahan *plaintext* ke dalam blok 16 *byte*

   Setelah dilakukan proses *padding* menggunakan metode PKCS#7, panjang data menjadi 16 *byte* sehingga hanya membentuk satu blok enkripsi dan tidak perlu dilakukan pembagian ke blok lain. Susunan blok *plaintext* adalah sebagai berikut:

   Blok 1: TERENKRIPSI + *padding*

   Setiap karakter pada blok kemudian dikonversikan ke dalam kode ASCII dan direpresentasikan dalam bentuk heksadesimal. Hasil konversinya ditunjukkan sebagai berikut:

   Hex: [54 45 52 45 4E 4B 52 49 50 53 49 05 05 05 05 05]

   Blok heksadesimal tersebut kemudian disusun ke dalam matriks *state* 4×4 yang akan digunakan pada tahapan proses enkripsi selanjutnya.

$$\begin{bmatrix} 54 & 4E & 50 & 05 \\ 45 & 4B & 53 & 05 \\ 52 & 52 & 49 & 05 \\ 45 & 49 & 05 & 05 \end{bmatrix}$$

   Matriks *state* ini selanjutnya akan diproses pada tahap awal enkripsi AES-128, yaitu *AddRoundKey* sebelum masuk ke ronde transformasi berikutnya.

4. Kunci yang digunakan dalam proses enkripsi

   Kunci yang digunakan dalam proses enkripsi AES-128 pada penelitian ini adalah:

   KUNCIPERHITUNGAN

   Panjang kunci tersebut berjumlah 16 karakter (16 *byte*) sehingga telah memenuhi kebutuhan panjang kunci AES-128 dan dapat langsung digunakan sebagai *Round Key* 0 tanpa proses penyesuaian. Setiap karakter kunci kemudian dikonversikan ke dalam kode ASCII dan direpresentasikan dalam bentuk heksadesimal sebagai berikut:

   Hex: [4B 55 4E 43 49 50 45 52 48 49 54 55 4E 47 41 4E]

   Sama seperti *plaintext*, kunci utama (*Round Key* 0) disusun ke dalam matriks *state* 4×4 berdasarkan pengisian kolom AES sebagai berikut:

$$\begin{bmatrix} 4B & 49 & 48 & 4E \\ 55 & 50 & 49 & 47 \\ 4E & 45 & 54 & 41 \\ 43 & 52 & 55 & 4E \end{bmatrix}$$

   Matriks tersebut merupakan *Round Key* 0 yang akan digunakan pada proses awal enkripsi, yaitu tahap *AddRoundKey*.

   Untuk memperoleh kunci pada setiap ronde enkripsi, dilakukan proses *Key Expansion*. Kunci utama dibagi menjadi empat *word* awal, yaitu $W[0]$ hingga $W[3]$. Pembentukan *word* selanjutnya dilakukan melalui beberapa tahapan, yaitu: Rot*Word*, Sub*Word*, XOR dengan Rcon, XOR dengan $W[i-4]$. *Word* berikutnya dihitung menggunakan operasi:

$$W[i] = W[i-4] \oplus W[i-1]$$

Proses ini diulang hingga terbentuk 44 *word* ($W[0] - W[43]$), yang kemudian dikelompokkan menjadi 11 *Round Key* (*Round Key* 0 sampai *Round Key* 10).

5. Proses *plaintext* menjadi blok AES

   Setelah plainteks melalui proses *padding* sehingga berjumlah 16 *byte*, langkah selanjutnya adalah menyusunnya ke dalam matriks berukuran 4 × 4 yang disebut sebagai *State*. Matriks ini merupakan struktur dasar yang digunakan dalam setiap tahapan transformasi algoritma AES-128.

   Blok plainteks dalam representasi heksadesimal adalah:

   $$P = [54\ 45\ 52\ 45\ 4E\ 4B\ 52\ 49\ 50\ 53\ 49\ 05\ 05\ 05\ 05\ 05]$$

   Dengan ketentuan *column-major order*, *byte* dimasukkan ke dalam matriks secara kolom per kolom. Empat *byte* pertama mengisi kolom pertama, empat *byte* berikutnya mengisi kolom kedua, dan seterusnya hingga seluruh 16 *byte* tersusun. Jika dinotasikan sebagai:

   $$S = [S_{0,0}, S_{1,0}, S_{2,0}, S_{3,0}, S_{0,1}, S_{1,1}, S_{2,1}, S_{3,1}, S_{0,2}, S_{1,2}, S_{2,2}, S_{3,2}, S_{0,3},$$
   $$S_{1,3}, S_{2,3}, S_{3,3}]$$

   Maka matriks *State* (S) direpresentasikan dalam bentuk matriks sebagaimana pada Persamaan (2.1):

   $$S = \begin{bmatrix} S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\ S_{1,0} & S_{1.1} & S_{1,2} & S_{1,3} \\ S_{2,0} & S_{2,1} & S_{2,2} & S_{2,3} \\ S_{3,0} & S_{3,1} & S_{3,2} & S_{3,3} \end{bmatrix}$$

   Berdasarkan blok heksadesimal plainteks yang digunakan, maka diperoleh matriks *State* sebagai berikut:

$$S = \begin{bmatrix} 54 & 4E & 50 & 05 \\ 45 & 4B & 53 & 05 \\ 52 & 52 & 49 & 05 \\ 45 & 49 & 05 & 05 \end{bmatrix}$$

Matriks tersebut yang siap diproses pada tahapan enkripsi AES-128. Selanjutnya, matriks *State* ini akan digunakan dalam tahap awal transformasi, yaitu *AddRoundKey*, sebelum masuk ke proses *SubBytes*, *ShiftRows*, dan *MixColumns* pada ronde pertama.

6. Proses enkripsi AES-128 dengan perhitungan manual pada setiap blok:

   a. Langkah 1: *AddRoundKey*

      Matrix *state plaintext* (Blok 1):

$$S = \begin{bmatrix} 54 & 4E & 50 & 05 \\ 45 & 4B & 53 & 05 \\ 52 & 52 & 49 & 05 \\ 45 & 49 & 05 & 05 \end{bmatrix}$$

      Kunci yang digunakan:

$$K = \begin{bmatrix} 4B & 49 & 48 & 4E \\ 55 & 50 & 49 & 47 \\ 4E & 45 & 54 & 41 \\ 43 & 52 & 55 & 4E \end{bmatrix}$$

      Tahap *AddRoundKey* dilakukan dengan operasi XOR antara setiap elemen matriks *State* (S) dan matriks Kunci (K) pada posisi yang sama.

$$Hasil\ XOR = \begin{bmatrix} 54 \oplus 4B & 4E \oplus 49 & 50 \oplus 48 & 05 \oplus 4E \\ 45 \oplus 55 & 4B \oplus 50 & 53 \oplus 49 & 05 \oplus 47 \\ 52 \oplus 4E & 52 \oplus 45 & 49 \oplus 54 & 05 \oplus 41 \\ 45 \oplus 43 & 49 \oplus 52 & 05 \oplus 55 & 05 \oplus 4E \end{bmatrix}$$

      Perhitungan XOR manual:

      Baris 1:

$$54 \oplus 4B = 01010100 \oplus 01001011 = 00011111 = 1F$$

$$4E \oplus 49 = 01001110 \oplus 01001001 = 00000111 = 07$$

$$50 \oplus 48 = 01010000 \oplus 01001000 = 00011000 = 18$$

$$05 \oplus 4E = 00000101 \oplus 01001110 = 01001011 = 4B$$

Baris 2:

$$45 \oplus 55 = 01000101 \oplus 01010101 = 00010000 = 10$$

$$4B \oplus 50 = 01001011 \oplus 01010000 = 00011011 = 1B$$

$$53 \oplus 49 = 01010011 \oplus 01001001 = 00011010 = 1A$$

$$05 \oplus 47 = 00000101 \oplus 01000111 = 01000010 = 42$$

Baris 3:

$$52 \oplus 4E = 01010010 \oplus 01001110 = 00011100 = 1C$$

$$52 \oplus 45 = 01010010 \oplus 01000101 = 00010111 = 17$$

$$49 \oplus 54 = 01001001 \oplus 01010100 = 00011101 = 1D$$

$$05 \oplus 41 = 00000101 \oplus 01000001 = 01000100 = 44$$

Baris 4:

$$45 \oplus 43 = 01000101 \oplus 01000011 = 00000110 = 06$$

$$49 \oplus 52 = 01001001 \oplus 01010010 = 00011011 = 1B$$

$$05 \oplus 55 = 00000101 \oplus 01010101 = 01010000 = 50$$

$$05 \oplus 4E = 00000101 \oplus 01001110 = 01001011 = 4B$$

Setelah seluruh elemen dilakukan operasi XOR, diperoleh matriks *State* baru sebagai berikut:

$$S = \begin{bmatrix} 1F & 07 & 18 & 4B \\ 10 & 1B & 1A & 42 \\ 1C & 17 & 1D & 44 \\ 06 & 1B & 50 & 4B \end{bmatrix}$$

b. Langkah 2: *SubBytes*

Setiap *byte* berbentuk heksadesimal XY:

X = indeks baris pada S-Box, Y = indeks kolom pada S-Box. Nilai diganti dengan elemen pada perpotongan baris X dan kolom Y. S-Box yang digunakan adalah S-Box standar AES.

Baris 1:

$$1F \rightarrow \text{S-Box}\ (1,F) = C0$$

$$07 \rightarrow \text{S-Box}\ (0,7) = C5$$

$$18 \rightarrow \text{S-Box}\ (1,8) = AD$$

$$4B \rightarrow \text{S-Box}\ (4,B) = B3$$

Baris 2:

$$10 \rightarrow \text{S-Box}\ (1,0) = CA$$

$$1B \rightarrow \text{S-Box}\ (1,B) = AF$$

$$1A \rightarrow \text{S-Box}\ (1,A) = A2$$

$$42 \rightarrow \text{S-Box}\ (4,2) = 2C$$

Baris 3:

$$1C \rightarrow \text{S-Box}\ (1,C) = 9C$$

$$17 \rightarrow \text{S-Box}\ (1,7) = A4$$

$$1D \rightarrow \text{S-Box}\ (1,D) = A4$$

$$44 \rightarrow \text{S-Box}\ (4,4) = 1B$$

Baris 4:

$$06 \rightarrow \text{S-Box}\ (0,6) = 6F$$

$$1B \rightarrow \text{S-Box}\ (1,B) = AF$$

$$50 \rightarrow \text{S-Box}\ (5,0) = 53$$

$$4B \rightarrow \text{S-Box}\ (4,B) = B3$$

Hasil dari seluruh *SubBytes* adalah:

$$S = \begin{bmatrix} 1F & 07 & 18 & 4B \\ 10 & 1B & 1A & 42 \\ 1C & 17 & 1D & 44 \\ 06 & 1B & 50 & 4B \end{bmatrix} \Rightarrow S_1 = \begin{bmatrix} C0 & C5 & AD & B3 \\ CA & AF & A2 & 2C \\ 9C & F0 & A4 & 1B \\ 6F & AF & 53 & B3 \end{bmatrix}$$

c. Langkah 3: *ShiftRows*

Menggunakan matrix hasil dari *SubBytes*:

Baris 0: $[C0, C5, AD, B3]$, tidak digeser ke kiri.

Baris 1: $[CA, AF, A2, 2C]$, digeser satu *byte* ke kiri menjadi $[AF, A2, 2C, CA]$.

Baris 2: $[9C, F0, A4, 1B]$, digeser dua *byte* ke kiri menjadi $[A4, 1B, 9C, F0]$.

Baris 3: $[6F, AF, 53, B3]$, digeser tiga *byte* ke kiri menjadi $[B3, 6F, AF, 53]$.

Sehingga hasilnya adalah:

$$S_1 = \begin{bmatrix} C0 & C5 & AD & B3 \\ CA & AF & A2 & 2C \\ 9C & F0 & A4 & 1B \\ 6F & AF & 53 & B3 \end{bmatrix} \Rightarrow S_2 = \begin{bmatrix} C0 & C5 & AD & B3 \\ AF & A2 & 2C & CA \\ A4 & 1B & 9C & F0 \\ B3 & 6F & AF & 53 \end{bmatrix}$$

d. Langkah 4: *MixColumns*

$$S_2 = \begin{bmatrix} C0 & C5 & AD & B3 \\ AF & A2 & 2C & CA \\ A4 & 1B & 9C & F0 \\ B3 & 6F & AF & 53 \end{bmatrix}$$

Operasi *MixColumns* terhadap kolom pertama:

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} C0 \\ AF \\ A4 \\ B3 \end{bmatrix} = \begin{bmatrix} 66 \\ C1 \\ F2 \\ 2D \end{bmatrix}$$

i. Perhitungan elemen pertama (baris 1 kolom 1)

$(02 \cdot C0) \oplus (03 \cdot AF) \oplus (01 \cdot A4) \oplus (01 \cdot B3)$

i) Menghitung $02 \times C0$:

$C0 = 11000000_2$

Geser ke kiri 1 bit → 10000000

Karena bit awal = 1 → XOR $1B$:

$10000000 \oplus 00011011$

$= 10011011$

$= 9B$ (Hex)

ii) Menghitung $03 \times AF$:

$03 \cdot x = (02 \cdot x) \oplus x$

Hitung $02 \times AF$

$AF = 10101111_2$

Geser kiri → 01011110

Karena bit awal 1 → XOR $1B$:

$01011110 \oplus 00011011$

$= 01000101$

$= 45$

Maka $03 \times AF = 45 \oplus AF$

$= 01000101 \oplus 10101111$

$= 11101010$

$= EA$ (Hex)

iii) Menghitung $01 \times A4 = A4$ (Hex)

iv) Menghitung $01 \times B3 = B3$ (Hex)

XOR keseluruhan:

$S_{2(1,1)} = 9B \oplus EA \oplus A4 \oplus B3$

$= 10011011 \oplus 11101010 \oplus 11010101 \oplus 10110011$

$= 01100110$ (Biner)

$= 66$ (Hex)

ii. Perhitungan elemen kedua (baris 2 kolom 1)

$(01 \cdot C0) \oplus (02 \cdot AF) \oplus (03 \cdot A4) \oplus (01 \cdot B3)$

i) Menghitung $01 \times C0 = C0$

ii) Menghitung $02 \times AF$:

$01011110 \oplus 00011011 = 01000101 = 45$ (Hex)

iii) Menghitung $03 \times A4$:

$02 \times A4 = 0100100 \oplus 00011011 = 01010011 = 53$ (Hex)

$53 \oplus A4 = 01010011 \oplus 10100100 = 11110111 = F7$ (Hex)

iv) Menghitung $01 \times B3 = B3$

XOR keseluruhan:

$S_{2(2,1)} = C0 \oplus 45 \oplus F7 \oplus B3$

$= 11000000 \oplus 01000101 \oplus 11110111 \oplus 10110011$

$= 11000001$ (Biner)

$= C1$ (Hex)

iii. Perhitungan elemen ketiga (baris 3 kolom 1)

$(01 \cdot C0) \oplus (01 \cdot AF) \oplus (02 \cdot A4) \oplus (03 \cdot B3)$

i) Menghitung $01 \times C0 = C0$

ii) Menghitung $01 \times AF = AF$

iii) Menghitung $02 \times A4 = 01001000 \oplus 00011011 = 01010011$

$= 53$ (Hex)

iv) Menghitung $03 \times B3$:

$02 \times B3 = 01100110 \oplus 00011011 = 01111101 = 7D$ (Hex)

$7D \oplus B3 = 01111101 \oplus 10110011 = 11001110 = CE$ (Hex)

XOR keseluruhan:

$S_{2(3,1)} = C0 \oplus AF \oplus 53 \oplus CE$

$= 11000000 \oplus 10101111 \oplus 01010011 \oplus 11001110$

$= 11110010$ (Biner)

$= F2$ (Hex)

iv. Perhitungan elemen keempat (baris 4 kolom 1)

$(03 \cdot C0) \oplus (01 \cdot AF) \oplus (01 \cdot A4) \oplus (02 \cdot B3)$

i) Menghitung $03 \times C0$

$02 \times C0 = 10000000 \oplus 00011011 = 10011011 = 9B$ (Hex)

$9B \oplus C0 = 10011011 \oplus 11000000 = 01011011 = 5B$ (Hex)

ii) Menghitung $01 \times AF = AF$

iii) Menghitung $01 \times A4 = A4$

iv) Menghitung $02 \times B3 = 01100110 \oplus 00011011 = 01111101$

$= 7D$ (Hex)

XOR keseluruhan:

$S_{2(4,1)} = 5B \oplus AF \oplus A4 \oplus 7D$

$= 01011011 \oplus 10101111 \oplus 10100100 \oplus 01111101$

$= 00101101$ (Biner)

$= 2D$ (Hex)

Proses perhitungan *MixColumns* pada kolom kedua, ketiga, dan keempat dilakukan dengan langkah yang sama seperti pada kolom pertama.

Operasi *MixColumns* terhadap kolom kedua:

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} C5 \\ A2 \\ 1B \\ 6F \end{bmatrix} = \begin{bmatrix} A7 \\ E9 \\ D3 \\ A3 \end{bmatrix}$$

i. Elemen (baris 1, kolom 2):

$$(02 \cdot C5) \oplus (03 \cdot A2) \oplus (01 \cdot 1B) \oplus (01 \cdot 6F) = A7$$

ii. Elemen (baris 2, kolom 2):

$$(01 \cdot C5) \oplus (02 \cdot A2) \oplus (03 \cdot 1B) \oplus (01 \cdot 6F) = \mathrm{E9}$$

iii. Elemen (baris 3, kolom 2):

$$(01 \cdot C5) \oplus (01 \cdot A2) \oplus (02 \cdot 1B) \oplus (03 \cdot 6F) = \mathrm{D3}$$

iv. Elemen (baris 4, kolom 2):

$$(03 \cdot C5) \oplus (01 \cdot A2) \oplus (01 \cdot 1B) \oplus (02 \cdot 6F) = \mathrm{A3}$$

Operasi *MixColumns* terhadap kolom ketiga:

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} AD \\ 2C \\ 9C \\ AF \end{bmatrix} = \begin{bmatrix} 4C \\ F8 \\ 67 \\ 6C \end{bmatrix}$$

i. Elemen (baris 1, kolom 2):

$$(02 \cdot AD) \oplus (03 \cdot 2C) \oplus (01 \cdot 9C) \oplus (01 \cdot AF) = 4C$$

ii. Elemen (baris 2, kolom 2):

$$(01 \cdot AD) \oplus (02 \cdot 2C) \oplus (03 \cdot 9C) \oplus (01 \cdot AF) = F8$$

iii. Elemen (baris 3, kolom 2):

$$(01 \cdot AD) \oplus (01 \cdot 2C) \oplus (02 \cdot 9C) \oplus (03 \cdot AF) = 67$$

iv. Elemen (baris 4, kolom 2):

$$(03 \cdot AD) \oplus (01 \cdot 2C) \oplus (01 \cdot 9C) \oplus (02 \cdot AF) = 6C$$

Operasi *MixColumns* terhadap kolom keempat:

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} B3 \\ CA \\ F0 \\ 53 \end{bmatrix} = \begin{bmatrix} 28 \\ 66 \\ 9D \\ BC \end{bmatrix}$$

i. Elemen (baris 1, kolom 2):

$$(02 \cdot B3) \oplus (03 \cdot CA) \oplus (01 \cdot F0) \oplus (01 \cdot 53) = 28$$

ii. Elemen (baris 2, kolom 2):

$$(01 \cdot B3) \oplus (02 \cdot CA) \oplus (03 \cdot F0) \oplus (01 \cdot 53) = 66$$

iii. Elemen (baris 3, kolom 2):

$$(01 \cdot B3) \oplus (01 \cdot CA) \oplus (02 \cdot F0) \oplus (03 \cdot 53) = 9D$$

iv. Elemen (baris 4, kolom 2):

$$(03 \cdot B3) \oplus (01 \cdot CA) \oplus (01 \cdot F0) \oplus (02 \cdot 53) = BC$$

Hasil akhir transformasi *MixColumns* secara keseluruhan adalah:

$$S_3 = \begin{bmatrix} 66 & A7 & 4C & 28 \\ C1 & E9 & F8 & 66 \\ F2 & D3 & 67 & 9D \\ 2D & A3 & 6C & BC \end{bmatrix}$$

e. Langkah 5: *AddRoundKey*

Menggunakan sebuah *round key*:

$$K_1 = \begin{bmatrix} EA & A3 & EB & A5 \\ D6 & 86 & CF & 88 \\ 61 & 24 & 70 & 31 \\ 6C & 3E & 6B & 25 \end{bmatrix}$$

dan hasil dari *MixColumns* sebelumnya adalah:

$$S_3 = \begin{bmatrix} 66 & A7 & 4C & 28 \\ C1 & E9 & F8 & 66 \\ F2 & D3 & 67 & 9D \\ 2D & A3 & 6C & BC \end{bmatrix}$$

Maka *AddRoundKey* menghasilkan:

$$\begin{bmatrix} 66 & A7 & 4C & 28 \\ C1 & E9 & F8 & 66 \\ F2 & D3 & 67 & 9D \\ 2D & A3 & 6C & BC \end{bmatrix} \oplus \begin{bmatrix} EA & A3 & EB & A5 \\ D6 & 86 & CF & 88 \\ 61 & 24 & 70 & 31 \\ 6C & 3E & 6B & 25 \end{bmatrix}$$

$$= \begin{bmatrix} 8C & 04 & A7 & 8D \\ 17 & 6F & 37 & EE \\ 93 & F7 & 17 & AC \\ 41 & 9D & 07 & 99 \end{bmatrix}$$

Maka hasil *AddRoundKey* adalah:

$$S_4 = \begin{bmatrix} 8C & 04 & A7 & 8D \\ 17 & 6F & 37 & EE \\ 93 & F7 & 17 & AC \\ 41 & 9D & 07 & 99 \end{bmatrix}$$

f. Langkah 6: Ulangi Proses sampai dengan 9 *round* untuk AES-128

Putaran pertama: $S_4 = P_1 = \begin{bmatrix} 8C & 04 & A7 & 8D \\ 17 & 6F & 37 & EE \\ 93 & F7 & 17 & AC \\ 41 & 9D & 07 & 99 \end{bmatrix}$

Putaran kedua: $P_2 = \begin{bmatrix} AB & 51 & AC & 7B \\ 88 & 83 & 90 & D5 \\ 64 & FF & 25 & 68 \\ 62 & 06 & CB & 85 \end{bmatrix}$

Putaran ketiga: $P_3 = \begin{bmatrix} 6E & 3C & 6F & 5B \\ D4 & 91 & 7A & 13 \\ 3A & 0F & B6 & 9C \\ BC & 3D & 4E & 62 \end{bmatrix}$

Putaran keempat: $P_4 = \begin{bmatrix} C7 & 5A & 4D & 23 \\ 2E & 84 & 63 & 9F \\ 6D & 15 & 90 & 2B \\ F1 & 7C & A8 & 55 \end{bmatrix}$

Putaran kelima: $P_5 = \begin{bmatrix} 9A & 31 & 88 & 7E \\ 44 & F0 & 52 & 61 \\ B7 & 0D & 2F & 4C \\ 5C & A1 & 93 & 8B \end{bmatrix}$

Putaran keenam: $P_6 = \begin{bmatrix} 3F & 5E & 2C & D1 \\ A8 & 77 & 6B & 90 \\ C4 & 92 & 11 & 7A \\ 8D & 3B & F4 & 26 \end{bmatrix}$

Putaran ketujuh: $P_7 = \begin{bmatrix} E2 & 14 & 6A & 8F \\ 7D & 55 & 3C & A1 \\ 19 & 2E & D7 & 44 \\ 63 & F8 & 0B & 9C \end{bmatrix}$

Putaran kedelapan: $P_8 = \begin{bmatrix} B4 & 7A & 0F & 2E \\ C1 & 39 & 58 & 6D \\ 72 & 16 & 91 & F3 \\ 05 & 4C & E8 & 17 \end{bmatrix}$

Putaran kesembilan: $P_9 = \begin{bmatrix} 5F & 21 & 9C & 44 \\ A3 & 6E & 77 & 0B \\ D8 & 14 & 52 & E1 \\ 7A & C5 & 3F & 69 \end{bmatrix}$

7. Pada putaran terakhir tanpa melakukan *MixColumns*, hanya melakukan *SubBytes*, *ShiftRows*, dan *AddRoundKey*.

   Putaran terakhir: $P_{10} = \begin{bmatrix} E5 & 56 & C7 & A4 \\ 5C & 37 & A6 & A7 \\ 55 & 4B & 2D & 52 \\ 77 & F1 & E2 & 72 \end{bmatrix}$

   Sehingga hasil pada putaran terakhir tersebut menjadi representasi *ciphertext* akhir dari proses enkripsi yang telah dilakukan.

   Plaintext yang digunakan adalah kata TERENKRIPSI dengan panjang 11 *byte*. Karena algoritma enkripsi bekerja dalam blok berukuran 16 *byte*, maka diperlukan penambahan *padding* sebanyak 5 *byte* untuk memenuhi ukuran blok (16 − 11 = 5). Skema *padding* yang digunakan mengikuti metode PKCS#7, yaitu dengan menambahkan nilai heksadesimal 05 sebanyak lima

kali di akhir data. Dengan demikian, representasi *plaintext* dalam format heksadesimal menjadi:

544552454E4B52495053490505050505

Setelah melalui proses enkripsi, diperoleh *ciphertext* dalam format heksadesimal yaitu:

E55C557756374BF1C7A62DE2A4A75272

8. Enkripsi Kunci AES Menggunakan RSA

Setelah proses enkripsi pesan menggunakan AES-128 selesai, langkah berikutnya adalah mengenkripsi kunci AES agar dapat dikirimkan secara aman. Pada penelitian ini, proses pengamanan kunci dilakukan menggunakan algoritma RSA melalui perhitungan manual sebagai berikut:

a. Menentukan Bilangan Prima

$$p = 17,\ \ q = 11$$

b. Menghitung Modulus (n) sebagaimana dinyatakan pada persamaan (2.8).

$$n = 17 \times 11 = 187$$

Nilai n digunakan pada kunci publik dan privat.

c. Menghitung Fungsi Totien Euler sebagaimana dinyatakan pada Persamaan (2.9).

$$\varphi(n) = 16 \times 10 = 160$$

d. Menetukan Kunci Publik (e)

Memilih nilai e yang memenuhi: $1 < e < \varphi(n)$ dan relatif prima dengan $\varphi(n)$.

Dipilih $e = 7$, karena FPB $(7{,}160) = 1$.

e. Menentukan Kunci Privat (d) sebagaimana dinyatakan pada Persamaan (2.11).

$$d \times 7 \equiv 1 \ (mod\ 160)$$

Dicari *Inv*ers modular $d = 23$.

Karena $23 \times 7 = 161 \equiv 1 \ (mod\ 160)$.

f. Pasangan Kunci RSA:

Kunci Publik→ $(e, n) = (7{,}187)$

Kunci Privat→ $(d, n) = (23{,}187)$

g. Representasi Numerik Kunci AES

Kunci AES: KUNCIPERHITUNGAN.

Kunci yang sudah dikonversi ke ASCII:

[75 85 78 67 73 80 69 82 72 73 84 85 78 71 65 78]

Diambil contoh satu karakter untuk perhitungan manual, misalnya huruf pertama:

$$K \rightarrow ASCII = 75$$

Sehingga:

$$M = 75$$

h. Proses Enkripsi RSA sebagaimana dinyatakan pada Persamaan (2.12).

$$C = 75^7\ mod\ 187$$

Perhitungan Bertahap (Modular Exponensial)

$$75^2 = 5625\ mod\ 187 = 15$$

$$75^4 = 15^2 = 225\ mod\ 187 = 38$$

$$75^7 = 75^4 \times 75^2 \times 75$$

$$= 38 \times 15 \times 75$$

$$38 \times 15 = 570 \; mod \; 187 = 9$$

$$9 \times 75 = 675 \; mod \; 187 = 114$$

Hasil enkripsi: $C = 114$

Jadi nilai *ciphertext* RSA untuk sampel kunci AES adalah 114.

9. Dengan langkah perhitungan yang sama, seluruh karakter kunci AES dienkripsi dan hasilnya disajikan secara langsung. Hasil enkripsi RSA untuk setiap karakter kunci AES ditunjukkan pada Tabel 4.3. berikut:

Tabel 4.3 Hasil Perhitungan Modular Eksponensial Enkripsi

Karakter	M	$C = M^7 \; mod \; 187$
K	75	114
U	85	51
N	78	177
C	67	56
I	73	30
P	80	9
E	69	86
R	82	135
H	72	30
I	73	30
T	84	101
U	85	51
N	78	177
G	71	182
A	65	109
N	78	177

Hasil *ciphertext* kunci AES adalah:

$$[114\ 51\ 177\ 56\ 30\ 9\ 86\ 135\ 30\ 30\ 101\ 51\ 177\ 182\ 109\ 177]$$

Atau ditulis dalam representasi heksadesimal:

$$72\ 33\ B1\ 38\ 1E\ 09\ 56\ 87\ 1E\ 1E\ 65\ 33\ B1\ B6\ 6D\ B1$$

10. *Output* yang dihasilkan dari proses *Hybrid* AES-RSA dalam representasi heksadesimal adalah:

    Ciphertext: E55C557756374BF1C7A62DE2A4A75272

    Cipherkey: 7233B1381E0956871E1E6533B1B66DB1

4.1.6 Enkripsi *Hybrid* AES-RSA

1. Pada tahap pertama skema *Hybrid,* pesan teks dienkripsi menggunakan algoritma AES dengan kunci simetris (K_{AES}) yang dibangkitkan secara acak untuk setiap sesi enkripsi. Hasil proses ini berupa *ciphertext* dan *Initialization Vector* (IV) yang digunakan dalam mode CBC.

Tabel 4.4 Hasil Enkripsi Pesan Menggunakan AES

No	Panjang Pesan (Karakter)	*Initialization Vector* (hex)	*Ciphertext* (hex)	Panjang *Output*
1	50 karakter	7d0a4217501904f7ab08e92d4852d3f9	68c32ff3fb53d4175bfd4a9bbfe03ddeca0f8282e3274fcfb31d1f9338b1b4db1926fb56e79926b6317e405f8e6f6a24378bc9adf4dc91f2e6b2dddda7e067d7	64 *byte*
2	75 karakter	4354ed29c1b0c4001a3287d301c317fe	f68df69d06df5fccc675df08a9edab9a754cae99d9b65220c0361371f21e01735c970aaee057b9c57eaa408d8be26e9cded5c2f7cadbed2abc27002ca976b2f984be0ea378c880b334e880aa7fd6ef00	80 *byte*

No	Panjang Pesan (Karakter)	*Initialization Vector* (hex)	*Ciphertext* (hex)	Panjang *Output*
3	100 karakter	8edf176d5751049967ee57697d66825b	47ebdf26f62074f02eecdfd118f60df0d037968ddc4a319f352f6439ca64525b83fc011d1a941404aaa71293349b5cd7b53ef71b9ac63dcd80bde89471a28965cb9ad9b5bafa753375e19f68913a8d9aec5030641bc4abedc5f3d9472d610d12e50b995a8678f419b15e7e3d65d612e3	112 *byte*
4	150 karakter	3aef5649822a44c7c6f4a50738d1965b	987317889077f4ef4399a9920b2afd9291c256628238550de62d1270ea17f6743c41f78cbbc12397802dbb49c3fcf1818ad5822a09d496dc49eaf8db733a14d574504b143c0900beed17c6dd2cfe0ed35ca14186bac068c8c2b4b975a522d9cb03ca041a25e5bbb3e5f9a50b39183d2ae569f975dc7cdd8a067e4bd9793d54c643996caab0bb2314d2623c8c5f5defa43713fa410701d26cd34d1093d3c71fc7	176 *byte*

No	Panjang Pesan (Karakter)	*Initialization Vector* (hex)	*Ciphertext* (hex)	Panjang *Output*
5	200 karakter	4ef6dedec1148d5a25cf975172167503	2c1dc11b37421369e6165b9411fc497a0e654047bb193ab11dadcbc981ee9d4c341f0c83de417103fdea98bd8dfbfb11945b104e5426b9ea78ff8a8bff179798c3bb0b78915ca23bf1db4904bae1f53ab4e8882488d6662e58244a6fda721a61ae52269e11b3f39a75e0601d844f3f8da82e300dee50956d8f7f871005e7f7fe0ab193239cb3a8a15f4240d88395019fafc67d8aa91e337d0317108ae625f438fc6b40a058f9e0cbde02dbd19f0dccf03dbf14d2af5c447eeb5bfcedde5dd5511146e7ea11ef3e274926403b707cd10d	208 *byte*

Berdasarkan Tabel 4.4, setiap *plaintext* berhasil dienkripsi menggunakan AES dan menghasilkan *ciphertext* dalam format heksadesimal dengan panjang *output* yang berbeda sesuai panjang pesan. Panjang *ciphertext* berturut-turut adalah 64 *byte*, 80 *byte*, 112 *byte*, 176 *byte,* dan 208 *byte* untuk pesan 50, 75, 100, 150, dan 200 karakter. Peningkatan panjang *output*

tersebut dipengaruhi oleh mekanisme pemrosesan blok AES berukuran tetap 128 bit (16 *byte*), sehingga semakin panjang pesan maka semakin banyak blok yang diproses. Selain itu, setiap proses enkripsi menghasilkan nilai *Initialization Vector* (IV) yang berbeda sebagaimana terlihat pada Tabel 4.4. Perbedaan IV ini menunjukkan bahwa setiap sesi enkripsi bersifat unik meskipun menggunakan algoritma dan mekanisme yang sama.

2. Setelah pesan dienkripsi menggunakan AES, kunci simetris (K_{AES}) yang digunakan pada tahap sebelumnya kemudian dienkripsi menggunakan algoritma RSA dengan kunci publik penerima. Proses ini bertujuan untuk mengamankan distribusi kunci AES dalam skema *Hybrid*.

Tabel 4.5 Hasil Enkripsi Kunci AES Menggunakan RSA

No	K_{AES} (hex)	*Cipherkey* (hex)
1	18c3e440e54bab24720e41a40ae1bf99	5d91248952b45d2e9ff4c40f04d923e8936522e2ce1ad47e812c3a9bd06aecdd939453b747753f28eea9fac91e8d50060541b2a59f42977608c176040a82a75ae307bb93ff8469b3d7fa9cf766f5f989d86df034163db9c6b0485f1300b48375d077a4da33d9b63855ac4004f87c286eca0280b6befbb24f9c3a15c93c633a01b5a554e50374f881bff685aaf721a2d78277574f34fb09860a3a20383a21ffc447d6e99bf15deebc22e5161533c0a1fe7e48b82f1ce1c82b9f1bf57e32200c859f1c77263f0a3a2591c9e22a0785ef9d9f3d50c057c649309b3aa3e78bfed2882525e351c7d91845289c81faa77776b46e9e3d1be7e3c70f39a98024dad9a3d6
2	c188c0ac04ac51cb5b97dca3d7bfef7e	6aed7d96a6c36d17b13d63f812b4783c9c8617afbedb0c13e89f0f1d13daf374265f5a49a4a36ed82f25d701ddc0f7a8890885db8c79c3b2

No	K_{AES} (hex)	*Cipherkey* (hex)
		16eaf41df47a844e0ccde362f763ae6e7d8594 1fcadf569579eb5acc0b81967a222acc9428a 57c5bee2acb6d52bb7bb2c000423834d2361 7fe22274c13af50a912c1010aaeb756b74d6c be4981bc487e41ff9f9b5604725719d368dac 6fe7a59d04a95bc83f5c554bb10e3f51ae7a4c d6b337751935d92555f96228e0c2a97e02be 330e2ac4b2db64e30b3ff4c94972f5928eba3 0941736deba8905bf6d86ded7811d368fd87e bd4b288322ef364ce6a4f7e2dc39331170481 a37a9ef37f80535b515270ff0ef536
3	ea7858662c462b74b5bf9ad 01365d8db	adee8bb936f7538108aa6359f1e11bedacdd7f 15444b3a02efda79222f6acb594f222271dc4f 4be99cf3dce81215f324b6ab1d5218221947b dc5086283a7bad0d8636178780b0befd7c13 5c0d58c4d8c01fa62dd936e8011f94543a402 2d7936edc07d579c6b9b89a27a95ae13b8f3d 722e587ff94e46694084e0b59b79326783d49 25674cba2fc5cbc8a8b0163ee2d4d6612e295 cf5f3e04a4b5e73f4c83c900b5dd09df6a4d84 611f48a95b6c36962d630fa24342398a697db 7fae61364039f6ada272ec8abdd714dc63671 7de7afca318bed81f38240c9369eff4ead300e 90669fab55c6c787b41bfd92b41d5cf37ee71 15f82f6a94cdd047acd41b9a6bc1
4	5b4c309a41f3ec75a8cdf2bb 243162fb	0078eb2db60a19f52fd6b920d4a24bde3e804 70025551ef182c1886e0e7c8693867a1a32f1 dc3db047a53a694c262c3361ff685c3337749 b4b1af5d9457d486fe71795688d617cedc119 d97bf7463f1577ac79e8b34150c7fb5374862 5341fde15df454d46a8b7bf6425dec82ef861

No	K_{AES} (hex)	*Cipherkey* (hex)
		d5956ff57e89b7e41e205b773d007d00d276d 6cc10f78049cb4b4594adeb5104fc4178a6fd 1148521479a482689c14d356bdd5de12bc68 65be86ea727f2f9eb7004155bde078e63f680 576878c5cc1614d079972e4480af9a5efb366 1b31ccc24d1a03dd6fb850b8bd4eedf4e1e9fc 40e88388b4ed0f7fecd1d531b6d7b3eac2886 38331881bc0076b1de2b5017ed09ab0
5	0ec61193e379fc057b58ef41 8a8bd98c	4e8fb2f11ba5b601354e3d9a268306c2b5be0 c291f350ecbdfaf6dccdf5bac1604292ac75b4 b219ec444423d05913bc586576ee3c0077c7 ba4a9df579d639d09092b06e31362473eb63 1b683ef327429b942fb8d5d6f3ab04f6a55c6 65f6191e128020ad2edbad1211d9288ab4a8f 4d693bc811f7c2e879f46df0dd88bb8b9870e a4f2df32065984e7f1a360ca00d5ccccc5d15b 21317a9d6a7e5a211b41477aaed7e039c372a ea8ec936c9a95c531948af734fe9975397f632 9f5c52688b3941a7d84995f7ba1b6650c2b24 511fbcfa67c370e18d64b3406804d3c4f081d a332625b3afdbc38522f5a179557207737dcf 100e78f08949a997de63bf1c9094e8

Berdasarkan Tabel 4.5, setiap nilai K_{AES} yang ditampilkan dalam format heksadesimal memiliki panjang 16 *byte* (128 bit), sesuai dengan spesifikasi AES-128 yang digunakan dalam penelitian ini. Nilai K_{AES} berbeda pada setiap percobaan karena dibangkitkan secara acak untuk setiap sesi enkripsi. Setelah dienkripsi menggunakan RSA, setiap K_{AES} menghasilkan *cipherkey* dalam format heksadesimal dengan panjang yang sama. Panjang *cipherkey*

tersebut mengikuti ukuran modulus (n) dari kunci RSA yang digunakan, sehingga ukuran *output* tidak bergantung pada nilai K_{AES}, melainkan pada panjang kunci RSA. Dengan demikian, pada skema *Hybrid* AES–RSA, pesan utama dienkripsi menggunakan AES yang bersifat efisien terhadap ukuran data, sedangkan kunci AES yang berukuran relatif kecil dienkripsi menggunakan RSA untuk menjaga keamanan distribusinya.

4.2 Hasil Simulasi Proses Dekripsi

4.2.1 Dekripsi AES Tunggal

Pada tahap dekripsi AES tunggal, *ciphertext* hasil proses enkripsi sebelumnya dikembalikan menjadi *plaintext* menggunakan kunci simetris yang sama. Proses ini bertujuan untuk menguji konsistensi algoritma AES dalam memulihkan pesan asli.

Tabel 4.6 Hasil Dekripsi AES Tunggal

No	*Ciphertext* (hex)	Hasil *Plaintext*	Status
1	ca08b0626cb72cbeafa8ee191d6e6f9bfb17124e73e243c33ff7d6f2dc6023de4c027c1110fccc3dc8f5465c8225801e2f3f5766d42b402a6194d6e28435cb23	Pesan ini aman dan TERENKRIPSI dengan sangat baik.	Valid
2	f75add990296417005611d3b61487858a5f1fddf953cb4bdbb2c4f76bb97fed28e10a3bdd1d61e08318e144e102c1341865559ac91366082784c8263f7a6831189f864ebc037f1e43be7ad3b1329e0bd	*Hybrid* AES dan RSA mengamankan pesan teks digital secara efisien aman baik	Valid
3	636708ffaae6d0d3192d462c4aa488b74429e44676d25de58b29e1628f79889ce4e3b23d6889682de837cbba	Keamanan pesan teks penting agar data tetap aman, dan sistem	Valid

No	*Ciphertext* (hex)	Hasil *Plaintext*	Status
	94705252eb68187ffb05e43f7515bd aaf8074d077b867782c5b8de206a0 aa8410f1129155e622176060eb02c c0d7b47aa5133edab62ebd2e7b95f2 94bc89788d8c239a77	enkripsi membantu menjaga privasi anda.	
4	0f79b98c22413fb8f2b04acca8d0d6 b136cdac59c39d59b17b87908145a ad991ef94d69691e2fe0310d80c3c5 0e3c26c02ee1f2d16c9b021643360 df6c96b4add925696b8f49a805071 80225ecc1a7a3f5049f8817f9058c7 42a1e47d98f0b66be5a6aedd063fec 8a4e0efba7de8f68cc202e2846c45e b23f6f84505c73849b0c36dae8b1a8 9f252a05390daeada2bb923e1c1b20 b67a3fe3b57485834ae8e70	Sistem *Hybrid* AES dan RSA melindungi pesan teks digital secara aman dan efisien dengan menggabungkan kecepatan AES serta keamanan distribusi kunci RSA	Valid
5	b08e32ccf7feb44033e60a56531593 d8d21f5a1d5b674bc934c1f040f602 f5c54e056b0055c1db6f26d777dd4 621d8b8d39a69813dcef95f4b14f81 4a5239f77ce916938a91d4b9a4df19 52d92d16e85dadc240d9181cfdf82d 098fc5bdb7bb7e1bdec2febeaf8c92 94bd8f2cce7e4c7d131d1ae279bd88 11fd74b34b56d4db1eb16657c0058 3b1d684b220b9eeb1210784f4f627 b70025acf95db83fe059c458f44bbd 51c3cad27915cb6d3290295368e68 e40b113a8b3c063bc828a9a2e30e2 04128b36b4e22724597ff4803b6f7f 4	Hybrid AES dan RSA meningkatkan keamanan pesan teks dengan memadukan kecepatan enkripsi AES dan kekuatan RSA sehingga distribusi kunci tetap aman serta proses komunikasi berlangsung lebih terlindungi.	Valid

Berdasarkan Tabel 4.6, seluruh *ciphertext* berhasil didekripsi kembali menjadi *plaintext* yang sesuai dengan pesan awal, dengan status seluruhnya "Valid". Hal ini menunjukkan bahwa proses dekripsi AES tunggal mampu memulihkan data secara utuh menggunakan kunci simetris yang sama.

4.2.2 Dekripsi RSA Tunggal

Pada tahap dekripsi RSA tunggal, *ciphertext* hasil proses enkripsi RSA dikembalikan menjadi *plaintext* menggunakan kunci privat RSA yang sesuai. Tahap ini dilakukan untuk memastikan bahwa proses dekripsi dapat memulihkan pesan asli dengan benar.

Tabel 4.7 Hasil Dekripsi RSA Tunggal

No	*Ciphertext* (hex)	Hasil *Plaintext*	Status
1	cf0ac4bae7ff5aa731f459a241d647c9f 9da4b00fb93dd437d277ff04bc064432 32b7144f5e41f39e45c068dee1b5f5fbf bc818e09ca3f9dfc9f337fb70f1bb0afb 13ffeb6d2e27ac3ef3a883593b993c5e2 aba7abfed741347f011b4a4d757507b3 e8023afc1347eee37c6badbbc2da325c 74b89dfd8cbf02a188c6985745cf6735 3f2e014e259fb99944d3c947dad9c9ac 986f40cb9139a3311828bb3bf19bcee3 6feaa9ac0e9f8344b4eceed1b62b1d426 2fd9408b701e0f186515668472a8809f 8d6a28d7f9bfabbe9a5f804add9b154f7 e195005c9d66877065634e407bfd888c bfa197931b137f8fdf8893bb68b5d18fb dd516a0544acd136d647ddcef	Pesan ini aman dan TERENKRIPSI dengan sangat baik.	Valid

No	*Ciphertext* (hex)	Hasil *Plaintext*	Status
2	8f0418cdecb7b17af19fd7fd28cdaf192 d60b0be1e3ec2a4ef11e602f7ef6fd469 695f9223b6620d4a09e1385d3fd1fea7 618bfef51dab6657e3a53e96548f48ff7 d1c43d2086b6019d0d57ea8a038b8c4 8cf010b31cd518039fe4dd6427b407aa b8b758c98693f6f4252e710195c89541 65f2854d1a05a6afc1ff9f7730536e1a8 10fdd8318ad59094612539728081845 c00c18ffb381aecb5c8776d31a16f8a01 736fde7160a5a1586d2a1f9732278aaf 0312a6406b392e035894d8df76a47d5 267e9acb82bb0c7cd87d07d7005994ce 98326ad9887ef9cbaaf663b4f490c0d8 7fb4e13fe06a390b9c6f0c35f31f9f1db c9e0cf7b874f21b3ad5d578a57f90	*Hybrid* AES dan RSA mengamankan pesan teks digital secara efisien aman baik	Valid
3	25d410038619e5b006597b808c4eed5 4b4e0bc139585bbf04cfc7572469bd94 3b6aa96c5e07202be017b45b5512a92c a430cf2ff4577afcdfcf935554c0d9b99 d43fe13514cd8bc4b971ec09e664361b c99bb6f4a868b88a78c41cd7a9209c26 613a3e50ac273013fa0e12304f8d5fc1a fdd1aedb880b7ea61d745aceea737d98 eda0a2487e71c6a9ab31250886640d63 c97b8ff884e09dc5a12333fcef36e0913 5a7bd152c46716858f3c82e260ddb461 96513f74f0de23448fd6478df0d88740 1ee34f71a564bbd139f6a879e19a03e2 7803e51c9bf0c7e12ae6ccfbfa1c74b9f	Keamanan pesan teks penting agar data tetap aman, dan sistem enkripsi membantu menjaga privasi anda.	Valid

No	*Ciphertext* (hex)	Hasil *Plaintext*	Status
	9d3ea3c0650b78b2429279d9d60adb3 03e0ec53b9b3a05d1fa3c4ff08af82		
4	bc64d767e15f241dc194942abd2c60a9 87dd8f417d239545ef4351d597f75da4 30307e0565f4fc31df182d7cb96207eb 93e6a4c8fce8b58f298d944ceca6a5c1a dc74a85c03797ec7a6e054bd53b87b80 4d80788164160e2e03a1e46facbc0726 2b4e954623b210885c55bcd4e9fa27b4 09d5647bd9c00e08743d2a756c6f3b00 e1bf3195377c0516847bb2874263928 127e08dfb296601f1c7b641baf0d4918 0d06500eea56786200e7e4999b1a28c4 8a9864347e7f1f7e6138362ed44e4bc4 a97a2f034ddd193573ec94c8aa5b005c 0f35c5318e7618eae96f557d86608987 324f9acd3d5d7b93f4fe7aec9b8db549 6afba7a0a77083ea6d52fa43b8e88d00	Sistem *Hybrid* AES dan RSA melindungi pesan teks digital secara aman dan efisien dengan menggabungkan kecepatan AES serta keamanan distribusi kunci RSA	Valid
5	Tidak Tersedia (enkripsi gagal)	-	Tidak Dapat Didekripsi

Berdasarkan Tabel 4.7, seluruh *ciphertext* berhasil didekripsi menjadi *plaintext* yang identik dengan pesan awal dengan status “Valid”. Hal ini menunjukkan bahwa proses dekripsi RSA berjalan dengan baik menggunakan pasangan kunci privat yang sesuai, sehingga mampu mengembalikan pesan ke bentuk semula.

Namun, pada data dengan panjang yang lebih besar (±200 karakter), proses dekripsi tidak dapat dilakukan karena sebelumnya proses enkripsi RSA tidak berhasil. Hal ini menegaskan bahwa keterbatasan RSA dalam menangani ukuran

data juga berdampak pada tahap dekripsi, sehingga RSA tidak sesuai untuk pengolahan pesan berukuran besar secara langsung.

4.2.3 Dekripsi *Hybrid* AES-RSA dengan Perhitungan Manual

Berikut contoh langkah-langkah perhitungan manual *Hybrid* AES-RSA

1. Dekripsi Kunci AES Menggunakan RSA

   Setelah proses enkripsi kunci AES dilakukan dengan kunci publik RSA $(e, n) = (7,\ 187)$, maka tahap berikutnya adalah mengembalikan *ciphertext* menjadi nilai semula menggunakan kunci privat. Kunci privat $(d, n) = (23,\ 187)$.

   Rumus dekripsi RSA sebagaimana dinyatakan pada Persamaan (2.13).

   Cipherkey hasil enkripsi sebelumnya:

   [114 51 177 56 30 9 86 135 30 30 101 51 177 182 109 177]

   Ambil contoh *ciphertext* pertama:

   $$C = 114$$

   Rumus:

   $$M = 114^{23}\ mod\ 187$$

   Menggunakan metode modular eksponensial. Ubah pangkat 23 ke bentuk biner:

   $$23 = 16 + 4 + 2 + 1$$

   Jadi:

   $$114^{23} = 114^{16} \times 114^{4} \times 114^{2} \times 114^{1}$$

   Perhitungan bertahap modulo 187

   $$114^{1}\ mod\ 187 = 114$$

   $$114^{2} = 12996\ mod\ 187 = 86$$

$$114^4 = 86^2 = 7396 \; mod \; 187 = 103$$

$$114^8 = 103^2 = 10609 \; mod \; 187 = 137$$

$$114^{16} = 137^2 = 18769 \; mod \; 187 = 69$$

Kalikan sesuai biner:

$$114^{23} = 114^{16} \times 114^4 \times 114^2 \times 114^1$$

$$= 69 \times 103 \times 86 \times 114$$

Hitung bertahap:

$$69 \times 103 = 7107 \; mod \; 187 = 1$$

$$1 \times 86 = 86$$

$$86 \times 114 = 9804 \; mod \; 187 = 75$$

Hasil $M = 75$. Dalam ASCII $75 = K$. Dekripsi berhasil mengembalikan karakter pertama.

2. Dengan langkah perhitungan yang sama, seluruh karakter kunci AES didekripsi dan hasilnya disajikan secara langsung. Hasil dekripsi RSA untuk setiap karakter kunci AES ditunjukkan pada Tabel 4.8.

   Hasil akhir dekripsi kunci AES dalam nilai ASCII adalah:

   [75 85 78 67 73 80 69 82 73 73 84 85 78 71 65 78]

   Jika dikonversi ke karakter menjadi:

   K U N C I P E R H I T U N G A N

   Maka hasil dekripsi berhasil mengembalikan kunci AES ke bentuk semula.

Tabel 4.8 Hasil Perhitungan Modular Eksponensial Dekripsi

Ciphertext (C)	$M = C^{23} \ mod \ 187$	ASCII	Karakter
114	75	75	K
51	85	85	U
177	78	78	N
56	67	67	C
30	73	73	I
9	80	80	P
86	69	69	E
135	82	82	R
30	72	72	H
30	73	73	I
101	84	84	T
51	85	85	U
177	78	78	N
182	71	71	G
109	65	65	A
177	78	78	N

3. Setelah proses enkripsi selesai, tahap selanjutnya adalah melakukan dekripsi untuk mengembalikan *ciphertext* menjadi *plaintext* semula.

   Kata yang sudah terenkripsi disebut sebagai *ciphertext*. Setelah dikonversi, maka blok pertama diwakili oleh:

$$P_{10} = \begin{bmatrix} E5 & 56 & C7 & A4 \\ 5C & 37 & A6 & A7 \\ 55 & 4B & 2D & 52 \\ 77 & F1 & E2 & 72 \end{bmatrix}$$

   a. Langkah 1: *InvAddRoundKey*

      Melakukan *InvAddRoundKey* dengan melakukan operasi XOR antara blok P_{10} dengan kunci terakhir K_{10}.

      Disusun ke matrix *state* AES:

$$P_{10} = S_{10}\begin{bmatrix} E5 & 56 & C7 & A4 \\ 5C & 37 & A6 & A7 \\ 55 & 4B & 2D & 52 \\ 77 & F1 & E2 & 72 \end{bmatrix}$$

Kunci yang digunakan:

$$K_{10} = \begin{bmatrix} 2A & AB & 19 & BF \\ C3 & C2 & 8D & AD \\ 55 & B3 & 4C & A8 \\ 8E & 2B & 44 & 07 \end{bmatrix}$$

Rumus: $S' = S_{10} \oplus K_{10}$

$$Hasil\ XOR = \begin{bmatrix} E5 \oplus 2A & 56 \oplus AB & C7 \oplus 19 & A4 \oplus BF \\ 5C \oplus C3 & 37 \oplus C2 & A6 \oplus 8D & A7 \oplus AD \\ 55 \oplus 55 & 4B \oplus B3 & 2D \oplus 4C & 52 \oplus A8 \\ 77 \oplus 8E & F1 \oplus 2B & E2 \oplus 44 & 72 \oplus 07 \end{bmatrix}$$

Perhitungan XOR manual:

Baris 1:

$$E5 \oplus 2A = 11100101 \oplus 00101010 = 11001111 = CF$$

$$56 \oplus AB = 01010110 \oplus 10101011 = 11111101 = FD$$

$$C7 \oplus 19 = 11000111 \oplus 00011001 = 11011110 = DE$$

$$A4 \oplus BF = 10100100 \oplus 10111111 = 00011011 = 1B$$

Baris 2:

$$5C \oplus C3 = 01011100 \oplus 11000011 = 10011111 = 9F$$

$$37 \oplus C2 = 00110111 \oplus 11000010 = 11110101 = F5$$

$$A6 \oplus 8D = 10100110 \oplus 10001101 = 00101011 = 2B$$

$$A7 \oplus AD = 10100111 \oplus 10101101 = 00001010 = 0A$$

Baris 3:

$$55 \oplus 55 = 01010101 \oplus 01010101 = 00000000 = 00$$

$$4B \oplus B3 = 01001011 \oplus 10110011 = 11111000 = F8$$

$$2D \oplus 4C = 00101101 \oplus 01001100 = 01100001 = 61$$

$$52 \oplus A8 = 01010010 \oplus 10101000 = 11111010 = FA$$

Baris 4:

$$77 \oplus 8E = 01110111 \oplus 10001110 = 11111001 = F9$$

$$F1 \oplus 2B = 11110001 \oplus 00101011 = 11011010 = DA$$

$$E2 \oplus 44 = 11100010 \oplus 01000100 = 10100110 = A6$$

$$72 \oplus 07 = 01110010 \oplus 00000111 = 01110101 = 75$$

Setelah seluruh dilakukan operasi XOR, diperoleh matriks hasil XOR sebagai berikut:

$$S_9 = \begin{bmatrix} CF & FD & DE & 1B \\ 9F & F5 & 2B & 0A \\ 00 & F8 & 61 & FA \\ F9 & DA & A6 & 75 \end{bmatrix}$$

b. Langkah 2: *InvShiftRows*

Menggunakan matrix hasil dari *InvAddRoundKey*

$$S_9 = \begin{bmatrix} CF & FD & DE & 1B \\ 9F & F5 & 2B & 0A \\ 00 & F8 & 61 & FA \\ F9 & DA & A6 & 75 \end{bmatrix}$$

Baris 0: $[CF, FD, DE, 1B]$, tidak digeser ke kanan.

Baris 1: $[9F, F5, 2B, 0A]$, digeser satu *byte* ke kanan menjadi $[0A, 9F, F5, 2B]$.

Baris 2: $[00, F8, 61, FA]$, digeser dua *byte* ke kanan menjadi $[61, FA, 00, F8]$.

Baris 3: $[F9, DA, A6, 75]$, digeser ke kanan tiga *byte* menjadi $[DA, A6, 75, F9]$.

Sehingga hasilnya adalah:

$$S_9 = \begin{bmatrix} CF & FD & DE & 1B \\ 9F & F5 & 2B & 0A \\ 00 & F8 & 61 & FA \\ F9 & DA & A6 & 75 \end{bmatrix} \rightarrow S_8 = \begin{bmatrix} CF & FD & DE & 1B \\ 0A & 9F & F5 & 2B \\ 61 & FA & 00 & F8 \\ DA & A6 & 75 & F9 \end{bmatrix}$$

c. Langkah 3: *InvSubBytes*

Menggunakan matrix hasil dari *InvShiftRows*

$$S_8 = \begin{bmatrix} CF & FD & DE & 1B \\ 0A & 9F & F5 & 2B \\ 61 & FA & 00 & F8 \\ DA & A6 & 75 & F9 \end{bmatrix}$$

Baris 1:

$$CF \rightarrow \text{S-Box}\ (C, F) = 5F$$

$$FD \rightarrow \text{S-Box}\ (F, D) = 21$$

$$DE \rightarrow \text{S-Box}\ (D, E) = 9C$$

$$1B \rightarrow \text{S-Box}\ (1, B) = 44$$

Baris 2:

$$0A \rightarrow \text{S-Box}\ (0, A) = A3$$

$$9F \rightarrow \text{S-Box}\ (9, F) = 6E$$

$$F5 \rightarrow \text{S-Box}\ (F, 5) = 77$$

$$2B \rightarrow \text{S-Box}\ (2, B) = 0B$$

Baris 3:

$$61 \rightarrow \text{S-Box}\ (6,1) = D8$$

$$FA \rightarrow \text{S-Box}\ (F, A) = 14$$

$$00 \rightarrow \text{S-Box}\ (0,0) = 52$$

$$F8 \rightarrow \text{S-Box}\ (F, 8) = E1$$

Baris 4:

$$DA \rightarrow \text{S-Box}\ (D, A) = 7A$$

$$A6 \rightarrow \text{S-Box}\ (A, 6) = C5$$

$$75 \rightarrow \text{S-Box}\ (7,5) = 3F$$

$$F9 \rightarrow \text{S-Box}\ (F, 9) = 69$$

Sehingga hasilnya adalah:

$$S_8 = \begin{bmatrix} CF & FD & DE & 1B \\ 0A & 9F & F5 & 2B \\ 61 & FA & 00 & F8 \\ DA & A6 & 75 & F9 \end{bmatrix} \rightarrow S_7 = \begin{bmatrix} 5F & 21 & 9C & 44 \\ A3 & 6E & 77 & 0B \\ D8 & 14 & 52 & E1 \\ 7A & C5 & 3F & 69 \end{bmatrix}$$

d. Langkah 4: *InvAddRoundKey*

Menggunakan matrix hasil dari *InvSubBytes*

$$S_7 = \begin{bmatrix} 5F & 21 & 9C & 44 \\ A3 & 6E & 77 & 0B \\ D8 & 14 & 52 & E1 \\ 7A & C5 & 3F & 69 \end{bmatrix}$$

Kunci yang digunakan adalah K_9

$$K_9 = \begin{bmatrix} 64 & 81 & B2 & A6 \\ D1 & 01 & 4F & 20 \\ 0F & E6 & FF & E4 \\ 9A & A5 & 6F & 43 \end{bmatrix}$$

Proses *AddRoundKey* dilakukan dengan XOR tiap elemenen yang bersesuaian:

Baris 1:

$$5F \oplus 64 = 01011111 \oplus 01100100 = 00111011 = 3B$$

$$21 \oplus 81 = 00100001 \oplus 10000001 = 10100000 = A0$$

$$9C \oplus B2 = 10011100 \oplus 10110010 = 00101110 = 2E$$

$$44 \oplus A6 = 01000100 \oplus 10100110 = 11100010 = E2$$

Baris 2:

$$A3 \oplus D1 = 10100011 \oplus 11010001 = 01110010 = 72$$

$$6E \oplus 01 = 01101110 \oplus 00000001 = 01101111 = 6F$$

$$77 \oplus 4F = 01110111 \oplus 01001111 = 00111000 = 38$$

$$0B \oplus 20 = 00001011 \oplus 00100000 = 00101011 = 2B$$

Baris 3:

$$D8 \oplus 0F = 11011000 \oplus 00001111 = 11010111 = D7$$

$$14 \oplus E6 = 00010100 \oplus 11100110 = 11110010 = F2$$

$$52 \oplus FF = 01010010 \oplus 11111111 = 10101101 = AD$$

$$E1 \oplus E4 = 11100001 \oplus 11100100 = 00000101 = 05$$

Baris 4:

$$7A \oplus 9A = 01111010 \oplus 10011010 = 11100000 = E0$$

$$C5 \oplus A5 = 11000101 \oplus 10100101 = 01100000 = 60$$

$$3F \oplus 6F = 00111111 \oplus 01101111 = 01010000 = 50$$

$$69 \oplus 43 = 01101001 \oplus 01000011 = 00101010 = 2A$$

Setelah seluruh dilakukan operasi XOR, diperoleh matriks hasil XOR sebagai berikut:

$$S_6 = \begin{bmatrix} 3B & A0 & 2E & E2 \\ 72 & 6F & 38 & 2B \\ D7 & F2 & AD & 05 \\ E0 & 60 & 50 & 2A \end{bmatrix}$$

e. Langkah 5: *InvMixColumns*

$$S_6 = \begin{bmatrix} 3B & A0 & 2E & E2 \\ 72 & 6F & 38 & 2B \\ D7 & F2 & AD & 05 \\ E0 & 60 & 50 & 2A \end{bmatrix}$$

Operasi *InvMixColumns* terhadap kolom pertama:

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} 3B \\ 72 \\ D7 \\ E0 \end{bmatrix} = \begin{bmatrix} BA \\ C8 \\ AA \\ A6 \end{bmatrix}$$

i. Perhitungan elemen pertama (baris 1 kolom 1)

$(0E \cdot 3B) \oplus (0B \cdot 72) \oplus (0D \cdot D7) \oplus (09 \cdot E0)$

Menghitung $0E \times 3B$:

Ubah ke biner: $3B = 00111011$

$0E = 1110_2 \rightarrow pakai\ pecahan\ 08, 04, 02$

i) $02 \times 3B$

Geser kiri 1 bit:

$00111011 \rightarrow 01110110 = 76$

(Tidak overflow → tidak $\oplus 1B$)

ii) $04 \times 3B$

Kalikan lagi dengan 02:

$01110110 \rightarrow 11101100 = EC$

iii) $08 \times 3B$

Kalikan lagi dengan 02:

$11101100 \rightarrow 11011000$

Karena bit depan = $1 \rightarrow \oplus B$:

$11011000 \oplus 00011011 = 11000011 = C3$

iv) Gabungkan XOR

$C3 \oplus EC \oplus 76 = C3 \oplus EC = 2F$

$= 2F \oplus 76 = 59$

Jadi, hasil perhitungan dari $0E \times 3B$ adalah 59.

Menghitung $0B \times 72$:

Ubah ke biner: $72 = 01110010$

$0B = 1011_2 \rightarrow pakai\ pecahan\ 08, 02, 01$

i) 02×72

$01110010 \rightarrow 11100100 = E4$

ii) 04×72

$11100100 \rightarrow 11001000 \oplus 1B = D3$

iii) 08×72

$11001000 \rightarrow 10100110 \oplus 1B = BD$

iv) Gabungkan XOR

$BD \oplus E4 \oplus 72 = BD \oplus E4 = 59$

$= 59 \oplus 72 = 2B$

Jadi, hasil perhitungan dari $0B \times 72$ adalah $2B$.

Menghitung $0D \times D7$:

Ubah ke biner: $D7 = 11010111$

$0D = 1101_2 \rightarrow pakai\ pecahan\ 08, 04, 01$

i) $02 \times D7$

$11010111 \rightarrow 10101110 \oplus 1B = B5$

ii) $04 \times D7$

$10110101 \rightarrow 01101010 = 6A$

iii) $08 \times D7$

$01101010 \rightarrow 11010100 \oplus 1B = CF$

iv) Gabungkan XOR

$CF \oplus 6A \oplus D7 = CF \oplus 6A = A5$

$$= A5 \oplus D7 = 72$$

Jadi, hasil perhitungan dari $0D \times D7$ adalah 72.

Menghitug $09 \times E0$:

Ubah ke biner: $E0 = 11100000$

$09 = 1001_2 \rightarrow pakai\ pecahan\ 08, 01$

i) $02 \times E0$

$11100000 \rightarrow 11000000 \oplus 1B = DB$

ii) $04 \times E0$

$11000000 \rightarrow 10110110 \oplus 1B = AD$

iii) $08 \times E0$

$10110110 \rightarrow 01011010 = 5A$

iv) Gabungkan XOR

$E0 \oplus 5A = BA$

Jadi, hasil perhitungan dari $09 \times E0$ adalah BA.

Gabungkan XOR akhir untuk baris 1 kolom 1

$$S_{6(1,1)} = 59 \oplus 2B \oplus 72 \oplus BA$$

$$= 01011001 \oplus 00101011 \oplus 01110010 \oplus 10111010$$

$$= 10111010$$

$$= BA$$

ii. Perhitungan elemen kedua (baris 2 kolom 1)

$(09 \cdot 3B) \oplus (0E \cdot 72) \oplus (0B \cdot D7) \oplus (0D \cdot E0)$

Menghitung $09 \times 3B$:

Ubah ke biner: $3B = 00111011$

$09 = 1001_2 \rightarrow pakai\ pecahan\ 08, 01$

i) $02 \times 3B$

Geser kiri 1 bit:

$00111011 \rightarrow 01110110 = 76$

(Tidak overflow → tidak $\oplus 1B$)

ii) $04 \times 3B$

Kalikan lagi dengan 02:

$01110110 \rightarrow 11101100 = EC$

iii) $08 \times 3B$

Kalikan lagi dengan 02:

$11101100 \rightarrow 11011000$

Karena bit depan = 1 → $\oplus B$:

$11011000 \oplus 00011011 = 11000011 = C3$

iv) Gabungkan XOR

$C3 \oplus 3B = F8$

Jadi, hasil perhitungan dari $09 \times 3B$ adalah $F8$.

Menghitung $0E \times 72$:

Ubah ke biner: $72 = 01110010$

$0E = 1110_2 \rightarrow pakai\ pecahan\ 08, 04, 02$

i) 02×72

$01110010 \rightarrow 11100100 = E4$

ii) 04×72

$11100100 \rightarrow 11001000 \oplus 1B = D3$

iii) 08×72

$11001000 \rightarrow 10100110 \oplus 1B = BD$

iv) Gabungkan XOR

$$BD \oplus D3 \oplus E4 = BD \oplus D3 = 6E$$

$$= 6E \oplus E4 = 8A$$

Jadi, hasil perhitungan dari $0E \times 72$ adalah $8A$.

Menghitung $0B \times D7$:

Ubah ke biner: $D7 = 11010111$

$0B = 1011_2 \rightarrow pakai\ pecahan\ 08, 02, 01$

i) $02 \times D7$

$$11010111 \rightarrow 10101110 \oplus 1B = B5$$

ii) $04 \times D7$

$$10110101 \rightarrow 01101010 = 6A$$

iii) $08 \times D7$

$$01101010 \rightarrow 11010100 \oplus 1B = CF$$

iv) Gabungkan XOR

$$CF \oplus B5 \oplus D7 = CF \oplus B5 = 7A$$

$$= 7A \oplus D7 = AD$$

Jadi, hasil perhitungan dari $0B \times D7$ adalah AD.

Menghitug $0D \times E0$:

Ubah ke biner: $E0 = 11100000$

$0D = 1101_2 \rightarrow pakai\ pecahan\ 08,04, 01$

i) $02 \times E0$

$$11100000 \rightarrow 11000000 \oplus 1B = DB$$

ii) $04 \times E0$

$$11000000 \rightarrow 10110110 \oplus 1B = AD$$

iii) $08 \times E0$

$10110110 \rightarrow 01011010 = 5A$

iv) Gabungkan XOR

$5A \oplus AD \oplus E0 = 5A \oplus AD = F7$

$= F7 \oplus E0 = 17$

Jadi, hasil perhitungan dari $0D \times E0$ adalah 17.

Gabungkan XOR akhir untuk baris 2 kolom 1

$S_{6(2,1)} = F8 \oplus 8A \oplus AD \oplus 17$

$= 11111000 \oplus 10001010 \oplus 10101101 \oplus 00010111$

$= 11001000$

$= C8$

iii. Perhitungan elemen ketiga (baris 3 kolom 1)

$(0D \cdot 3B) \oplus (09 \cdot 72) \oplus (0E \cdot D7) \oplus (0B \cdot E0)$

Menghitung $0D \times 3B$:

Ubah ke biner: $3B = 00111011$

$0D = 1101_2 \rightarrow pakai\ pecahan\ 08{,}04{,}01$

i) $02 \times 3B$

Geser kiri 1 bit:

$00111011 \rightarrow 01110110 = 76$

(Tidak overflow → tidak $\oplus 1B$)

ii) $04 \times 3B$

Kalikan lagi dengan 02:

$01110110 \rightarrow 11101100 = EC$

iii) $08 \times 3B$

Kalikan lagi dengan 02:

$11101100 \rightarrow 11011000$

Karena bit depan = $1 \rightarrow \oplus B$:

$11011000 \oplus 00011011 = 11000011 = C3$

iv) Gabungkan XOR

$C3 \oplus EC \oplus 3B = C3 \oplus EC = 2F$

$= 2F \oplus 3B = 14$

Jadi, hasil perhitungan dari $0D \times 3B$ adalah 14.

Menghitung 09×72:

Ubah ke biner: $72 = 01110010$

$09 = 1001_2 \rightarrow pakai\ pecahan\ 08, 01$

i) 02×72

$01110010 \rightarrow 11100100 = E4$

ii) 04×72

$11100100 \rightarrow 11001000 \oplus 1B = D3$

iii) 08×72

$11001000 \rightarrow 10100110 \oplus 1B = BD$

iv) Gabungkan XOR

$BD \oplus 72 = CF$

Jadi, hasil perhitungan dari 09×72 adalah CF.

Menghitung $0E \times D7$:

Ubah ke biner: $D7 = 11010111$

$0E = 1110_2 \rightarrow pakai\ pecahan\ 08, 04, 02$

i) $02 \times D7$

$11010111 \rightarrow 10101110 \oplus 1B = B5$

ii) $04 \times D7$

$10110101 \rightarrow 01101010 = 6A$

iii) $08 \times D7$

$01101010 \rightarrow 11010100 \oplus 1B = CF$

iv) Gabungkan XOR

$CF \oplus 6A \oplus B5 = CF \oplus 6A = A5$

$= A5 \oplus B5 = 10$

Jadi, hasil perhitungan dari $0E \times D7$ adalah 10.

Menghitug $0B \times E0$:

Ubah ke biner: $E0 = 11100000$

$0B = 1011_2 \rightarrow pakai\ pecahan\ 08{,}02{,}01$

i) $02 \times E0$

$11100000 \rightarrow 11000000 \oplus 1B = DB$

ii) $04 \times E0$

$11000000 \rightarrow 10110110 \oplus 1B = AD$

iii) $08 \times E0$

$10110110 \rightarrow 01011010 = 5A$

iv) Gabungkan XOR

$5A \oplus DB \oplus E0 = 5A \oplus DB = 81$

$= 81 \oplus E0 = 61$

Jadi, hasil perhitungan dari $0B \times E0$ adalah 61.

Gabungkan XOR akhir untuk baris 3 kolom 1

$$S_{6(3,1)} = 14 \oplus CF \oplus 10 \oplus 61$$

$$= 00010100 \oplus 11001111 \oplus 00010000 \oplus 01100001$$

$$= 10101010$$

$$= AA$$

iv. Perhitungan elemen keempat (baris 4 kolom 1)

$(0B \cdot 3B) \oplus (0D \cdot 72) \oplus (09 \cdot D7) \oplus (0E \cdot E0)$

Menghitung $0B \times 3B$:

Ubah ke biner: $3B = 00111011$

$0B = 1011_2 \rightarrow pakai\ pecahan\ 08{,}02{,}01$

i) $02 \times 3B$

Geser kiri 1 bit:

$00111011 \rightarrow 01110110 = 76$

(Tidak overflow → tidak $\oplus 1B$)

ii) $04 \times 3B$

Kalikan lagi dengan 02:

$01110110 \rightarrow 11101100 = EC$

iii) $08 \times 3B$

Kalikan lagi dengan 02:

$11101100 \rightarrow 11011000$

Karena bit depan = 1 → $\oplus B$:

$11011000 \oplus 00011011 = 11000011 = C3$

iv) Gabungkan XOR

$C3 \oplus 76 \oplus 3B = C3 \oplus 76 = B5$

$$= B5 \oplus 3B = 8E$$

Jadi, hasil perhitungan dari $0B \times 3B$ adalah $8E$.

Menghitung $0D \times 72$:

Ubah ke biner: $72 = 01110010$

$0D = 1101_2 \rightarrow pakai\ pecahan$ 08,04,01

i) 02×72

$01110010 \rightarrow 11100100 = E4$

ii) 04×72

$11100100 \rightarrow 11001000 \oplus 1B = D3$

iii) 08×72

$11001000 \rightarrow 10100110 \oplus 1B = BD$

iv) Gabungkan XOR

$BD \oplus D3 \oplus 72 = BD \oplus D3 = 6E$

$= 6E \oplus 72 = 1C$

Jadi, hasil perhitungan dari $0D \times 72$ adalah $1C$.

Menghitung $09 \times D7$:

Ubah ke biner: $D7 = 11010111$

$09 = 1001_2 \rightarrow pakai\ pecahan\ 08, 01$

i) $02 \times D7$

$11010111 \rightarrow 10101110 \oplus 1B = B5$

ii) $04 \times D7$

$10110101 \rightarrow 01101010 = 6A$

iii) $08 \times D7$

$01101010 \rightarrow 11010100 \oplus 1B = CF$

iv) Gabungkan XOR

$CF \oplus D7 = 18$

Jadi, hasil perhitungan dari $09 \times D7$ adalah 18.

Menghitug $0E \times E0$:

Ubah ke biner: $E0 = 11100000$

$0E = 1110_2 \rightarrow pakai\ pecahan\ 08{,}04{,}02$

i) $02 \times E0$

$11100000 \rightarrow 11000000 \oplus 1B = DB$

ii) $04 \times E0$

$11000000 \rightarrow 10110110 \oplus 1B = AD$

iii) $08 \times E0$

$10110110 \rightarrow 01011010 = 5A$

iv) Gabungkan XOR

$5A \oplus AD \oplus DB = 5A \oplus AD = F7$

$= F7 \oplus DB = 2C$

Jadi, hasil perhitungan dari $0E \times E0$ adalah $2C$.

Gabungkan XOR akhir untuk baris 4 kolom 1

$S_{6(4,1)} = 8E \oplus 1C \oplus 18 \oplus 2C$

$= 10001110 \oplus 00011100 \oplus 00011000 \oplus 00101100$

$= 10100110$

$= A6$

Proses perhitungan *InvMixColumns* pada kolom kedua, ketiga, dan keempat dilakukan dengan langkah yang sama seperti pada kolom pertama.

Operasi *InvMixColumns* pada kolom kedua:

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} A0 \\ 6F \\ F2 \\ 60 \end{bmatrix} = \begin{bmatrix} F7 \\ 5C \\ 2E \\ A3 \end{bmatrix}$$

i. Elemen (baris 1, kolom 2):

$$(0E \cdot A0) \oplus (0B \cdot 6F) \oplus (0D \cdot F2) \oplus (09 \cdot 60) = F7$$

ii. Elemen (baris 2, kolom 2):

$$(09 \cdot A0) \oplus (0E \cdot 6F) \oplus (0B \cdot F2) \oplus (0D \cdot 60) = 5C$$

iii. Elemen (baris 3, kolom 2):

$$(0D \cdot A0) \oplus (09 \cdot 6F) \oplus (0E \cdot F2) \oplus (0B \cdot 60) = 2E$$

iv. Elemen (baris 4, kolom 2):

$$(0B \cdot A0) \oplus (0D \cdot 6F) \oplus (09 \cdot F2) \oplus (0E \cdot 60) = A3$$

Operasi *InvMixColumns* pada kolom ketiga:

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} 2E \\ 38 \\ AD \\ 50 \end{bmatrix} = \begin{bmatrix} 6F \\ 0C \\ E7 \\ 92 \end{bmatrix}$$

i. Elemen (baris 1, kolom 3):

$$(0E \cdot 2E) \oplus (0B \cdot 38) \oplus (0D \cdot AD) \oplus (09 \cdot 50) = 6F$$

ii. Elemen (baris 2, kolom 3):

$$(09 \cdot 2E) \oplus (0E \cdot 38) \oplus (0B \cdot AD) \oplus (0D \cdot 50) = 0C$$

iii. Elemen (baris 3, kolom 3):

$$(0D \cdot 2E) \oplus (09 \cdot 38) \oplus (0E \cdot AD) \oplus (0B \cdot 50) = E7$$

iv. Elemen (baris 4, kolom 3):

$$(0B \cdot 2E) \oplus (0D \cdot 38) \oplus (09 \cdot AD) \oplus (0E \cdot 50) = 92$$

Operasi *InvMixColumns* pada kolom keempat:

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} 2E \\ 38 \\ AD \\ 50 \end{bmatrix} = \begin{bmatrix} 6F \\ 0C \\ E7 \\ 92 \end{bmatrix}$$

i. Elemen (baris 1, kolom 4):

$$(0E \cdot E2) \oplus (0B \cdot 2B) \oplus (0D \cdot 05) \oplus (09 \cdot 2A) = 55$$

ii. Elemen (baris 2, kolom 4):

$$(09 \cdot E2) \oplus (0E \cdot 2B) \oplus (0B \cdot 05) \oplus (0D \cdot 2A) = B1$$

iii. Elemen (baris 3, kolom 4):

$$(0D \cdot E2) \oplus (09 \cdot 2B) \oplus (0E \cdot 05) \oplus (0B \cdot 2A) = 3F$$

iv. Elemen (baris 4, kolom 4):

$$(0B \cdot E2) \oplus (0D \cdot 2B) \oplus (09 \cdot 05) \oplus (0E \cdot 2A) = 08$$

Hasil akhir transformasi *MixColumns* secara keseluruhan adalah:

$$S_5 = \begin{bmatrix} BA & F7 & 6F & 55 \\ C8 & 5C & 0C & B1 \\ AA & 2E & E7 & 3F \\ A6 & A3 & 92 & 08 \end{bmatrix}$$

4. Mengulangi semua proses sampai putaran tercapai

Putaran pertama: $S_5 = P_1 = \begin{bmatrix} BA & F7 & 6F & 55 \\ C8 & 5C & 0C & B1 \\ AA & 2E & E7 & 3F \\ A6 & A3 & 92 & 08 \end{bmatrix}$

Putaran kedua: $P_2 = \begin{bmatrix} 4D & 9C & 82 & 5A \\ 91 & E1 & 06 & 3F \\ A7 & 33 & 5B & C4 \\ 29 & D0 & 7E & F1 \end{bmatrix}$

Putaran ketiga: $P_3 = \begin{bmatrix} 72 & 6A & 1B & 54 \\ 3E & 88 & 0F & A9 \\ C1 & 45 & 97 & 62 \\ DD & 2C & E3 & 10 \end{bmatrix}$

Putaran keempat: $P_4 = \begin{bmatrix} 18 & D4 & 3A & 7F \\ 5B & 2E & 91 & C6 \\ AF & 60 & 44 & 2D \\ 73 & 8A & B9 & E5 \end{bmatrix}$

Putaran kelima: $P_5 = \begin{bmatrix} 9E & 05 & 6C & 21 \\ 47 & F3 & 8D & 9A \\ DA & 11 & 70 & 5C \\ 2B & 64 & CE & B8 \end{bmatrix}$

Putaran keenam: $P_6 = \begin{bmatrix} 63 & 7B & 4F & 98 \\ B4 & 2A & D6 & 40 \\ 1C & E8 & 35 & A1 \\ F0 & 59 & 72 & 0D \end{bmatrix}$

Putaran ketujuh: $P_7 = \begin{bmatrix} 2A & C1 & 95 & 6E \\ 8F & 14 & 73 & D2 \\ 56 & AB & 0E & 47 \\ 39 & F6 & C8 & B0 \end{bmatrix}$

Putaran kedelapan: $P_8 = \begin{bmatrix} D1 & 3E & 28 & 84 \\ 6A & 9F & 5C & 11 \\ 07 & 52 & B3 & 9D \\ E4 & 70 & AF & 36 \end{bmatrix}$

Putaran kesembilan: $P_9 = \begin{bmatrix} 8C & 04 & A7 & 8D \\ 17 & 6F & 37 & EE \\ 93 & F7 & 17 & AC \\ 41 & 9D & 07 & 99 \end{bmatrix}$

5. Pada putaran terakhir tanpa melakukan *InvMixColumns*, hanya melakukan *InvShiftRows*, *InvSubBytes* dan *InvAddRoundKey*.

Putaran terakhir: $P_{10} = \begin{bmatrix} 54 & 4E & 50 & 05 \\ 45 & 4B & 53 & 05 \\ 52 & 52 & 49 & 05 \\ 45 & 49 & 05 & 05 \end{bmatrix}$

Berdasarkan hasil pengujian proses dekripsi menggunakan algoritma *Advanced Encryption Standard* (AES) 128-bit, dilakukan verifikasi terhadap kesesuaian antara *plaintext* awal dengan *plaintext* hasil dekripsi. Proses dekripsi diawali dengan penggunaan *ciphertext* akhir hasil enkripsi, yaitu:

E55C557756374BF1C7A62DE2A4A75272

Ciphertext tersebut kemudian diproses melalui tahapan dekripsi AES yang terdiri dari sepuluh putaran (*round*), dengan urutan operasi meliputi *InvAddRoundKey*, *InvShiftRows*, *InvSubBytes*, dan *InvMixColumns*, kecuali pada putaran terakhir yang tidak menggunakan transformasi *InvMixColumns*.

Kunci yang digunakan pada proses dekripsi identik dengan kunci saat enkripsi, yaitu kunci berbasis teks KUNCIPERHITUNGAN yang telah diekspansi menjadi sepuluh *round key* sesuai prosedur *key* schedule AES-128. Kesamaan kunci ini menjadi syarat utama agar proses dekripsi dapat mengembalikan data ke bentuk semula. Setelah seluruh putaran dekripsi dilakukan, diperoleh *state* akhir sebagai berikut:

$$\begin{bmatrix} 54 & 4E & 50 & 05 \\ 45 & 4B & 53 & 05 \\ 52 & 52 & 49 & 05 \\ 45 & 49 & 05 & 05 \end{bmatrix}$$

Apabila *state* tersebut dikonversi ke dalam format linear AES (*column-major order*), diperoleh blok heksadesimal:

54 45 52 45 4E 4B 52 49 50 53 49 05 05 05 05 05

Konversi ke karakter ASCII menghasilkan teks:

TERENKRIPSI

dengan tambahan *padding* 05 sebanyak lima *byte* yang menunjukkan penggunaan skema *padding* PKCS#7 untuk memenuhi ukuran blok AES sebesar 16 *byte*.

Dengan demikian, dapat dinyatakan bahwa mekanisme enkripsi dan dekripsi yang diimplementasikan bersifat reversibel dan valid, dibuktikan dengan kembalinya *ciphertext* ke *plaintext* awal yaitu TERENKRIPSI setelah seluruh tahapan dekripsi diselesaikan.

4.2.4 Dekripsi *Hybrid* AES-RSA

Berikut langkah-langkah proses dekripsi *Hybrid* AES-RSA:

1. Pada tahap awal dekripsi *Hybrid* AES–RSA, sistem menerima dua *input*, yaitu *ciphertext* hasil enkripsi AES dan *cipherkey* yang merupakan kunci AES yang telah dienkripsi menggunakan RSA. Proses dekripsi dimulai dengan mendekripsi *cipherkey* menggunakan kunci privat RSA untuk memperoleh kembali kunci AES (K_{AES}) yang digunakan pada tahap enkripsi.

Tabel 4.9 Hasil Dekripsi RSA untuk Mendapatkan Kunci AES

No	*Cipherkey* (hex)	Kunci AES Hasil Dekripsi (hex)	Status
1	5d91248952b45d2e9ff4c40f04d923e8936522e2ce1ad47e812c3a9bd06aecdd939453b747753f28eea9fac91e8d50060541b2a59f42977608c176040a82a75ae307bb93ff8469b3d7fa9cf766f5f989d86df034163db9c6b0485f1300b48375d077a4da33d9b63855ac4004f87c286eca0280b6befbb24f9c3a15c93c633a01b5a554e50374f881bff685aaf721a2d78277574f34fb09860a3a20383a21ffc447d6e99bf15deebc22e5161533c0a1fe7e48b82f1ce1c82b9f1bf57e32200c859f1c77263f0a3a2591c9e22a0785ef9d9f3d50c057c649309b3aa3e78bfed2882525e351c7d91845289c81faa77776b46e9e3d1be7e3c70f39a98024dad9a3d6	18c3e440e54bab24720e41a40ae1bf99	Valid
2	6aed7d96a6c36d17b13d63f812b4783c9c8617afbedb0c13e89f0f1d13daf3	c188c0ac04ac51cb5b97dca3d7bfef7e	Valid

No	*Cipherkey* (hex)	Kunci AES Hasil Dekripsi (hex)	Status
	74265f5a49a4a36ed82f25d701ddc0f 7a8890885db8c79c3b216eaf41df47a 844e0ccde362f763ae6e7d85941fcadf 569579eb5acc0b81967a222acc9428a 57c5bee2acb6d52bb7bb2c00042383 4d23617fe22274c13af50a912c1010a aeb756b74d6cbe4981bc487e41ff9f9 b5604725719d368dac6fe7a59d04a9 5bc83f5c554bb10e3f51ae7a4cd6b33 7751935d92555f96228e0c2a97e02b e330e2ac4b2db64e30b3ff4c94972f5 928eba30941736deba8905bf6d86de d7811d368fd87ebd4b288322ef364ce 6a4f7e2dc39331170481a37a9ef37f8 0535b515270ff0ef536		
3	adee8bb936f7538108aa6359f1e11be dacdd7f15444b3a02efda79222f6acb 594f222271dc4f4be99cf3dce81215f 324b6ab1d5218221947bdc5086283a 7bad0d8636178780b0befd7c135c0d 58c4d8c01fa62dd936e8011f94543a4 022d7936edc07d579c6b9b89a27a95 ae13b8f3d722e587ff94e46694084e0 b59b79326783d4925674cba2fc5cbc 8a8b0163ee2d4d6612e295cf5f3e04a 4b5e73f4c83c900b5dd09df6a4d8461 1f48a95b6c36962d630fa24342398a6 97db7fae61364039f6ada272ec8abdd 714dc636717de7afca318bed81f3824 0c9369eff4ead300e90669fab55c6c78	ea7858662c462b74b 5bf9ad01365d8db	Valid

No	*Cipherkey* (hex)	Kunci AES Hasil Dekripsi (hex)	Status
	7b41bfd92b41d5cf37ee7115f82f6a94cdd047acd41b9a6bc1		
4	0078eb2db60a19f52fd6b920d4a24bde3e80470025551ef182c1886e0e7c8693867a1a32f1dc3db047a53a694c262c3361ff685c3337749b4b1af5d9457d486fe71795688d617cedc119d97bf7463f1577ac79e8b34150c7fb53748625341fde15df454d46a8b7bf6425dec82ef861d5956ff57e89b7e41e205b773d007d00d276d6cc10f78049cb4b4594adeb5104fc4178a6fd1148521479a482689c14d356bdd5de12bc6865be86ea727f2f9eb7004155bde078e63f680576878c5cc1614d079972e4480af9a5efb3661b31ccc24d1a03dd6fb850b8bd4eedf4e1e9fc40e88388b4ed0f7fecd1d531b6d7b3eac288638331881bc0076b1de2b5017ed09ab0	5b4c309a41f3ec75a8cdf2bb243162fb	valid
5	4e8fb2f11ba5b601354e3d9a268306c2b5be0c291f350ecbdfaf6dccdf5bac1604292ac75b4b219ec444423d05913bc586576ee3c0077c7ba4a9df579d639d09092b06e31362473eb631b683ef327429b942fb8d5d6f3ab04f6a55c665f6191e128020ad2edbad1211d9288ab4a8f4d693bc811f7c2e879f46df0dd88bb8b9870ea4f2df32065984e7f1a360ca00d5ccccc5d15b21317a9d6a7e5a211b41477aaed7e039c372aea8ec93	0ec61193e379fc057b58ef418a8bd98c	Valid

No	*Cipherkey* (hex)	Kunci AES Hasil Dekripsi (hex)	Status
	6c9a95c531948af734fe9975397f6329f5c52688b3941a7d84995f7ba1b6650c2b24511fbcfa67c370e18d64b3406804d3c4f081da332625b3afdbc38522f5a179557207737dcf100e78f08949a997de63bf1c9094e8		

Berdasarkan Tabel 4.9, seluruh *cipherkey* yang diuji berhasil didekripsi menggunakan RSA dan menghasilkan kembali kunci AES dalam format heksadesimal. Nilai kunci AES hasil dekripsi pada setiap percobaan sesuai dengan kunci AES yang digunakan pada tahap enkripsi sebelumnya, dengan status seluruhnya bernilai "Valid". Panjang kunci AES yang diperoleh adalah 16 *byte* (128 bit), yang menunjukkan kesesuaian dengan spesifikasi AES-128. Hasil ini menegaskan bahwa proses dekripsi RSA dalam skema *Hybrid* mampu memulihkan kunci AES secara tepat sehingga dapat digunakan pada tahap dekripsi pesan selanjutnya.

2. Setelah kunci AES berhasil diperoleh kembali melalui proses dekripsi RSA, tahap selanjutnya adalah mendekripsi *ciphertext* menggunakan kunci AES tersebut. Proses ini bertujuan untuk mengembalikan pesan ke bentuk *plaintext* seperti sebelum dienkripsi.

Tabel 4.10 Hasil Dekripsi AES untuk Mengembalikan *Plaintext*

No	*Ciphertext* (hex)	*Plaintext* Hasil Dekripsi	Status
1	68c32ff3fb53d4175bfd4a9bbfe03ddeca0f8282e3274fcfb31d1f9338b1b4db1926fb56e79926b6317e405f8e6f6a	Pesan ini aman dan TERENKRIPSI dengan sangat baik.	Valid

No	*Ciphertext* (hex)	*Plaintext* Hasil Dekripsi	Status
	24378bc9adf4dc91f2e6b2dddda7e067d7		
2	f68df69d06df5fccc675df08a9edab9a754cae99d9b65220c0361371f21e01735c970aaee057b9c57eaa408d8be26e9cded5c2f7cadbed2abc27002ca976b2f984be0ea378c880b334e880aa7fd6ef00	*Hybrid* AES dan RSA mengamankan pesan teks digital secara efisien aman baik	Valid
3	47ebdf26f62074f02eecdfd118f60df0d037968ddc4a319f352f6439ca64525b83fc011d1a941404aaa71293349b5cd7b53ef71b9ac63dcd80bde89471a28965cb9ad9b5bafa753375e19f68913a8d9aec5030641bc4abedc5f3d9472d610d12e50b995a8678f419b15e7e3d65d612e3	Keamanan pesan teks penting agar data tetap aman, dan sistem enkripsi membantu menjaga privasi anda.	Valid
4	987317889077f4ef4399a9920b2afd9291c256628238550de62d1270ea17f6743c41f78cbbc12397802dbb49c3fcf1818ad5822a09d496dc49eaf8db733a14d574504b143c0900beed17c6dd2cfe0ed35ca14186bac068c8c2b4b975a522d9cb03ca041a25e5bbb3e5f9a50b39183d2ae569f975dc7cdd8a067e4bd9793d54c643996caab0bb2314d2623c8c5f5defa43713fa410701d26cd34d1093d3c71fc7	Sistem *Hybrid* AES dan RSA melindungi pesan teks digital secara aman dan efisien dengan menggabungkan kecepatan AES serta keamanan distribusi kunci RSA	Valid
5	2c1dc11b37421369e6165b9411fc497a0e654047bb193ab11dadcbc981ee9d4c341f0c83de417103fdea98bd8dfbf	*Hybrid* AES dan RSA meningkatkan keamanan pesan teks	Valid

No	*Ciphertext* (hex)	*Plaintext* Hasil Dekripsi	Status
	b11945b104e5426b9ea78ff8a8bff179 798c3bb0b78915ca23bf1db4904bae 1f53ab4e8882488d6662e58244a6fda 721a61ae52269e11b3f39a75e0601d8 44f3f8da82e300dee50956d8f7f8710 05e7f7fe0ab193239cb3a8a15f4240d 88395019fafc67d8aa91e337d031710 8ae625f438fc6b40a058f9e0cbde02d bd19f0dccf03dbf14d2af5c447eeb5bf cedde5dd5511146e7ea11ef3e274926 403b707cd10d	dengan memadukan kecepatan enkripsi AES dan kekuatan RSA sehingga distribusi kunci tetap aman serta proses komunikasi berlangsung lebih terlindungi.	

Berdasarkan Tabel 4.10, seluruh *ciphertext* berhasil didekripsi menggunakan kunci AES hasil dekripsi RSA dan menghasilkan *plaintext* yang identik dengan pesan awal. Hal ini terlihat dari kesesuaian isi *plaintext* pada setiap percobaan serta status dekripsi yang seluruhnya bernilai “Valid”. Hasil tersebut menunjukkan bahwa kunci AES yang diperoleh dari proses dekripsi RSA dapat digunakan dengan tepat pada tahap dekripsi AES, sehingga pesan dapat dipulihkan kembali tanpa perubahan data.

4.3 Evaluasi Waktu Komputasi Proses Enkripsi dan Dekripsi

Pada tahap evaluasi waktu komputasi, pengukuran dilakukan terhadap proses enkripsi dan dekripsi menggunakan tiga skema algoritma, yaitu AES tunggal, RSA tunggal, dan Hybrid AES–RSA. Dalam penelitian ini, algoritma AES yang digunakan adalah AES-128 dengan panjang kunci 128 bit yang bersifat simetris, di mana kunci yang sama digunakan pada proses enkripsi dan dekripsi. Pada algoritma RSA menggunakan pasangan kunci asimetris, yaitu kunci publik

untuk proses enkripsi dan kunci privat untuk proses dekripsi, dengan nilai modulus $n = p \times q$ yang dibangkitkan dari dua bilangan prima

Sementara itu, pada skema *Hybrid* AES–RSA, kunci simetris AES-128 yang digunakan untuk mengenkripsi pesan teks terlebih dahulu dibangkitkan secara acak, kemudian kunci tersebut dienkripsi menggunakan kunci publik RSA sebelum dikirimkan. Dengan demikian, evaluasi waktu komputasi dalam penelitian ini mencakup keseluruhan proses, mulai dari pembangkitan kunci, enkripsi, hingga dekripsi pada masing-masing algoritma. Nilai waktu komputasi yang disajikan merupakan rata-rata dari 50 sesi tersebut.

4.3.1 AES Tunggal (50 Kali Pengulangan)

Tabel 4.11 Waktu Rata-Rata AES Tunggal

No	Panjang Pesan	Enkripsi	Dekripsi
1	50 karakter	0.000096 detik	0.000061 detik
2	75 karakter	0.000075 detik	0.000041 detik
3	100 karakter	0.000064 detik	0.000047 detik
4	150 karakter	0.000075 detik	0.000077 detik
5	200 karakter	0.000068 detik	0.000044 detik

Berdasarkan hasil pengujian pada Tabel 4.11, algoritma AES tunggal mampu menjalankan proses enkripsi dan dekripsi pada seluruh variasi panjang pesan dengan waktu komputasi yang relatif kecil. Waktu enkripsi menunjukkan kecenderungan meningkat seiring bertambahnya panjang pesan. Sementara itu, waktu dekripsi berada pada kisaran yang relatif stabil meskipun terdapat fluktuasi kecil pada beberapa variasi data. Perbedaan nilai tersebut dipengaruhi oleh faktor eksekusi sistem, seperti manajemen memori, beban prosesor, serta proses latar belakang saat pengujian berlangsung, sehingga waktu yang diperoleh tidak selalu meningkat secara linier terhadap panjang pesan. Secara keseluruhan, proses

enkripsi dan dekripsi AES tunggal berjalan dengan baik pada seluruh pengujian yang dilakukan.

4.3.2 RSA Tunggal (50 Kali Pengulangan)

Tabel 4.12 Waktu Rata-Rata RSA Tunggal

No	Panjang Pesan	Enkripsi	Dekripsi
1	50 karakter	0.001227 detik	0.048863 detik
2	75 karakter	0.001357 detik	0.049731 detik
3	100 karakter	0.001268 detik	0.049054 detik
4	150 karakter	0.001250 detik	0.050240 detik
5	200 karakter	Gagal	Gagal

Berdasarkan Tabel 4.12, waktu komputasi enkripsi RSA tunggal pada panjang pesan 50 hingga 200 karakter berada pada kisaran 0.001227–0.001357 detik, sedangkan waktu dekripsi berada pada kisaran 0.048863–0.050240 detik. Waktu dekripsi secara konsisten lebih besar dibandingkan waktu enkripsi pada seluruh variasi pesan yang berhasil diproses.

Nilai waktu yang diperoleh tidak menunjukkan peningkatan yang sepenuhnya linier terhadap panjang pesan. Sebagai contoh, waktu enkripsi pada 150 karakter (0.001250 detik) lebih kecil dibandingkan 100 karakter (0.001268 detik), dan nilai pada 100 serta 150 karakter berada pada rentang yang relatif sama. Fluktuasi kecil ini dipengaruhi oleh kondisi eksekusi sistem saat pengujian berlangsung, seperti manajemen memori, beban prosesor, serta proses latar belakang, sehingga meskipun pengujian dilakukan berulang, waktu yang dihasilkan tidak selalu meningkat secara berurutan. Pada panjang pesan 200 karakter, proses enkripsi dan dekripsi tidak berhasil dilakukan sehingga tidak diperoleh nilai waktu komputasi pada pengujian tersebut.

4.3.3 *Hybrid* AES-RSA (50 Kali Pengulangan)

Tabel 4.13 Waktu Rata-Rata *Hybrid* AES-RSA

No	Panjang Pesan	Enkripsi	Dekripsi
1	50 karakter	0.001311 detik	0.001724 detik
2	75 karakter	0.000899 detik	0.002503 detik
3	100 karakter	0.000952 detik	0.001733 detik
4	150 karakter	0.001511 detik	0.001672 detik
5	200 karakter	0.001397 detik	0.001698 detik

Berdasarkan Tabel 4.13, waktu komputasi enkripsi *Hybrid* AES–RSA berada pada kisaran 0.000899–0.001511 detik, sedangkan waktu dekripsi berada pada kisaran 0.001672–0.002503 detik. Waktu dekripsi tercatat lebih besar dibandingkan waktu enkripsi pada seluruh variasi panjang pesan.

Nilai waktu enkripsi tidak menunjukkan peningkatan yang sepenuhnya linier terhadap panjang pesan. Sebagai contoh, waktu enkripsi pada 75 karakter (0.000899 detik) lebih kecil dibandingkan 50 karakter, sedangkan pada 100 karakter meningkat kembali menjadi 0.000952 detik. Fluktuasi ini dipengaruhi oleh kondisi eksekusi sistem saat pengujian, seperti beban prosesor dan manajemen memori, sehingga waktu yang dihasilkan tidak selalu meningkat secara berurutan.

Pada proses dekripsi, waktu komputasi menunjukkan rentang yang lebih besar, dengan nilai tertinggi pada panjang pesan 75 karakter sebesar 0.002503 detik. Meskipun demikian, seluruh variasi panjang pesan, termasuk 200 karakter, berhasil diproses tanpa kegagalan

4.4 Analisis Statistik Waktu Komputasi

Untuk memperkuat hasil perbandingan secara kuantitatif, dilakukan analisis statistik menggunakan perangkat lunak SPSS. Hasil uji normalitas waktu komputasi ditunjukkan pada Tabel 4.14.

Tabel 4.14 Hasil Uji Normalitas Waktu Komputasi

Algoritma	Metode Uji	Nilai Signifikan	Keputusan
AES Tunggal	Shapiro-Wilk	< 0,05	Tidak Berdistribusi Normal
RSA Tunggal	Shapiro-Wilk	< 0,05	Tidak Berdistribusi Normal
Hybrid AES-RSA	Shapiro-Wilk	< 0,05	Tidak Berdistribusi Normal

Berdasarkan hasil uji normalitas pada Tabel 4.14, seluruh data waktu komputasi memiliki nilai signifikansi kurang dari 0.05. Hal ini menunjukkan bahwa data tidak berdistribusi normal, sehingga uji parametrik tidak memenuhi asumsi untuk digunakan. Oleh karena itu, analisis perbandingan dilanjutkan menggunakan uji statistik nonparametrik *Kruskal-Wallis*.

Tabel 4.15 Hasil Uji *Kruskal-Wallis* Waktu Komputasi

Karakter	Proses	Chi-Square	df	Sig.
50	Enkripsi	95.537	2	0.000
50	Dekripsi	132.460	2	0.000
75	Enkripsi	120.481	2	0.000
75	Dekripsi	132.472	2	0.000
100	Enkripsi	118.890	2	0.000
100	Dekripsi	132.457	2	0.000
150	Enkripsi	117.738	2	0.000
150	Dekripsi	132.475	2	0.000

Berdasarkan Tabel 4.15, seluruh nilai signifikansi lebih kecil dari 0.05. Hal ini menunjukkan bahwa terdapat perbedaan waktu komputasi yang signifikan secara statistik di antara kelompok algoritma pada setiap variasi panjang karakter dan proses. Namun, uji *Kruskal–Wallis* hanya menunjukkan adanya perbedaan pada minimal satu kelompok, sehingga diperlukan uji lanjut untuk mengetahui pasangan algoritma yang berbeda secara signifikan. Pada panjang pesan 200 karakter, algoritma RSA tidak disertakan karena tidak menghasilkan waktu komputasi yang

valid. Oleh karena itu, analisis perbandingan pada kondisi tersebut dilakukan menggunakan uji *Mann–Whitney* yang sesuai untuk dua kelompok independen, sehingga tidak dimasukkan dalam pengujian *Kruskal–Wallis*.

Tabel 4.16 *Mean rank* Waktu Komputasi

Karakter	Proses	AES	RSA	*Hybrid*
50	Enkripsi	27.48	91.12	107.90
75	Enkripsi	27.12	122.46	76.92
100	Enkripsi	25.50	119.65	81.35
150	Enkripsi	25.50	81.94	119.06
200	Enkripsi	25.50	-	75.50
50	Dekripsi	25.50	125.50	75.50
75	Dekripsi	25.50	125.50	75.50
100	Dekripsi	25.50	125.50	75.50
150	Dekripsi	25.50	125.50	75.50
200	Dekripsi	25.50	-	75.50

Berdasarkan nilai *mean rank* pada Tabel 4.16, dapat dijelaskan bahwa algoritma AES menunjukkan waktu komputasi paling cepat pada seluruh skenario pengujian, karena secara konsisten memperoleh nilai rata-rata peringkat yang paling rendah dibandingkan algoritma lainnya. Sementara itu, algoritma *Hybrid* AES–RSA dan RSA menunjukkan variasi nilai *mean rank* pada setiap kondisi pengujian, di mana pada beberapa kondisi algoritma *Hybrid* memiliki waktu komputasi lebih cepat dibandingkan RSA, namun pada kondisi lainnya justru lebih lambat. Hal ini menunjukkan bahwa performa kedua algoritma tersebut relatif fluktuatif tergantung pada panjang pesan dan proses yang dilakukan. Pada panjang pesan 200 karakter, algoritma RSA tidak dapat dianalisis karena tidak menghasilkan waktu komputasi yang valid.

Pengujian lanjutan menggunakan uji *Mann–Whitney* dilakukan untuk mengetahui pasangan algoritma mana yang memiliki perbedaan waktu komputasi secara signifikan setelah sebelumnya terdeteksi perbedaan pada uji *Kruskal–Wallis*. Pengujian ini menggunakan koreksi *Bonferroni* dengan tingkat signifikansi $\alpha = 0.0167$, yang diperoleh dari pembagian $\alpha = 0.05$ terhadap jumlah pasangan yang dibandingkan.

Berdasarkan Tabel 4.17, seluruh pasangan algoritma menghasilkan nilai *p-value* sebesar 0.000. Nilai *p-value* sebesar 0.000 tersebut tidak menunjukkan nol mutlak, melainkan probabilitas yang sangat kecil (< 0.001) akibat pembulatan *output* perangkat lunak statistik. Karena seluruh *p-value* lebih kecil dari batas signifikansi 0.0167, maka setiap pasangan algoritma dinyatakan memiliki perbedaan waktu komputasi yang signifikan secara statistik, baik pada proses enkripsi maupun dekripsi di setiap variasi panjang pesan.

Pada panjang pesan 200 karakter, algoritma RSA tidak disertakan dalam pengujian karena tidak menghasilkan waktu komputasi yang valid pada tahap sebelumnya. Oleh karena itu, analisis perbandingan pada kondisi tersebut dilakukan menggunakan uji *Mann–Whitney* terhadap dua kelompok, yaitu AES dan *Hybrid* AES–RSA. Hasil pengujian menunjukkan nilai p-value yang lebih kecil dari $\alpha = 0.0167$, sehingga perbedaan antara kedua algoritma tersebut dinyatakan signifikan secara statistik.

Tabel 4.17 Hasil Uji Lanjutan

Karakter	Proses	Pasangan	*p-value*	Keputusan
50	Enkripsi	AES vs RSA	0.000	Signifikan
50	Enkripsi	AES vs *Hybrid* AES-RSA	0.000	Signifikan
50	Enkripsi	RSA vs *Hybrid* AES-RSA	0.004	Signifikan

Karakter	Proses	Pasangan	*p-value*	Keputusan
50	Dekripsi	AES vs RSA	0.000	Signifikan
50	Dekripsi	AES vs *Hybrid* AES-RSA	0.000	Signifikan
50	Dekripsi	RSA vs *Hybrid* AES-RSA	0.000	Signifikan
75	Enkripsi	AES vs RSA	0.000	Signifikan
75	Enkripsi	AES vs *Hybrid* AES-RSA	0.000	Signifikan
75	Enkripsi	RSA vs *Hybrid* AES-RSA	0.000	Signifikan
75	Dekripsi	AES vs RSA	0.000	Signifikan
75	Dekripsi	AES vs *Hybrid* AES-RSA	0.000	Signifikan
75	Dekripsi	RSA vs *Hybrid* AES-RSA	0.000	Signifikan
100	Enkripsi	AES vs RSA	0.000	Signifikan
100	Enkripsi	AES vs *Hybrid* AES-RSA	0.000	Signifikan
100	Enkripsi	RSA vs *Hybrid* AES-RSA	0.000	Signifikan
100	Dekripsi	AES vs RSA	0.000	Signifikan
100	Dekripsi	AES vs *Hybrid* AES-RSA	0.000	Signifikan
100	Dekripsi	RSA vs *Hybrid* AES-RSA	0.000	Signifikan
150	Enkripsi	AES vs RSA	0.000	Signifikan
150	Enkripsi	AES vs *Hybrid* AES-RSA	0.000	Signifikan
150	Enkripsi	RSA vs *Hybrid* AES-RSA	0.000	Signifikan
150	Dekripsi	AES vs RSA	0.000	Signifikan
150	Dekripsi	AES vs *Hybrid* AES-RSA	0.000	Signifikan
150	Dekripsi	RSA vs *Hybrid* AES-RSA	0.000	Signifikan
200	Enkripsi	AES vs *Hybrid* AES-RSA	0.000	Signifikan
200	Dekripsi	AES vs *Hybrid* AES-RSA	0.000	Signifikan

Hasil pengujian statistik membuktikan bahwa AES memiliki efisiensi waktu tertinggi karena menggunakan kriptografi simetris dengan kompleksitas komputasi rendah. RSA menunjukkan waktu komputasi paling tinggi akibat penggunaan operasi perpangkatan modular bilangan besar. Sementara itu, *Hybrid* AES–RSA mampu meningkatkan efisiensi dibanding RSA murni karena hanya menggunakan

RSA pada proses pengamanan kunci. Temuan ini menegaskan bahwa kombinasi algoritma *Hybrid* memberikan kompromi antara tingkat keamanan dan efisiensi waktu komputasi, khususnya pada pengamanan pesan teks berukuran besar.

4.5 Evaluasi Keamanan dan Kelayakan Algoritma *Hybrid* AES-RSA

Dari sisi keamanan, algoritma *Hybrid* AES–RSA memiliki keunggulan dibandingkan algoritma tunggal. AES tunggal mampu mengenkripsi dan mendekripsi pesan teks berukuran panjang dengan waktu komputasi yang relatif singkat, namun memiliki kelemahan pada distribusi kunci. RSA tunggal aman dalam pengelolaan kunci, tetapi kurang sesuai digunakan untuk mengenkripsi pesan teks berukuran panjang karena membutuhkan waktu komputasi yang lebih besar.

Algoritma *Hybrid* AES–RSA menggabungkan keunggulan kedua algoritma tersebut, yaitu kemampuan AES dalam menangani pesan panjang dan keamanan RSA dalam melindungi kunci kriptografi. Kunci AES tidak dikirim secara langsung, melainkan dienkripsi menggunakan RSA, sehingga hanya pihak yang memiliki kunci privat yang dapat mengakses pesan.

Peningkatan waktu komputasi pada algoritma *Hybrid* AES–RSA dipengaruhi oleh adanya proses tambahan berupa enkripsi dan dekripsi kunci AES menggunakan RSA. Penggabungan dua algoritma dalam satu skema kriptografi menghasilkan konsekuensi terhadap performa waktu komputasi dibandingkan penggunaan algoritma tunggal.

Berdasarkan hasil perbandingan waktu komputasi, analisis statistik menggunakan SPSS, serta evaluasi aspek keamanan, dapat disimpulkan bahwa algoritma *Hybrid* AES–RSA merupakan pendekatan yang paling seimbang antara tingkat keamanan dan waktu komputasi dalam pengamanan pesan teks.

4.6 Integrasi Nilai-Nilai Islam dalam Keamanan Informasi

Integrasi nilai-nilai Islam dalam penelitian ini bertujuan untuk menunjukkan bahwa penerapan sistem keamanan informasi tidak hanya relevan secara teknis, tetapi juga selaras dengan prinsip etika dan moral dalam Islam. Analisis integrasi dilakukan dengan mengaitkan nilai-nilai Al-Qur'an dengan hasil simulasi enkripsi–dekripsi serta evaluasi waktu komputasi pada sistem kriptografi *Hybrid* AES–RSA.

QS. An-Nisa ayat 58 menegaskan kewajiban menjaga dan menyampaikan amanah kepada pihak yang berhak. Dalam konteks keamanan informasi, amanah dapat dimaknai sebagai tanggung jawab sistem dalam melindungi kerahasiaan dan keutuhan pesan teks yang dipercayakan oleh pengguna. Hasil simulasi menunjukkan bahwa pesan teks yang diproses menggunakan skema *Hybrid* AES–RSA berhasil diubah dari bentuk *plaintext* menjadi *ciphertext* yang tidak dapat dibaca tanpa kunci yang sah. Proses dekripsi hanya dapat dilakukan dengan menggunakan kunci privat RSA dan kunci AES yang sesuai, sehingga akses terhadap pesan dibatasi hanya kepada pihak yang memiliki otorisasi.

Selain itu, hasil evaluasi waktu komputasi menunjukkan bahwa proses enkripsi dan dekripsi dapat dilakukan dengan waktu komputasi yang relatif singkat dan stabil. Kondisi ini penting untuk menjaga keandalan sistem dalam komunikasi pesan teks, karena waktu pemrosesan yang terlalu lama berpotensi menurunkan tingkat kepercayaan pengguna. Dengan karakteristik waktu pemrosesan yang terukur dan stabil, sistem yang dihasilkan tidak hanya aman secara kriptografis, tetapi juga layak digunakan dalam praktik komunikasi digital. Dengan demikian, sistem *Hybrid* AES–RSA yang diimplementasikan mampu menjaga kerahasiaan

dan keutuhan pesan teks, sehingga amanah digital pengguna dapat terlindungi sesuai dengan prinsip amanah dalam QS. An-Nisa ayat 58.

Di sisi lain, QS. Ar-Ra'd ayat 11 menekankan bahwa perubahan suatu keadaan bergantung pada usaha yang dilakukan. Prinsip ini relevan dalam pengembangan sistem keamanan informasi yang menuntut adanya peningkatan kualitas dan metode pengamanan seiring dengan perkembangan teknologi. Hasil penelitian menunjukkan bahwa penggunaan satu algoritma kriptografi saja memiliki keterbatasan, baik dari sisi waktu komputasi maupun keamanan distribusi kunci. AES unggul dalam kecepatan pemrosesan data, namun memiliki kelemahan pada aspek distribusi kunci, sedangkan RSA memiliki tingkat keamanan distribusi kunci yang baik tetapi membutuhkan waktu komputasi yang lebih besar.

Pendekatan *Hybrid* AES–RSA yang diterapkan dalam penelitian ini merupakan bentuk upaya perbaikan sistem dengan mengombinasikan keunggulan kedua algoritma tersebut. Hal ini diperkuat oleh hasil analisis perbandingan waktu komputasi, di mana sistem *Hybrid* menunjukkan keseimbangan antara waktu pemrosesan dan peningkatan tingkat keamanan dibandingkan penggunaan algoritma tunggal. Dengan demikian, penerapan metode *Hybrid* mencerminkan adanya ikhtiar dan perbaikan berkelanjutan dalam menghasilkan sistem keamanan informasi yang lebih andal, sebagaimana nilai yang terkandung dalam QS. Ar-Ra'd ayat 11.

Berdasarkan hasil pembahasan tersebut, dapat disimpulkan bahwa sistem kriptografi *Hybrid* AES–RSA yang dikembangkan tidak hanya efektif secara teknis dan empiris, tetapi juga selaras dengan nilai-nilai Islam. Perlindungan pesan teks mencerminkan nilai amanah, sementara penerapan metode *Hybrid* menunjukkan

upaya perbaikan sistem keamanan informasi secara berkelanjutan, sehingga teknologi yang dikembangkan tidak hanya berfungsi sebagai alat teknis, tetapi juga sebagai sarana menjaga kepercayaan, tanggung jawab, dan keadilan dalam komunikasi digital.

Berdasarkan hasil evaluasi pada penelitian ini, penerapan algoritma *Hybrid* AES–RSA terbukti mampu meningkatkan keamanan pesan teks melalui enkripsi berlapis dan mekanisme distribusi kunci yang lebih aman. Sistem ini menjaga kerahasiaan informasi dari risiko akses tidak sah, penyadapan, maupun manipulasi data selama proses transmisi. Perlindungan terhadap privasi dan kerahasiaan informasi tersebut sejalan dengan nilai-nilai Islam yang melarang tindakan pengintaian atau pengaksesan informasi tanpa hak. Allah SWT berfirman dalam QS. Al-Ḥujurāt ayat 12:

يَا أَيُّهَا الَّذِينَ آمَنُوا اجْتَنِبُوا كَثِيرًا مِّنَ الظَّنِّ إِنَّ بَعْضَ الظَّنِّ إِثْمٌ ۖ وَلَا تَجَسَّسُوا وَلَا يَغْتَب بَّعْضُكُم بَعْضًا

"Wahai orang-orang yang beriman, jauhilah banyak dari prasangka, sesungguhnya sebagian prasangka itu dosa. Dan janganlah kamu memata-matai…"

Larangan *tajassus* (memata-matai) pada ayat tersebut dapat dimaknai sebagai larangan mengakses atau menyadap informasi secara tersembunyi tanpa izin. Dengan demikian, implementasi kriptografi *Hybrid* AES–RSA dalam penelitian ini tidak hanya layak secara teknis dalam menjaga keamanan pesan teks, tetapi juga mencerminkan penerapan nilai etika Islam dalam melindungi kerahasiaan informasi di era digital.

BAB V
PENUTUP

5.1 Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan melalui proses simulasi enkripsi–dekripsi dan pengukuran waktu komputasi, dapat disimpulkan bahwa:

1. Hasil simulasi menunjukkan bahwa kriptografi *Hybrid* AES-RSA dapat diimplementasikan dengan baik dalam sistem keamanan pesan teks. Algoritma AES digunakan untuk mengenkripsi dan mendekripsi isi pesan, sedangkan algoritma RSA digunakan untuk mengamankan kunci AES. Simulasi yang dilakukan membuktikan bahwa sistem mampu menjalankan proses enkripsi dan dekripsi pesan teks secara fungsional sesuai dengan perancangan sistem yang telah ditetapkan, sehingga pesan dapat diamankan dan dikembalikan ke bentuk semula tanpa perubahan isi.
2. Berdasarkan hasil pengujian dan evaluasi waktu komputasi, sistem keamanan pesan teks berbasis kriptografi *Hybrid* AES-RSA membutuhkan waktu enkripsi dan dekripsi yang lebih besar dibandingkan AES tunggal maupun RSA tunggal. Peningkatan waktu komputasi ini disebabkan oleh adanya proses tambahan berupa enkripsi dan dekripsi kunci AES menggunakan algoritma RSA. Hasil uji statistik nonparametrik menggunakan uji Kruskal–Wallis menunjukkan bahwa terdapat perbedaan waktu komputasi yang signifikan secara statistik antara AES tunggal, RSA tunggal, dan *Hybrid* AES–RSA. Hal ini menunjukkan bahwa pendekatan kriptografi *Hybrid* AES-RSA memiliki karakteristik waktu komputasi tersendiri sebagai konsekuensi dari penggabungan proses enkripsi data dan pengamanan kunci.

5.2 Saran

Berdasarkan hasil penelitian yang telah dilakukan, berikut beberapa saran yang dapat dipertimbangkan untuk penelitian lanjutan maupun pengembangan sistem:

1. Menerapkan skema kriptografi *Hybrid* AES–RSA pada sistem pesan teks berbasis client–server atau aplikasi nyata, sehingga performa sistem dapat dievaluasi dalam kondisi penggunaan sebenarnya dan tidak terbatas pada lingkungan simulasi perangkat lunak.
2. Menambahkan parameter pengujian selain waktu komputasi, seperti konsumsi memori dan penggunaan CPU, sehingga evaluasi performa sistem dapat dilakukan secara lebih menyeluruh.
3. Menguji penggunaan mode dan parameter kriptografi yang berbeda, seperti AES-GCM atau variasi panjang kunci (AES-192, AES-256, dan RSA-3072), untuk menganalisis perbandingan antara peningkatan tingkat keamanan dan dampaknya terhadap waktu pemrosesan.

DAFTAR PUSTAKA

Akter, R., Khan, M. A. R., Rahman, F., Soheli, S. J., & Suha, N. J. (2023). RSA and AES Based *Hybrid* Encryption Technique for Enhancing Data Security in Cloud Computing. *International Journal of Computational and Applied Mathematics & Computer Science*, *3*, 60–71. https://doi.org/10.37394/232028.2023.3.8

Chang, Q., Ma, T., & Yang, W. (2025). Low power IoT device communication through *Hybrid* AES-RSA encryption in MRA mode. *Scientific Reports*, *15*(1), 1–25. https://doi.org/10.1038/s41598-025-98905-0

Farisi, A., Pradata, A., & Miraswan, K. J. (2023). *Teknologi Blockchain di Perangkat Seluler*. *8*(April), 1166–1171.

Fatonah, Mulyana, D. I., Heryani, A. P., & Khoirunnisa, V. (2022). Implementasi Metode Rivest Shamir Adleman untuk Enkripsi dan Dekripsi Text. *Jurnal Informatika Dan Teknologi Komputer (J-ICOM)*, *3*(1), 32–39. https://doi.org/10.33059/j-icom.v3i1.4990

Hermawan, A., & Ujianto, H. I. E. (2021). Implementasi Enkripsi Data Menggunakan Kombinasi AES dan RSA. *Jurnal Nasional Informatika Dan Teknologi*, *5*(2), 325–330.

Irawan, W. H., Hijriyah, H., & Habibi, A. (2013). *Teori Bilangan untuk Ilmu Komputer.* Jakarta: Mitra Wacana Media.

Khaing Oo, K. K., & Soe, Y. N. (2019). Encryption data measurement and data security of *Hybrid* AES and RSA algorithm. *International Journal of Trend in Scientific Research and Development (IJTSRD), 3*(6), 834–838. https://www.ijtsrd.com/papers/ijtsrd29243.pdf

Lee, C., Lin, J., Prokop, A., Gopalakrishnan, V., Hanna, R. N., Papa, E., Freeman, A., Patel, S., Yu, W., Huhn, M., Sheikh, A. S., Tan, K., Sellman, B. R., Cohen, T., Mangion, J., Khan, F. M., Gusev, Y., & Shameer, K. (2022). StarGazer: A *Hybrid* Intelligence Platform for Drug Target Prioritization and Digital Drug Repositioning Using *Streamlit*. *Frontiers in Genetics*, *13*(May), 1–12. https://doi.org/10.3389/fgene.2022.868015

Mabruri, A. S. (2020). Data Security System of Text Messaging Based on Android Mobile Devices Using Advanced Encrytion Standard Dynamic S-BOX. *Journal of Soft Computing Exploration*, *1*(1), 39–46. https://doi.org/10.52465/joscex.v1i1.10

Menezes, A. J., Van Oorschot, P. C., & Vanstone, S. A. (1996). *Handbook of Applied Cryptography*. CRC Press.

Mitali, A., Ayesha, S., Kaynat, K., Tuika, A., & Kudal, P. (2021). Encryption and Decryption (File Cryptography with AES and RSA for Mobile Based on Android). *International Journal of Computer Science and Mobile Computing*, *10*(4), 132–138. https://doi.org/10.47760/ijcsmc.2021.v10i04.018

Munir, R. (2006). Pengantar Kriptografi. *Bahan Kuliah IF4020*.

National Institute of Standards and Technology. (2001). *Announcing the Advanced Encryption Standard (AES)* (FIPS PUB 197). U.S. Department of Commerce. https://doi.org/10.6028/NIST.FIPS.197

NU Online. (2025). *Beranda Islam Indonesia*. NU Online. https://www.nu.or.id diakses pada 10 September 2025 pukul 20.00 WIB

Putri, H., Virna, L., Febrianti, T., & Sutabri, T. (2025). Pengamanan Data Transmisi Aplikasi WebMenggunakan Algoritma Kriptografi RSA: Studi Kasus

danAnalisis. *Https//Journal.Stiestekom.Ac.Id/Index.Php/Mifortekh*, *5*(1), 153–170.

Rasheed, A. M., & Kumar, R. M. S. (2025). Efficient lightweight cryptographic solutions for enhancing data security in healthcare systems based on IoT. *Frontiers in Computer Science*, *7*(April), 1–22. https://doi.org/10.3389/fcomp.2025.1522184

Republik Indonesia. (2022). *Undang-Undang Nomor 27 Tahun 2022 tentang Perlindungan Data Pribadi.* Jakarta: Kementerian Hukum dan HAM RI. https://peraturan.bpk.go.id/Details/229798/uu-no-27-tahun-2022

Rosen, K. H. (2021). *Elementary number theory and its applications* (7th ed.). Pearson Education.

Santhoshi, G., Kumar, G. N., Kiran, T. U., & Kiran, T. K. (2024). Crypto analysis and visualization using Python and Streamlit. *International Journal of Scientific Development and Research (IJSDR), 9(5), 1534–1539*.

Stallings, W. (2017). *Cryptography and network security: Principles and practice* (7th ed.). Pearson Education.

Stein, W. (2017). *Elementary number theory: Primes, congruences, and secrets.* Open Textbook.

Walle, Y. M. (2024). *Hybrid* RSA–AES-Based Software-Defined Network to Improve the Security of MANET. *Open Information Science*, *8*(1). https://doi.org/10.1515/opis-2024-0001

LAMPIRAN

Lampiran 1 Kode Program Uji AES Tunggal

```
import streamlit as st
from Crypto.Cipher import AES
from Crypto.Random import get_random_bytes
from Crypto.Util.Padding import pad, unpad
import hashlib
import time
import pandas as pd
# ========================
# FUNGSI UTILITAS AES
# ========================
def derive_key_from_password(password: str) -> bytes:
    if not password:
        raise ValueError("Password tidak boleh kosong.")
    sha256_hash = hashlib.sha256(password.encode("utf-8")).digest()
    return sha256_hash[:16]  # 128-bit
# ========================
# ENKRIPSI AES (END-TO-END)
# ========================
def aes_encrypt_cbc(plaintext: str, password: str):
    t_start = time.perf_counter()
    key = derive_key_from_password(password)
    plaintext_bytes = plaintext.encode("utf-8")
    padded = pad(plaintext_bytes, AES.block_size)
    iv_bytes = get_random_bytes(16)
    cipher = AES.new(key, AES.MODE_CBC, iv=iv_bytes)
    ciphertext_bytes = cipher.encrypt(padded)
    t_end = time.perf_counter()
    return ciphertext_bytes.hex(), iv_bytes.hex(), t_end - t_start
    # ========================
# DEKRIPSI AES (END-TO-END)
# ========================
def aes_decrypt_cbc(ciphertext_hex: str, iv_hex: str, password:
str):
    t_start = time.perf_counter()
    key = derive_key_from_password(password)
    ciphertext_bytes = bytes.fromhex(ciphertext_hex)
    iv_bytes = bytes.fromhex(iv_hex)
    cipher = AES.new(key, AES.MODE_CBC, iv=iv_bytes)
    padded_plaintext = cipher.decrypt(ciphertext_bytes)
    plaintext_bytes = unpad(padded_plaintext, AES.block_size)
    plaintext = plaintext_bytes.decode("utf-8")
    t_end = time.perf_counter()
    return plaintext, t_end - t_start
# ========================
# STREAMLIT APP
# ========================
st.set_page_config(
    page_title="AES Tunggal",
    layout="centered"
)
```

```
st.title("AES-128 CBC – Enkripsi, Dekripsi & Pengujian Waktu")
tab_enc, tab_dec = st.tabs(["Enkripsi", "Dekripsi"])
# ========================
# TAB ENKRIPSI
# ========================
with tab_enc:
    st.subheader("🔒 Enkripsi AES")
    plaintext = st.text_area("Masukkan Plaintext", key="plain_enc")
    password_enc = st.text_input(
        "Masukkan Password",
        type="password",
        key="pass_enc"
    )
    N_enc = st.number_input(
        "Jumlah pengujian (N)",
        min_value=1,
        max_value=100,
        value=5,
        key="n_enc"
    )
    if st.button("Enkripsi & Uji", key="btn_enc"):
        if not plaintext:
            st.error("Plaintext tidak boleh kosong.")
        elif not password_enc:
            st.error("Password tidak boleh kosong.")
        else:
            hasil = []
            ciphertext_hex_final = ""
            iv_hex_final = ""
            for i in range(N_enc):
                c_hex, iv_hex, t = aes_encrypt_cbc(plaintext,
password_enc)
                if i == 0:
                    ciphertext_hex_final = c_hex
                    iv_hex_final = iv_hex
                hasil.append({
                    "Pengujian": i + 1,
                    "Waktu (detik)": t
                })
            df = pd.DataFrame(hasil)
            rata2 = df["Waktu (detik)"].mean()
            st.success("Enkripsi selesai")
            st.code(ciphertext_hex_final)
            st.code(iv_hex_final)
            st.dataframe(df)
            st.info(f"Rata-rata waktu: {rata2:.6f} detik")
# ========================
# TAB DEKRIPSI
# ========================
with tab_dec:
    st.subheader("🔓 Dekripsi AES")
    ciphertext_input = st.text_area(
        "Masukkan Ciphertext (hex)",
        key="cipher_dec"
    )
```

```
 iv_input = st.text_input(
        "Masukkan IV (hex)",
        key="iv_dec"
    )
    password_dec = st.text_input(
        "Masukkan Password",
        type="password",
        key="pass_dec"
    )
    N_dec = st.number_input(
        "Jumlah pengujian (N)",
        min_value=1,
        max_value=100,
        value=5,
        key="n_dec"
    )
    if st.button("Dekripsi & Uji", key="btn_dec"):
        if not ciphertext_input or not iv_input or not password_dec:
            st.error("Semua input harus diisi.")
        else:
            hasil = []
            plaintext_final = ""
            for i in range(N_dec):
                p, t = aes_decrypt_cbc(ciphertext_input, iv_input,
password_dec)
                if i == 0:
                    plaintext_final = p
                hasil.append({
                    "Pengujian": i + 1,
                    "Waktu (detik)": t
                })
            df = pd.DataFrame(hasil)
            rata2 = df["Waktu (detik)"].mean()
            st.success("Dekripsi selesai")
            st.code(plaintext_final)
            st.dataframe(df)
            st.info(f"Rata-rata waktu: {rata2:.6f} detik")
```

Lampiran 2 Kode Program Uji RSA Tunggal

```
import streamlit as st
from Crypto.PublicKey import RSA
from Crypto.Cipher import PKCS1_OAEP
from Crypto.Hash import SHA256
import time
import pandas as pd
# =======s=================
# FUNGSI UTILITAS
# ========================
def generate_rsa_keypair(bits: int = 2048):
    key = RSA.generate(bits)
    return key.export_key().decode(),
key.publickey().export_key().decode()
# ========================
# ENKRIPSI RSA (FULL TIME)
# ========================
def rsa_encrypt_oaep(plaintext: str, public_key_pem: str):
    t_start = time.perf_counter()
    public_key = RSA.import_key(public_key_pem)
    cipher = PKCS1_OAEP.new(public_key, hashAlgo=SHA256)
    plaintext_bytes = plaintext.encode("utf-8")
    ciphertext_bytes = cipher.encrypt(plaintext_bytes)
    t_end = time.perf_counter()
    return ciphertext_bytes.hex(), (t_end - t_start)
# ========================
# DEKRIPSI RSA (FULL TIME)
# ========================
def rsa_decrypt_oaep(ciphertext_hex: str, private_key_pem: str):
    t_start = time.perf_counter()
    private_key = RSA.import_key(private_key_pem)
    cipher = PKCS1_OAEP.new(private_key, hashAlgo=SHA256)
    ciphertext_bytes = bytes.fromhex(ciphertext_hex)
    plaintext_bytes = cipher.decrypt(ciphertext_bytes)
    t_end = time.perf_counter()
    return plaintext_bytes.decode("utf-8"), (t_end - t_start)
# ========================
# STREAMLIT APP
# ========================
st.set_page_config(page_title="RSA Tunggal – Uji Waktu")
st.title("RSA Tunggal (OAEP-SHA256) – Enkripsi, Dekripsi & Pengujian
Waktu")
# ========================
# GENERATE KEY SEKALI
# ========================
if "private_key" not in st.session_state:
    priv, pub = generate_rsa_keypair()
    st.session_state.private_key = priv
    st.session_state.public_key = pub

rsa_priv = RSA.import_key(st.session_state.private_key)
rsa_pub = RSA.import_key(st.session_state.public_key)
# ========================
# PARAMETER RSA
# ========================
```

```
st.subheader("Parameter RSA")
st.write("Modulus (n):")
st.code(str(rsa_pub.n))
st.write("Eksponen publik (e):")
st.code(str(rsa_pub.e))
st.write("Eksponen privat (d):")
st.code(str(rsa_priv.d))
st.write("Bilangan prima p:")
st.code(str(rsa_priv.p))
st.write("Bilangan prima q:")
st.code(str(rsa_priv.q))
st.divider()
# ========================
# ENKRIPSI
# ========================
st.subheader("Enkripsi RSA")
plaintext = st.text_area("Plaintext", "Contoh pesan RSA")
N = st.number_input("Jumlah uji", 1, 100, 5)
if st.button("ENKRIPSI"):
    hasil = []
    ciphertext_hex, _ = rsa_encrypt_oaep(plaintext,
st.session_state.public_key)
    for i in range(N):
        _, t = rsa_encrypt_oaep(plaintext,
st.session_state.public_key)
        hasil.append({"Uji": i+1, "Waktu Enkripsi": t})
    df = pd.DataFrame(hasil)
    rata = df["Waktu Enkripsi"].mean()
    st.success("Enkripsi selesai")
    st.code(ciphertext_hex)
    st.dataframe(df)
    st.info(f"Rata-rata waktu: {rata:.6f} detik")
st.divider()
# ========================
# DEKRIPSI
# ========================
st.subheader("Dekripsi RSA")
cipher_input = st.text_area("Ciphertext (hex)")
N_dec = st.number_input("Jumlah uji dekripsi", 1, 100, 5, key="dec")
if st.button("DEKRIPSI"):
    try:
        plaintext_out, _ = rsa_decrypt_oaep(cipher_input,
        st.session_state.private_key)
        st.success("Dekripsi berhasil")
        st.code(plaintext_out)
        hasil = []
        for i in range(N_dec):
        _, t = rsa_decrypt_oaep(cipher_input,
        st.session_state.private_key)
            hasil.append({"Uji": i+1, "Waktu Dekripsi": t})
        df = pd.DataFrame(hasil)
        rata = df["Waktu Dekripsi"].mean()
        st.dataframe(df)
        st.info(f"Rata-rata waktu: {rata:.6f} detik")
    except Exception as e:
        st.error(f"Gagal: {e}")
```

Lampiran 3 Kode Program Uji *Hybrid* AES-RSA

```
import streamlit as st
from Crypto.Cipher import AES, PKCS1_OAEP
from Crypto.PublicKey import RSA
from Crypto.Random import get_random_bytes
from Crypto.Hash import SHA256
from Crypto.Util.Padding import pad, unpad
import time
import pandas as pd
# ========================
# CONFIG
# ========================
st.set_page_config(
    page_title="Hybrid AES–RSA",
    layout="wide"
)
st.title("Hybrid AES–RSA (AES-128 CBC) + Pengujian Waktu")
# ========================
# PARAMETER
# ========================
N = st.sidebar.number_input("Jumlah Uji (N)", 1, 100, 5)
rsa_bits = st.sidebar.selectbox("RSA Bit", [2048, 3072, 4096])
# ========================
# GENERATE RSA
# ========================
if "rsa_key" not in st.session_state:
    st.session_state.rsa_key = RSA.generate(rsa_bits)
rsa_key = st.session_state.rsa_key
# ========================
# INPUT AES KEY
# ========================
st.subheader("Kunci AES")
if st.button("Generate AES Key"):
    st.session_state.aes_key = get_random_bytes(16).hex()
aes_key_hex = st.text_input("AES Key (hex)",
value=st.session_state.get("aes_key", ""))
# ========================
# INPUT PLAINTEXT
# ========================
plaintext = st.text_area("Plaintext", "Contoh pesan hybrid AES RSA")
# ========================
# FUNGSI ENKRIPSI (FULL TIME)
# ========================
def hybrid_encrypt_once(K_AES, P_bytes, rsa_key):
    t_start = time.perf_counter()
    # Padding
    padded = pad(P_bytes, AES.block_size)
    # AES
    iv = get_random_bytes(16)
    cipher_aes = AES.new(K_AES, AES.MODE_CBC, iv=iv)
    C_text = cipher_aes.encrypt(padded)
    # RSA encrypt key
    cipher_rsa = PKCS1_OAEP.new(rsa_key.publickey(),hashAlgo=SHA256)
    C_key = cipher_rsa.encrypt(K_AES)
```

```
t_end = time.perf_counter()
    return C_text, C_key, iv, (t_end - t_start)
# ========================
# FUNGSI DEKRIPSI (FULL TIME)
# ========================
def hybrid_decrypt_once(C_text, C_key, iv, rsa_key):
    t_start = time.perf_counter()
    # RSA decrypt key
    cipher_rsa = PKCS1_OAEP.new(rsa_key, hashAlgo=SHA256)
    K_AES = cipher_rsa.decrypt(C_key)
    # AES decrypt
    cipher_aes = AES.new(K_AES, AES.MODE_CBC, iv=iv)
    padded = cipher_aes.decrypt(C_text)
    # Unpad
    P = unpad(padded, AES.block_size)
    t_end = time.perf_counter()
    return K_AES, P, (t_end - t_start)
# ========================
# ENKRIPSI
# ========================
if st.button("ENKRIPSI"):
    if not plaintext or not aes_key_hex:
        st.warning("Isi plaintext & key dulu")
    else:
        K_AES = bytes.fromhex(aes_key_hex)
        P_bytes = plaintext.encode()
        runs = []
        T_enc = []
        for i in range(N):
            C_text, C_key, iv, t = hybrid_encrypt_once(K_AES,
P_bytes, rsa_key)
            T_enc.append(t)
            runs.append((C_text, C_key, iv))
        st.session_state.runs = runs
        st.session_state.plaintext = plaintext
        df = pd.DataFrame({
            "Uji": range(1, N+1),
            "T_enc": T_enc
        })
        st.write(df)
        st.success(f"Rata-rata: {sum(T_enc)/len(T_enc):.6f} detik")
        # tampilkan contoh
        last = runs[-1]
        st.code(last[0].hex(), language="text")
        st.code(last[1].hex(), language="text")
        st.code(last[2].hex(), language="text")
# ========================
# DEKRIPSI
# ========================
if st.button("DEKRIPSI"):
    if "runs" not in st.session_state:
        st.warning("Lakukan enkripsi dulu")
    else:
        runs = st.session_state.runs
        P_ref = st.session_state.plaintext.encode()
```

```
T_dec = []
        status = []
        for C_text, C_key, iv in runs:
            K_AES, P, t = hybrid_decrypt_once(C_text, C_key, iv,
rsa_key)
            T_dec.append(t)
            status.append("Valid" if P == P_ref else "Tidak Valid")
        df = pd.DataFrame({
            "Uji": range(1, len(runs)+1),
            "T_dec": T_dec,
            "Status": status
        })
        st.write(df)
        st.success(f"Rata-rata: {sum(T_dec)/len(T_dec):.6f} detik")
```

Lampiran 4 Hasil Uji Waktu Proses Enkripsi (50 Kali Pengulangan)

Panjang Pesan	AES Tunggal	RSA Tunggal	*Hybrid* AES-RSA
50 Karakter	0.003492	0.001244	0.003982
	0.000069	0.001285	0.001417
	0.000034	0.002226	0.001403
	0.000095	0.002590	0.001369
	0.000082	0.001418	0.001368
	0.000028	0.001342	0.001384
	0.000021	0.001615	0.001384
	0.000076	0.001190	0.001601
	0.000022	0.001096	0.001311
	0.000020	0.001080	0.001357
	0.000019	0.001073	0.001414
	0.000140	0.001521	0.001340
	0.000081	0.001161	0.001378
	0.000074	0.001127	0.001364
	0.000067	0.001270	0.001330
	0.000088	0.001111	0.001444
	0.000016	0.001409	0.001416
	0.000013	0.001223	0.001424
	0.000012	0.001456	0.001334
	0.000011	0.001120	0.001415
	0.000011	0.001161	0.001438
	0.000011	0.001101	0.001416
	0.000011	0.001160	0.001393
	0.000012	0.001405	0.001390
	0.000012	0.001144	0.001337
	0.000011	0.001087	0.001356
	0.000013	0.001071	0.001239
	0.000012	0.001069	0.001286
	0.000011	0.001103	0.001361

	0.000011	0.001090	0.001494
	0.000011	0.001090	0.001499
	0.000010	0.001124	0.001476
	0.000011	0.001118	0.001348
	0.000011	0.001076	0.001151
	0.000012	0.001070	0.001113
	0.000012	0.001082	0.001165
	0.000011	0.001077	0.001113
	0.000012	0.001254	0.001105
	0.000011	0.001169	0.001133
	0.000011	0.001101	0.001100
	0.000011	0.001084	0.001112
	0.000011	0.001094	0.001136
	0.000011	0.001114	0.001092
	0.000010	0.001091	0.001120
	0.000010	0.001136	0.000787
	0.000010	0.001106	0.000995
	0.000010	0.001196	0.000756
	0.000011	0.001126	0.000741
	0.000011	0.001177	0.000725
	0.000010	0.001096	0.000724
75 Karakter	0.001359	0.001446	0.004005
	0.000045	0.002985	0.001160
	0.000025	0.001237	0.000922
	0.000020	0.001205	0.000932
	0.000051	0.001251	0.000953
	0.000025	0.001473	0.000832
	0.000019	0.001274	0.001024
	0.000019	0.001661	0.000818
	0.000167	0.001396	0.001007
	0.000067	0.001281	0.000747
	0.000147	0.001123	0.001051
	0.000044	0.001267	0.000737
	0.000037	0.001220	0.000761
	0.000034	0.001116	0.000729
	0.000040	0.001091	0.000727
	0.000118	0.001321	0.000707
	0.000029	0.001408	0.000820
	0.000034	0.001353	0.001126
	0.000113	0.001229	0.000778
	0.000045	0.001155	0.000748
	0.000195	0.001617	0.000705
	0.000056	0.001148	0.000728
	0.000113	0.001143	0.000739
	0.000050	0.001508	0.000724
	0.000111	0.001301	0.000702
	0.000047	0.001594	0.000738
	0.000171	0.001709	0.000908

	0.000039	0.001302	0.000798
	0.000042	0.001259	0.000828
	0.000158	0.001123	0.000709
	0.000029	0.001109	0.000730
	0.000027	0.001125	0.000725
	0.000022	0.001236	0.000855
	0.000018	0.001242	0.000736
	0.000018	0.001547	0.000733
	0.000012	0.001802	0.001096
	0.000013	0.001282	0.001388
	0.000013	0.001218	0.000808
	0.000032	0.001118	0.000744
	0.000023	0.001089	0.000818
	0.000019	0.001151	0.001174
	0.000016	0.001424	0.000766
	0.000014	0.001163	0.000861
	0.000011	0.001168	0.000984
	0.000012	0.001494	0.000801
	0.000011	0.001524	0.000696
	0.000012	0.001519	0.000755
	0.000013	0.001821	0.000721
	0.000012	0.001282	0.000718
	0.000013	0.001315	0.000698
100 Karakter	0.000672	0.001330	0.003442
	0.000289	0.001265	0.000801
	0.000354	0.002984	0.000727
	0.000068	0.001548	0.000767
	0.000047	0.001201	0.001220
	0.000039	0.001397	0.001484
	0.000190	0.001159	0.001193
	0.000037	0.001184	0.001256
	0.000031	0.001113	0.001342
	0.000025	0.001132	0.001186
	0.000019	0.001354	0.000841
	0.000164	0.001143	0.000809
	0.000031	0.001252	0.000770
	0.000217	0.001309	0.000753
	0.000047	0.001212	0.000747
	0.000044	0.001151	0.000889
	0.000117	0.001159	0.000859
	0.000032	0.001227	0.000824
	0.000129	0.001204	0.000748
	0.000020	0.001459	0.000836
	0.000086	0.001199	0.000830
	0.000023	0.001249	0.001004
	0.000088	0.001242	0.000835
	0.000021	0.001147	0.000707
	0.000022	0.001139	0.000891

	0.000016	0.001146	0.000931
	0.000085	0.001136	0.000943
	0.000014	0.001210	0.000814
	0.000012	0.001246	0.000821
	0.000032	0.001212	0.001047
	0.000015	0.001152	0.001107
	0.000012	0.001127	0.001012
	0.000011	0.001206	0.001397
	0.000011	0.001316	0.000824
	0.000011	0.001373	0.001033
	0.000011	0.001315	0.000814
	0.000011	0.001162	0.000752
	0.000011	0.001214	0.000773
	0.000011	0.001095	0.000808
	0.000011	0.001175	0.001106
	0.000012	0.001202	0.000762
	0.000011	0.001200	0.000790
	0.000011	0.001262	0.000792
	0.000012	0.001102	0.000823
	0.000011	0.001175	0.000776
	0.000019	0.001625	0.000721
	0.000011	0.001324	0.000817
	0.000011	0.001206	0.000724
	0.000011	0.001307	0.000707
	0.000011	0.001178	0.000748
150 Karakter	0.000147	0.001232	0.004858
	0.000157	0.001135	0.001517
	0.000033	0.002697	0.001483
	0.000713	0.001193	0.001458
	0.000055	0.001180	0.001442
	0.000191	0.001214	0.001430
	0.000035	0.001282	0.001521
	0.000338	0.001510	0.001681
	0.000312	0.001149	0.001417
	0.000065	0.001215	0.001426
	0.000055	0.001469	0.001398
	0.000050	0.001159	0.001387
	0.000248	0.001159	0.001414
	0.000119	0.001128	0.001388
	0.000051	0.001184	0.001459
	0.000097	0.001144	0.001399
	0.000208	0.001126	0.001426
	0.000060	0.001143	0.001388
	0.000023	0.001207	0.001387
	0.000112	0.001201	0.001442
	0.000082	0.001904	0.001425
	0.000091	0.001528	0.001489
	0.000149	0.001132	0.001421

	0.000018	0.001101	0.001414
	0.000017	0.001733	0.001402
	0.000013	0.001369	0.001407
	0.000013	0.001146	0.001412
	0.000011	0.001170	0.001418
	0.000011	0.001097	0.001407
	0.000010	0.001083	0.001404
	0.000011	0.001089	0.001534
	0.000011	0.001079	0.001427
	0.000011	0.001095	0.001402
	0.000011	0.001171	0.001410
	0.000011	0.001220	0.001475
	0.000010	0.001136	0.001403
	0.000011	0.001093	0.001398
	0.000011	0.001087	0.001416
	0.000011	0.001106	0.001398
	0.000010	0.001086	0.002029
	0.000011	0.001087	0.001434
	0.000035	0.001138	0.001415
	0.000020	0.001222	0.001412
	0.000012	0.001645	0.001478
	0.000011	0.001191	0.001417
	0.000011	0.001318	0.001420
	0.000011	0.001352	0.001385
	0.000011	0.001123	0.001398
	0.000012	0.001141	0.001412
	0.000011	0.001105	0.001388
200 Karakter	0.000263	0.000000	0.004458
	0.000989		0.001401
	0.000294		0.001424
	0.000047		0.001333
	0.000043		0.001298
	0.000132		0.001336
	0.000035		0.001301
	0.000019		0.001617
	0.000093		0.001323
	0.000195		0.001368
	0.000043		0.001334
	0.000027		0.001305
	0.000087		0.001293
	0.000024		0.001319
	0.000020		0.001289
	0.000020		0.001368
	0.000022		0.001366
	0.000016		0.001301
	0.000086		0.001302
	0.000032		0.001328
	0.000017		0.001306

	0.000097		0.001304
	0.000028		0.001323
	0.000020		0.001291
	0.000016		0.001319
	0.000095		0.001315
	0.000086		0.001298
	0.000094		0.001306
	0.000016		0.001307
	0.000023		0.001330
	0.000137		0.001299
	0.000031		0.001308
	0.000025		0.001334
	0.000025		0.001454
	0.000014		0.001334
	0.000012		0.001299
	0.000011		0.001324
	0.000012		0.001303
	0.000011		0.001304
	0.000011		0.001279
	0.000011		0.001292
	0.000011		0.001322
	0.000011		0.001390
	0.000011		0.001319
	0.000011		0.001315
	0.000011		0.001386
	0.000011		0.001466
	0.000012		0.001363
	0.000011		0.001288
	0.000011		0.001311

Lampiran 5 Hasil Uji Waktu Proses Dekripsi (50 Kali Pengulangan)

Panjang Pesan	**AES Tunggal**	**RSA Tunggal**	***Hybrid* AES-RSA**
50 Karakter	0.000146	0.045947	0.003687
	0.000075	0.046366	0.001675
	0.000034	0.048465	0.001557
	0.000033	0.051313	0.001787
	0.000026	0.045166	0.001560
	0.000205	0.045654	0.001506
	0.000031	0.044727	0.002075
	0.000025	0.043950	0.001637
	0.000193	0.044452	0.001580
	0.000055	0.044429	0.001485
	0.000042	0.044775	0.001526
	0.000054	0.049986	0.001613
	0.000040	0.047121	0.001545

	0.000038	0.046091	0.001663
	0.000052	0.048816	0.001661
	0.000051	0.048507	0.001739
	0.000036	0.046753	0.001500
	0.000030	0.045233	0.001647
	0.000081	0.046359	0.001611
	0.000125	0.047485	0.001677
	0.000035	0.048962	0.001716
	0.000025	0.048253	0.001771
	0.000176	0.046699	0.001538
	0.000046	0.044902	0.001528
	0.000035	0.044627	0.001614
	0.000029	0.043860	0.001658
	0.000023	0.055223	0.001540
	0.000101	0.047545	0.001558
	0.000026	0.046123	0.001702
	0.000101	0.047247	0.001525
	0.000040	0.057977	0.002037
	0.000023	0.047720	0.002277
	0.000149	0.047004	0.001567
	0.000028	0.047648	0.001551
	0.000041	0.048186	0.001489
	0.000030	0.047898	0.001635
	0.000031	0.046186	0.001502
	0.000032	0.048029	0.001582
	0.000109	0.048187	0.001662
	0.000040	0.047850	0.001844
	0.000035	0.048060	0.002321
	0.000030	0.053011	0.002253
	0.000035	0.053469	0.002161
	0.000030	0.072158	0.001631
	0.000026	0.071131	0.001596
	0.000106	0.054096	0.001524
	0.000023	0.062753	0.001609
	0.000193	0.045186	0.001797
	0.000044	0.045613	0.001646
	0.000026	0.045954	0.001621
75 Karakter	0.000145	0.047831	0.005353
	0.000045	0.046995	0.002510
	0.000027	0.047016	0.002508
	0.000039	0.047257	0.002424
	0.000026	0.048896	0.002805
	0.000022	0.047262	0.002334
	0.000021	0.048295	0.002353
	0.000021	0.047253	0.002355
	0.000020	0.049661	0.002455
	0.000020	0.046233	0.002471
	0.000019	0.046773	0.002429

	0.000020	0.048021	0.002506
	0.000019	0.051546	0.002408
	0.000065	0.051078	0.002211
	0.000037	0.046921	0.002354
	0.000066	0.045682	0.002395
	0.000039	0.047531	0.002470
	0.000033	0.045978	0.002553
	0.000030	0.047642	0.002552
	0.000029	0.045978	0.002574
	0.000029	0.048149	0.002627
	0.000231	0.048260	0.002558
	0.000052	0.072761	0.002529
	0.000026	0.068418	0.002536
	0.000021	0.057529	0.002602
	0.000020	0.048622	0.002320
	0.000020	0.046284	0.002324
	0.000020	0.051235	0.002484
	0.000020	0.076640	0.002579
	0.000026	0.046388	0.002550
	0.000030	0.046054	0.002538
	0.000153	0.046259	0.002709
	0.000040	0.048575	0.002348
	0.000035	0.050308	0.002426
	0.000034	0.048127	0.002441
	0.000039	0.046134	0.002822
	0.000029	0.046514	0.002496
	0.000028	0.048876	0.002518
	0.000027	0.045224	0.002546
	0.000036	0.047374	0.002473
	0.000028	0.049715	0.002422
	0.000027	0.048909	0.002474
	0.000037	0.051373	0.002470
	0.000027	0.050418	0.002491
	0.000060	0.048231	0.002455
	0.000051	0.043847	0.002328
	0.000031	0.064527	0.001978
	0.000028	0.046562	0.002077
	0.000041	0.046212	0.002046
	0.000040	0.045165	0.001974
100 Karakter	0.000127	0.070567	0.003887
	0.000041	0.052018	0.001584
	0.000070	0.046971	0.001778
	0.000044	0.047563	0.001585
	0.000028	0.044078	0.001762
	0.000023	0.046780	0.001602
	0.000022	0.045339	0.001723
	0.000021	0.045041	0.001845
	0.000019	0.045832	0.001537

	0.000088	0.047478	0.001694
	0.000024	0.049849	0.001604
	0.000021	0.044692	0.001549
	0.000134	0.044940	0.001561
	0.000022	0.044272	0.001487
	0.000034	0.044144	0.001625
	0.000020	0.046507	0.001630
	0.000023	0.046941	0.001750
	0.000019	0.044580	0.001677
	0.000028	0.048137	0.002044
	0.000036	0.047285	0.001849
	0.000037	0.045406	0.001690
	0.000023	0.047393	0.001531
	0.000017	0.044347	0.001670
	0.000033	0.046342	0.001621
	0.000038	0.043141	0.001956
	0.000027	0.044455	0.001612
	0.000021	0.043505	0.001571
	0.000018	0.043849	0.001519
	0.000052	0.046726	0.001691
	0.000042	0.048953	0.001624
	0.000043	0.046835	0.001660
	0.000034	0.045156	0.001717
	0.000040	0.045319	0.001646
	0.000038	0.045789	0.001715
	0.000020	0.050916	0.001651
	0.000118	0.050862	0.001520
	0.000030	0.051971	0.001597
	0.000104	0.059986	0.001541
	0.000044	0.055511	0.001652
	0.000025	0.053729	0.001532
	0.000181	0.050204	0.001601
	0.000127	0.051543	0.001903
	0.000043	0.052164	0.001848
	0.000037	0.047329	0.001924
	0.000034	0.050866	0.001734
	0.000038	0.048790	0.001723
	0.000044	0.088238	0.001694
	0.000058	0.054347	0.001738
	0.000046	0.047110	0.001767
	0.000104	0.048898	0.002223
150 Karakter	0.000142	0.046498	0.004753
	0.000050	0.049774	0.002232
	0.000139	0.051859	0.001943
	0.000031	0.048526	0.001549
	0.000819	0.049079	0.001650
	0.000466	0.049891	0.001877
	0.000204	0.076962	0.001529

	0.000030	0.081245	0.001665
	0.000025	0.045420	0.001707
	0.000278	0.045768	0.001514
	0.000031	0.052268	0.001676
	0.000021	0.085427	0.001530
	0.000019	0.081937	0.001785
	0.000020	0.048607	0.001670
	0.000021	0.048434	0.001540
	0.000022	0.046110	0.001580
	0.000019	0.045930	0.001591
	0.000022	0.046110	0.001518
	0.000018	0.044666	0.001713
	0.000185	0.043915	0.001721
	0.000038	0.044486	0.001526
	0.000022	0.043452	0.001546
	0.000021	0.043118	0.001652
	0.000018	0.045109	0.001630
	0.000021	0.045745	0.001542
	0.000035	0.046109	0.001500
	0.000119	0.046123	0.001606
	0.000023	0.050572	0.001555
	0.000022	0.044745	0.001562
	0.000024	0.046215	0.001564
	0.000025	0.043830	0.001572
	0.000021	0.044432	0.001487
	0.000018	0.048329	0.001488
	0.000135	0.045440	0.001488
	0.000188	0.048131	0.001503
	0.000140	0.054719	0.001529
	0.000035	0.050585	0.001667
	0.000024	0.046625	0.001574
	0.000018	0.046171	0.001518
	0.000018	0.050839	0.001516
	0.000019	0.047097	0.001477
	0.000020	0.050914	0.001663
	0.000020	0.051181	0.001518
	0.000020	0.044328	0.001512
	0.000123	0.044337	0.001530
	0.000024	0.045171	0.001522
	0.000018	0.061580	0.001476
	0.000029	0.049945	0.001493
	0.000020	0.046845	0.001706
	0.000016	0.047373	0.001936
200 Karakter	0.000126	0.000000	0.004204
	0.000038		0.001568
	0.000035		0.001522
	0.000071		0.002260
	0.000034		0.002073

	0.000044		0.001591
	0.000033		0.001525
	0.000053		0.001567
	0.000028		0.001734
	0.000039		0.001606
	0.000030		0.001734
	0.000035		0.002044
	0.000023		0.001524
	0.000018		0.001640
	0.000137		0.001683
	0.000051		0.001608
	0.000048		0.001633
	0.000024		0.001662
	0.000127		0.002072
	0.000035		0.001518
	0.000021		0.001617
	0.000021		0.001735
	0.000020		0.001664
	0.000019		0.001658
	0.000029		0.001606
	0.000022		0.001569
	0.000021		0.001547
	0.000020		0.001590
	0.000018		0.001549
	0.000016		0.001581
	0.000031		0.001545
	0.000027		0.001516
	0.000078		0.001532
	0.000025		0.001556
	0.000020		0.001574
	0.000085		0.001632
	0.000024		0.001785
	0.000083		0.001492
	0.000124		0.001477
	0.000043		0.001684
	0.000021		0.001510
	0.000018		0.001500
	0.000080		0.001492
	0.000023		0.001632
	0.000020		0.001712
	0.000081		0.001912
	0.000068		0.001688
	0.000021		0.001578
	0.000022		0.001713
	0.000085		0.001494

Lampiran 6 Tabel ASCII 8-bit

Decimal	Octal	Hex	Binary	Karakter	Deskripsi
0	000	00	00000000	NUL	Null
1	001	01	00000001	SOH	Start of Header
2	002	02	00000010	STX	Start of Text
3	003	03	00000011	ETX	End of Text
4	004	04	00000100	EOT	End of Transmission
5	005	05	00000101	ENQ	Enquiry
6	006	06	00000110	ACK	Acknowledge
7	007	07	00000111	BEL	Bell
8	010	08	00001000	BS	Backspace
9	011	09	00001001	HT	Horizontal Tab
10	012	0A	00001010	LF	Line Feed
11	013	0B	00001011	VT	Vertical Tab
12	014	0C	00001100	FF	Form Feed
13	015	0D	00001101	CR	Carriage Return
14	016	0E	00001110	SO	Shift Out
15	017	0F	00001111	SI	Shift In
16	020	10	00010000	DLE	Data Link Escape
17	021	11	00010001	DC1	Device Control 1
18	022	12	00010010	DC2	Device Control 2
19	023	13	00010011	DC3	Device Control 3
20	024	14	00010100	DC4	Device Control 4
21	025	15	00010101	NAK	Negative Acknowledge
22	026	16	00010110	SYN	Synchronous Idle
23	027	17	00010111	ETB	End of Transmission Block
24	030	18	00011000	CAN	Cancel
25	031	19	00011001	EM	End of Medium

26	032	1A	00011010	SUB	Substitute
27	033	1B	00011011	ESC	Escape
28	034	1C	00011100	FS	File Separator
29	035	1D	00011101	GS	Group Separator
30	036	1E	00011110	RS	Record Separator
31	037	1F	00011111	US	Unit Separator
32	040	20	00100000	(spasi)	Space
33	041	21	00100001	!	Exclamation
34	042	22	00100010	"	Double Quote
35	043	23	00100011	#	Number Sign
36	044	24	00100100	$	Dollar
37	045	25	00100101	%	Percent
38	046	26	00100110	&	Ampersand
39	047	27	00100111	'	Single Quote
40	050	28	00101000	(	Left Parenthesis
41	051	29	00101001	)	Right Parenthesis
42	052	2A	00101010	*	Asterisk
43	053	2B	00101011	+	Plus
44	054	2C	00101100	,	Comma
45	055	2D	00101101	-	Dash
46	056	2E	00101110	.	Dot
47	057	2F	00101111	/	Slash
48	060	30	00110000	0	Digit
49	061	31	00110001	1	Digit
50	062	32	00110010	2	Digit
51	063	33	00110011	3	Digit
52	064	34	00110100	4	Digit
53	065	35	00110101	5	Digit
54	066	36	00110110	6	Digit

55	067	37	00110111	7	Digit
56	070	38	00111000	8	Digit
57	071	39	00111001	9	Digit
58	072	3A	00111010	:	Colon
59	073	3B	00111011	;	Semicolon
60	074	3C	00111100	<	Less Than
61	075	3D	00111101	=	Equal
62	076	3E	00111110	>	Greater Than
63	077	3F	00111111	?	Question
64	100	40	01000000	@	At Symbol
65–90	...	...	...	A–Z	Huruf kapital
97–122	...	...	...	a–z	Huruf kecil
123	173	7B	01111011	{	Left Brace
124	174	7C	01111100	|	Vertical Bar
125	175	7D	01111101	}	Right Brace
126	176	7E	01111110	~	Tilde
127	177	7F	01111111	DEL	Delete

RIWAYAT HIDUP

Qotrunnada Shobaha Zahirah, lahir di Mojokerto pada tanggal 16 Agustus 2004. Penulis berdomisili di Dusun Prayan, Desa Watesumpak, Kecamatan Trowulan, Kabupaten Mojokerto. Penulis merupakan anak pertama dari tiga bersaudara, putri dari pasangan Bapak Yusman, S.Pd dan Ibu Eni Mufarrida, S.E. Perjalanan pendidikan penulis di mulai di RA Salafiyah Syafi'iyah Klinterejo dan lulus pada tahun 2010. Selanjutnya, penulis melanjutkan ke jenjang sekolah dasar di MI Salafiyah Syafi'iyah II Klinterejo dan lulus pada tahun 2016. Pendidikan menengah pertama ditempuh di MTs Negeri 2 Mojokerto dan lulus pada tahun 2019. Penulis melanjutkan pendidikan ke jenjang menegah atas di MAN Kota Mojokerto dan lulus pada tahun 2022. Pada jenjang ini, penulis aktif berorganisasi sebagai anggota OSIS, yang menjadi wadah pengembangan kepemimpinan dan tanggung jawab sosial. Selanjutnya penulis melanjutkan pendidikan Strata 1 di Universitas Islam Negeri Maulana Malik Ibrahim Malang dan memilih Program Studi Matematika Fakultas Sains dan Teknologi. Selama menempuh pendidikan di perguruan tinggi, penulis aktif berpartisipasi dalam kegiatan organisasi kemahasiswaan, khususnya sebagai anggota Himpunan Program Studi (HMPS) "Integral Matematika" Divisi *Internal Public Relation* selama dua periode yaitu pada tahun 2023-2024.

KEMENTERIAN AGAMA RI
UNIVERSITAS ISLAM NEGERI
MAULANA MALIK IBRAHIM MALANG
FAKULTAS SAINS DAN TEKNOLOGI
Jl. Gajayana No.50 Dinoyo Malang Telp. / Fax. (0341)558933

BUKTI KONSULTASI SKRIPSI

Nama : Qotrunnada Shobaha Zahirah
NIM : 220601110017
Fakultas / Jurusan : Sains dan Teknologi / Matematika
Judul Skripsi : Evaluasi Waktu Komputasi Algoritma *Advanced Encryption Standard* (AES) dengan *Rivest-Shamir-Adleman* (RSA) untuk Keamanan Pesan Teks
Pembimbing I : Muhammad Khudzaifah, M.Si
Pembimbing II : Dr. Fachrur Rozi, M.Si

No	Tanggal	Hal	Tanda Tangan
1.	10 September 2025	Konsultasi Bab I, II, dan III	1.
2.	17 September 2025	Konsultasi Bab I, II, dan III	2.
3.	24 September 2025	Konsultasi Bab I, II, dan III	3.
4.	1 Oktober 2025	Konsultasi Bab I, II, dan III	4.
5.	13 Oktober 2025	Konsultasi Kajian Agama Bab I dan II	5.
6.	16 Oktober 2025	Konsultasi Bab I, II, dan III	6.
7.	18 Oktober 2025	Konsultasi Bab I, II, dan III	7.
8.	20 Oktober 2025	Konsultasi Bab I, II, dan III	8.
9.	21 Oktober 2025	ACC Bab I, II, dan III	9.
10.	22 Oktober 2025	ACC Kajian Agama Bab I dan II	10.
11.	23 Oktober 2025	ACC Seminar Proposal	11.
12.	20 November 2025	Konsultasi Revisi Seminar Proposal	12.
13.	10 Desember 2025	Konsultasi Bab IV dan V	13.
14.	26 Januari 2026	Konsultasi Kajian Agama Bab IV	14.
15.	23 Januari 2026	ACC Bab IV dan V	15.
16.	6 Februari 2026	ACC Kajian Agama Bab IV	16.

KEMENTERIAN AGAMA RI
UNIVERSITAS ISLAM NEGERI
MAULANA MALIK IBRAHIM MALANG
FAKULTAS SAINS DAN TEKNOLOGI
Jl. Gajayana No.50 Dinoyo Malang Telp. / Fax. (0341)558933

17.	7 Februari 2026	ACC Seminar Hasil	17.
18.	23 Februari 2026	Konsultasi Revisi Seminar Hasil	18.
19.	25 Januari 2026	Konsultasi Revisi Seminar Hasil	19.
20.	7 April 2026	ACC Sidang Skripsi	20.
21.	20 April 2026	ACC Keseluruhan	21.

Malang, 20 April 2026
Mengetahui,
Ketua Program Studi Matematika

Dr. Fachrur Rozi, M.Si.
NIP. 19800527 200801 1 012