JURNAL ILMIAH FIFO
Vol. 17 No. 1 (2025) 78-90
DOI: 10.22441/fifo.2025.v17i1.009

P-ISSN 2085-4315
E-ISSN 2502-8332

Sistem Enkripsi Dokumen Digital Melalui Kombinasi AES-128 dan *Hashing* SHA-256 Berbasis *Salt*

Faiz Muqorrir Kaaffah[1*], Nurhasan Nugroho[2], Desi Nurnaningsih[3], Harriansyah[4]

[1]Program Studi Informatika, Universitas Islam Negeri Siber Syekh Nurjati Cirebon
[2]Program Studi Ilmu Komputer, Fakultas Ilmu Komputer, Universitas Bina Bangsa
[3]Program Studi Teknologi Informasi, Fakultas Informatika, Universitas Telkom. Jakarta
[4]Program Studi Sistem Informasi, Sekolah Tinggi Teknologi Informasi NIIT
Email: [1]faiz@syekhnurjati.ac.id, [2]nurhasan.nugroho@binabangsa.ac.id,
[3]desinurnaningsih@telkomuniversity.ac.id, [4]harri@i-tech.ac.id
Penulis Korespondensi[*]

(***received*****:** 03-06-25, ***revised*****:** 13-06-25, ***accepted*****:** 23-06-25)

Abstrak

Pengelolaan dokumen digital menuntut sistem keamanan yang andal untuk mencegah akses tidak sah dan menjaga kerahasiaan informasi. Penelitian ini merancang dan mengimplementasikan sistem pengamanan dokumen berbasis web dengan mengombinasikan algoritma enkripsi simetris *Advanced Encryption Standard* (AES-128) dan hashing SHA-256 berbasis *salt*. AES-128 digunakan untuk mengenkripsi isi dokumen agar tidak dapat diakses oleh pihak yang tidak berwenang, sedangkan SHA-256 dengan salt digunakan untuk memperkuat autentikasi pengguna dan mencegah serangan berbasis *rainbow table*. Sistem dikembangkan menggunakan bahasa pemrograman PHP dengan basis data MySQL dan terdiri dari dua modul utama: enkripsi *file* dan autentikasi pengguna. Pengujian dilakukan secara fungsional dan performa terhadap 10 *file* uji dengan format berbeda, serta simulasi dictionary attack untuk mengukur ketahanan sistem terhadap serangan kamus. Hasil menunjukkan bahwa seluruh fitur utama berjalan dengan baik, proses enkripsi rata-rata 135 ms dan dekripsi di bawah 150 ms, serta autentikasi berhasil menolak seluruh login tidak sah. Skema penyimpanan password dalam format *salt:hash* menghasilkan hash yang unik dan aman. Batasan sistem ini adalah belum tersedianya fitur *reset password* enkripsi dan belum diuji terhadap serangan lanjutan seperti SQL *injection* atau *side-channel attack*. Integrasi dua metode kriptografi ini memberikan perlindungan berlapis terhadap dokumen digital serta meningkatkan keandalan sistem autentikasi. Antarmuka yang ramah pengguna menjadikan sistem ini relevan untuk diimplementasikan pada institusi yang membutuhkan tingkat keamanan data tinggi.

Kata Kunci: AES-128, SHA-256, *Salt*, Enkripsi Dokumen, Kriptografi, Pengamanan Dokumen Digital

Abstract

The management of digital documents requires a reliable security system to prevent unauthorized access and ensure information confidentiality. This study designs and implements a web-based document security system by combining the symmetric encryption algorithm Advanced Encryption Standard (AES-128) and salt-based SHA-256 hashing. AES-128 is used to encrypt document content, preventing access by unauthorized parties, while SHA-256 with salt is employed to strengthen user authentication and protect against rainbow table attacks. The system is developed using the PHP programming language with a MySQL database and consists of two main modules: file encryption and user authentication. Functional and performance testing was conducted on 10 test files in various formats, along with a dictionary attack simulation to evaluate the system's resilience against common password attacks. The results show that all core features functioned as intended, with average encryption time of 135 ms and decryption under 150 ms, while authentication successfully rejected all unauthorized login attempts. The password storage scheme using the salt:hash format generated unique and secure hashes. The system's limitations include the absence of an encryption password reset feature and the lack of testing against advanced attacks such as SQL injection or side-channel attacks. The integration of these two cryptographic methods provides layered protection for digital documents and enhances the reliability of the authentication system. Its user-friendly interface makes the system suitable for institutions requiring a high level of data security.

Keywords*: AES-128, SHA-256, Salt, Document Encryption, Cryptography, Digital Document Security*

JURNAL ILMIAH FIFO
Vol. 17 No. 1 (2025) 78-90
DOI: 10.22441/fifo.2025.v17i1.009
P-ISSN 2085-4315
E-ISSN 2502-8332

1. PENDAHULUAN

Dalam era digital, dokumen digital menjadi komponen vital dalam pertukaran informasi di sektor pemerintahan, pendidikan, dan industri. Namun, meningkatnya pemanfaatan sistem berbasis web juga memperbesar risiko keamanan, seperti pencurian, penyadapan, dan modifikasi dokumen oleh pihak tidak berwenang [1]. Ancaman ini menuntut penerapan mekanisme kriptografi yang mampu menjamin kerahasiaan, keutuhan, dan kontrol akses informasi [2]. Oleh karena itu, pengamanan dokumen digital membutuhkan solusi yang mengintegrasikan algoritma kriptografi yang kuat dan efisien untuk mengatasi berbagai potensi ancaman siber.

Salah satu algoritma kriptografi simetris yang banyak digunakan untuk pengamanan data adalah *Advanced Encryption Standard* (AES). Varian AES-128 dikenal karena efisiensinya dalam proses enkripsi dan dekripsi, serta tingkat keamanannya yang memadai untuk berbagai kebutuhan [3]. Berbagai penelitian sebelumnya telah menunjukkan efektivitas AES dalam mengamankan *file* digital. Salah satu penelitian mengimplementasikan AES-128 dalam sistem pengamanan dokumen keuangan di CV. Multikreasi Bersama, dan menunjukkan bahwa dokumen penting dapat diamankan dan dikelola dengan baik melalui sistem enkripsi berbasis web [4]. Penelitian lain memanfaatkan AES-128 untuk pengamanan data keuangan kas, dengan hasil bahwa meskipun *file* yang digunakan berupa Excel, algoritma ini tetap memberikan tingkat keamanan yang tinggi karena penggunaan kunci rahasia yang kompleks [5]. Penelitian selanjutnya dilakukan pada PT. Gunung Geulis Elok Abadi, di mana AES-128 digunakan untuk mengamankan dokumen Perusahaan [6]. Hasil percobaan pada dua puluh *file* menunjukkan rata-rata waktu enkripsi sebesar 12.769 milidetik dan waktu dekripsi sebesar 18.075 milidetik. Penelitian lainnya membuktikan bahwa AES-128 mampu mengubah isi *file* menjadi simbol-simbol yang tidak dapat dibaca tanpa proses dekripsi, sehingga memperkuat aspek kerahasiaan data [7].

Meskipun AES-128 memberikan jaminan keamanan pada level isi dokumen, aspek autentikasi pengguna juga memegang peran krusial dalam sistem keamanan secara menyeluruh. Akan tetapi, banyak sistem masih menggunakan metode *hashing* lama seperti MD5, yang diketahui rentan terhadap serangan *brute-force* dan *collision* [8]. Oleh karena itu, pendekatan modern seperti SHA-256 berbasis *salt* menjadi solusi yang lebih aman. Algoritma SHA-256 menghasilkan *hash* sepanjang 256 bit yang kuat terhadap serangan kriptografi, sementara penggunaan *salt* menambah kompleksitas dan mencegah serangan berbasis tabel *rainbow* [9].

Penelitian ini bertujuan untuk pengembangan sistem pengamanan dokumen digital yang mengombinasikan AES-128 untuk enkripsi isi *file* dan SHA-256 berbasis *salt* untuk pengamanan kata sandi pengguna. Kombinasi ini dirancang tidak hanya untuk menjaga kerahasiaan konten dokumen, tetapi juga untuk memperkuat proses autentikasi pengguna agar lebih tahan terhadap berbagai serangan siber. Sistem yang dibangun memungkinkan pengguna untuk mengunggah dokumen, melakukan proses enkripsi yang efisien, serta menyimpan *file* dengan aman. Proses dekripsi hanya dapat dilakukan oleh pengguna yang berhasil melalui autentikasi yang valid. Kontribusi utama dari penelitian ini adalah rancangan dan implementasi sistem enkripsi dokumen berbasis web yang mengintegrasikan dua pendekatan kriptografi, yaitu AES-128 dan SHA-256 dengan *salt*, dalam satu kesatuan sistem. Sistem ini tidak hanya meningkatkan perlindungan terhadap isi dokumen digital, tetapi juga meningkatkan keamanan identitas pengguna, sehingga menciptakan solusi komprehensif yang dapat digunakan pada berbagai sistem informasi yang membutuhkan keamanan tingkat tinggi.

2. METODE PENELITIAN

Penelitian ini bertujuan untuk mengembangkan sistem pengamanan dokumen digital berbasis web melalui kombinasi algoritma *Advanced Encryption Standard* (AES-128) dan *hashing* SHA-256 berbasis *salt*. Adapun tahapan dalam penelitian ini ditunjukkan pada Gambar 1.

Gambar 1. Alur Tahapan Penelitian

P-ISSN 2085-4315
E-ISSN 2502-8332

Penjelasan dari masing-masing tahapan pada Gambar 1 disampaikan sebagai berikut.

2.1. Identifikasi Masalah

Sistem berbasis web yang menyimpan dokumen digital sering kali menghadapi dua masalah utama: (1) isi *file* dapat dibaca atau disalin oleh pihak tidak berwenang, dan (2) proses *login* pengguna rentan terhadap pencurian kredensial. Penggunaan algoritma MD5 untuk *hashing password* juga terbukti lemah dan tidak mampu menghadapi serangan seperti *rainbow table* atau *collision attack* [8]. Oleh karena itu, solusi yang diajukan adalah penggunaan enkripsi simetris AES-128 untuk file dan hashing SHA-256 berbasis salt unik untuk setiap password, guna melindungi dokumen dari akses ilegal dan menjaga integritas kredensial pengguna.

2.2. Analisis Kebutuhan dan Perancangan Sistem

Analisis kebutuhan dilakukan untuk mengidentifikasi komponen fungsional dan non-fungsional dari sistem [10]. Secara fungsional, sistem harus dapat: (1) menerima unggahan *file* dari pengguna, (2) mengenkripsi *file* dengan AES-128, (3) menyimpan metadata dan hasil enkripsi, (4) menyediakan autentikasi berbasis *username* dan *password* yang diamankan dengan SHA-256 dan *salt*, serta (5) mendekripsi *file* untuk pengguna yang telah diverifikasi. Perancangan sistem dibagi menjadi dua modul utama: (a) modul enkripsi *file* dan (b) modul autentikasi pengguna. Modul pertama menangani unggahan *file*, proses enkripsi, dan penyimpanan *file* terenkripsi. Modul kedua menangani proses registrasi pengguna, pembuatan *salt* dan *hash password*, serta verifikasi saat *login*. Diagram alur sistem menggambarkan aliran proses dari *login* hingga proses enkripsi dan dekripsi.

2.3. Proses Enkripsi AES-128

Proses enkripsi dokumen menggunakan algoritma AES-128 (*Advanced Encryption Standard*) dilakukan terhadap *file* digital dalam blok 128 bit (16 *byte*) [11]. Setiap *file* yang diunggah akan diproses per blok oleh sistem dan dienkripsi menggunakan kunci tetap sepanjang 128 bit yang sudah ditentukan [12]. Berikut tahapan proses enkripsi AES-128:

1) Inisialisasi Kunci dan Pembentukan *State*
   File yang diunggah dipecah menjadi blok-blok berukuran 128 bit. Setiap blok akan dimasukkan ke dalam sebuah matriks 4x4 *byte* yang disebut sebagai state. Kunci utama sepanjang 128 bit juga dibentuk ke dalam bentuk matriks dan digunakan untuk menghasilkan serangkaian *round keys* yang digunakan dalam setiap putaran enkripsi [13].
2) AddRoundKey (Awal)
   Pada tahap awal, blok data dikombinasikan dengan *round key* pertama menggunakan operasi XOR. Tahap ini berfungsi untuk mengikat data ke kunci dan menjadi fondasi keamanan enkripsi karena seluruh operasi selanjutnya dilakukan pada data yang telah "terikat" dengan kunci pengguna.
3) Proses Enkripsi Putaran (10 *Round*)
   Enkripsi AES-128 berlangsung dalam sepuluh putaran. Setiap putaran dari 1 hingga 9 terdiri atas empat transformasi: SubBytes, ShiftRows, MixColumns, dan AddRoundKey, yang bertujuan meningkatkan difusi dan keamanan data. Pada putaran ke-10, proses dilakukan tanpa MixColumns, hanya melibatkan tiga transformasi pertama untuk menyederhanakan tahap akhir enkripsi tanpa mengurangi kekuatannya.
4) SubBytes
   Setiap *byte* dalam state digantikan dengan nilai substitusi dari tabel S-Box (*substitution box*). Transformasi ini memberikan non-linearitas dan meningkatkan difusi, sehingga satu perubahan kecil pada input akan mempengaruhi seluruh keluaran.
5) ShiftRows
   Baris kedua hingga keempat dari state dimodifikasi dengan pergeseran ke kiri sejumlah indeks tertentu. Baris pertama tidak digeser, baris kedua digeser satu *byte* ke kiri, dan seterusnya. Ini memperluas efek dari *byte* individual ke seluruh blok.
6) MixColumns
   Masing-masing kolom dalam state dimodifikasi melalui transformasi linear berbasis aritmetika GF(2^8). Tujuannya adalah mencampur *byte* dalam satu kolom agar tidak hanya bergantung pada satu elemen, meningkatkan keamanan terhadap analisis diferensial.
7) AddRoundKey (Lanjutan)
   Pada akhir setiap putaran, state dikombinasikan kembali dengan *round key* untuk memastikan bahwa perubahan sebelumnya diamankan dan diperkuat melalui XOR terhadap kunci yang berbeda tiap putaran.

Setelah semua blok *file* diproses melalui tahapan tersebut, seluruh hasil enkripsi disimpan dalam bentuk *file* terenkripsi dengan ekstensi khusus, misalnya .rda.

P-ISSN 2085-4315
E-ISSN 2502-8332

2.4. Proses *Hashing* SHA-256 dengan *Salt*

SHA-256 adalah algoritma *hash* kriptografi yang termasuk dalam keluarga SHA-2 (*Secure Hash Algorithm* 2) dan dirancang oleh *National Security Agency* (NSA) Amerika Serikat [14]. Algoritma ini menghasilkan nilai *hash* sepanjang 256 bit (32 *byte*) dalam bentuk heksadesimal yang bersifat *one-way*, artinya nilai asli tidak dapat diperoleh kembali dari hasil *hash* [15]. Selain itu, SHA-256 juga tahan terhadap *collision* (dua input menghasilkan *hash* sama) dan preimage *attack* (menemukan input dari *output hash*), sehingga sangat andal untuk keperluan autentikasi [16]. *Salt* merupakan nilai acak yang ditambahkan ke *password* sebelum proses *hashing* dilakukan [17]. Tujuannya adalah untuk mencegah serangan *rainbow table* dan memperkuat sistem terhadap serangan *brute-force* [18]. Dengan menambahkan *salt*, dua *password* identik akan menghasilkan *hash* yang berbeda, sehingga meningkatkan kompleksitas dalam membongkar *hash* secara paksa.

Untuk menjaga keamanan autentikasi pengguna, pada penelitian ini menggunakan kombinasi algoritma SHA-256 dan teknik penambahan *salt*. SHA-256 termasuk dalam keluarga algoritma SHA-2 dan menghasilkan nilai *hash* tetap sepanjang 256 bit. Berikut adalah proses lengkapnya:

1) Pembuatan *Salt* Unik untuk Setiap Pengguna
   Setiap kali pengguna membuat akun atau mengganti kata sandi, sistem menghasilkan *salt* sepanjang 128 bit (16 byte). *Salt* ini dibuat menggunakan fungsi kriptografi openssl_random_pseudo_bytes(16) dan kemudian dikonversi menjadi representasi heksadesimal. *Salt* berfungsi sebagai nilai acak yang unik untuk setiap pengguna, mencegah serangan *rainbow table* dan membuat hasil *hash* tidak dapat ditebak meskipun *password*-nya sama.
2) Penggabungan *Salt* dan *Password*
   Salt yang telah dibuat akan digabungkan dengan *password* pengguna. Format penggabungan yang digunakan dalam sistem ini adalah *salt* + *password*, bukan sebaliknya. Hal ini bertujuan agar karakter awal hasil *hash* tetap berbeda secara signifikan meskipun *password* pengguna sama.
3) Proses *Hashing* dengan SHA-256
   Gabungan *salt* dan *password* kemudian di-*hash* menggunakan algoritma SHA-256. Fungsi *hashing* ini menghasilkan nilai 64 karakter dalam bentuk heksadesimal. SHA-256 bersifat *one-way*, sehingga hasil hash tidak dapat dikembalikan ke bentuk asli dan tahan terhadap *collision* serta *brute-force attack* [19].
4) Penyimpanan Format *Salt*:Hash
   Setelah proses hashing, nilai *salt* dan *hash* digabungkan dengan tanda pemisah (:) sehingga hasil akhir memiliki format seperti *salt*:hash. Nilai inilah yang disimpan dalam kolom *password* pada basis data. Format ini memungkinkan sistem untuk mengekstrak kembali *salt* saat proses *login* untuk dibandingkan dengan *hash* baru yang dihasilkan. Contoh Implementasi:

```
$salt = bin2hex(openssl_random_pseudo_bytes(16));
$hash = hash('sha256', $salt . $password);
$stored = $salt . ':' . $hash;
```

5) Proses Verifikasi *Login*
   Saat pengguna melakukan *login*, sistem akan mengambil *salt* dari database, menggabungkannya kembali dengan input *password*, melakukan hash, dan membandingkan hasilnya dengan *hash* yang tersimpan. Jika cocok, maka autentikasi dinyatakan berhasil.

Dengan proses ini, sistem tidak hanya mencegah penyimpanan *password* dalam bentuk *plaintext*, tetapi juga secara signifikan meningkatkan perlindungan terhadap serangan yang menargetkan kredensial pengguna.

2.5. Implementasi Sistem

Tahap implementasi merupakan proses mewujudkan desain sistem ke dalam aplikasi nyata yang dapat dioperasikan oleh pengguna [20]. Sistem diimplementasikan menggunakan bahasa pemrograman PHP dengan database MySQL. Antarmuka berbasis web memungkinkan pengguna mendaftar, masuk, mengunggah *file*, dan mengunduh hasil dekripsi. Setiap *file* yang diunggah akan diproses melalui modul AES-128, disimpan dalam direktori tertentu, dan metadata-nya dicatat dalam basis data. *Password* yang dimasukkan pengguna akan diproses melalui SHA-256 dengan *salt* unik. *Salt* dihasilkan menggunakan openssl_random_pseudo_bytes(16) yang dikonversi ke bentuk heksadesimal, kemudian dikombinasikan dengan *password* dan di-hash. Nilai akhir yang disimpan dalam database berbentuk format *salt*:hash.

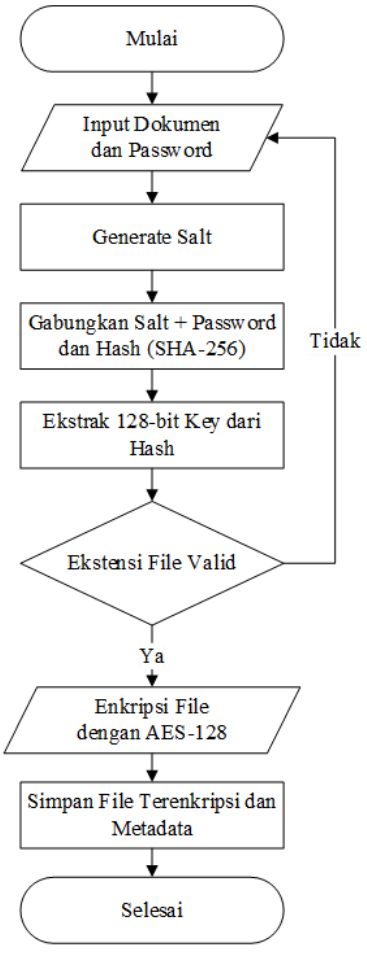
P-ISSN 2085-4315
E-ISSN 2502-8332

2.5. Pengujian dan Evaluasi Kinerja

Pengujian sistem mencakup aspek fungsionalitas dan performa enkripsi-dekripsi. Pengujian fungsional dilakukan untuk memastikan fitur *login*, unggah dokumen, enkripsi dengan AES-128, penyimpanan data terenkripsi, dan dekripsi oleh pengguna yang sah berjalan sesuai harapan. Selain itu, pengujian performa juga dilakukan terhadap 10 *file* dengan format dan ukuran berbeda (DOCX, PDF, XLSX) untuk mengukur efisiensi sistem. Parameter yang dinilai meliputi waktu proses enkripsi-dekripsi (dalam milidetik), perubahan ukuran file sebelum dan sesudah enkripsi, serta integritas hasil dekripsi dengan membandingkan isi file asli dan hasil dekripsi. Sebagai tambahan, dilakukan pula simulasi serangan *dictionary attack* dengan menggunakan daftar password umum untuk menguji ketahanan autentikasi berbasis hash SHA-256 dan *salt*. Hasilnya menunjukkan bahwa sistem berhasil menolak seluruh percobaan login tidak sah, yang menegaskan efektivitas mekanisme *salted hashing* dalam menghadapi serangan kamus.

3. HASIL DAN PEMBAHASAN

Penelitian ini mengembangkan sistem pengamanan dokumen digital berbasis web yang mengintegrasikan algoritma enkripsi AES-128 dan SHA-256 berbasis salt. AES-128 digunakan untuk mengenkripsi dan mendekripsi dokumen dalam blok 128-bit guna mencegah akses atau modifikasi oleh pihak yang tidak berwenang. Sementara itu, SHA-256 dengan teknik salting digunakan untuk mengamankan autentikasi pengguna secara unik, melindungi dari serangan *rainbow table* dan *dictionary attack*. Rangkaian proses pengembangan sistem ini ditunjukkan dalam flowchart pada Gambar 2.



Gambar 2. Flowchart Proses Enkripsi AES-128 dan SHA-256 Berbasis *Salt*

Flowchart pada Gambar 2 menunjukkan alur teknis dari proses enkripsi dokumen. Proses diawali dengan input dokumen dan *password* dari pengguna. Sistem kemudian menghasilkan nilai *salt* secara acak, yang

P-ISSN 2085-4315
E-ISSN 2502-8332

kemudian dikombinasikan dengan *password* dan di-*hash* menggunakan algoritma SHA-256. Hasil *hash* tersebut digunakan untuk dua hal: pertama, disimpan ke dalam basis data bersama dengan *salt* untuk keperluan autentikasi; dan kedua, sebagian hasil *hash* (16 *byte* awal) digunakan sebagai kunci enkripsi dokumen menggunakan algoritma AES-128. Sebelum proses enkripsi dijalankan, sistem terlebih dahulu melakukan validasi ekstensi *file* agar hanya format yang diperbolehkan (seperti .docx, .pdf, .xlsx, dll) yang diproses. Jika *file* tidak valid, proses akan dihentikan dan pengguna diminta mengunggah *file* yang sesuai. Jika valid, sistem akan melakukan proses enkripsi per blok menggunakan AES-128, dan menyimpan hasil enkripsi beserta metadata (termasuk *hash* dan *salt*) ke dalam basis data.

Untuk mengilustrasikan cara kerja sistem yang diusulkan, dilakukan studi kasus pada proses enkripsi dan autentikasi menggunakan data nyata. Studi ini bertujuan menunjukkan bagaimana algoritma AES-128 digunakan untuk mengenkripsi *file*, dan bagaimana SHA-256 berbasis *salt* digunakan untuk menjaga keamanan *password* pengguna. Studi ini dibagi menjadi tiga bagian: enkripsi dokumen menggunakan AES-128, proses *hashing password* dengan SHA-256 berbasis *salt*, serta verifikasi *password* saat dekripsi.

1) Enkripsi Dokumen dengan AES-128
   Pengguna mengunggah sebuah dokumen Excel bernama laporan-keuangan-2024.xlsx dengan ukuran 104 KB. *Password* yang dimasukkan adalah Rahasia123. Sistem kemudian melakukan *hashing* terhadap *password* menggunakan algoritma SHA-256 dan mengekstrak 16 *byte* pertama dari hasil *hash* tersebut sebagai kunci enkripsi AES-128. Contoh hasil kunci AES yang digunakan adalah:

   be8d6f14c1930fe3da342b12c13c4d77

   Setelah itu, *file* dibaca dalam blok 16 byte, dan setiap blok dienkripsi menggunakan AES-128. Hasil enkripsi disimpan sebagai *file* baru dengan ekstensi .rda, yaitu laporan-keuangan-2024.rda. Ukuran *file* terenkripsi bertambah sedikit menjadi 108 KB karena padding, dan proses enkripsi hanya memerlukan waktu sekitar 135 milidetik.
2) *Hashing Password* dengan SHA-256 dan *Salt*
   Pada saat proses penyimpanan *password*, sistem secara otomatis menghasilkan nilai *salt* secara acak sepanjang 16 byte, misalnya:

   9a4d16e4c1d2a98bb3e7e07e64a7e6dc

   Salt tersebut digabungkan dengan *password* yang dimasukkan (Rahasia123) dan kemudian diproses menggunakan SHA-256. Hasil hash-nya adalah:

   25ac6cd7426c4a7cf84c83156b0a1b83c5cfac97cbf36b6b1b7f37cd7b8e7e15

   Salt dan *hash* kemudian disimpan bersama dalam format:

   9a4d16e4c1d2a98bb3e7e07e64a7e6dc:25ac6cd7426c4a7c...

   Dengan pendekatan ini, meskipun dua pengguna memiliki *password* yang sama, hasil *hash* tetap berbeda karena nilai *salt*-nya unik.
3) Verifikasi *Password* saat Dekripsi
   Ketika pengguna ingin mendekripsi *file* laporan-keuangan-2024.rda, sistem akan mengambil data *salt*:*hash* yang tersimpan dalam database. *Salt* dan *hash* dipisahkan, lalu sistem melakukan *hashing* ulang terhadap *password* yang dimasukkan oleh pengguna dengan menggunakan *salt* yang sama. Contoh proses verifikasi:

```
list($salt, $storedHash) = explode(':', $stored);
$inputHash = hash('sha256', $salt . 'Rahasia123');
```

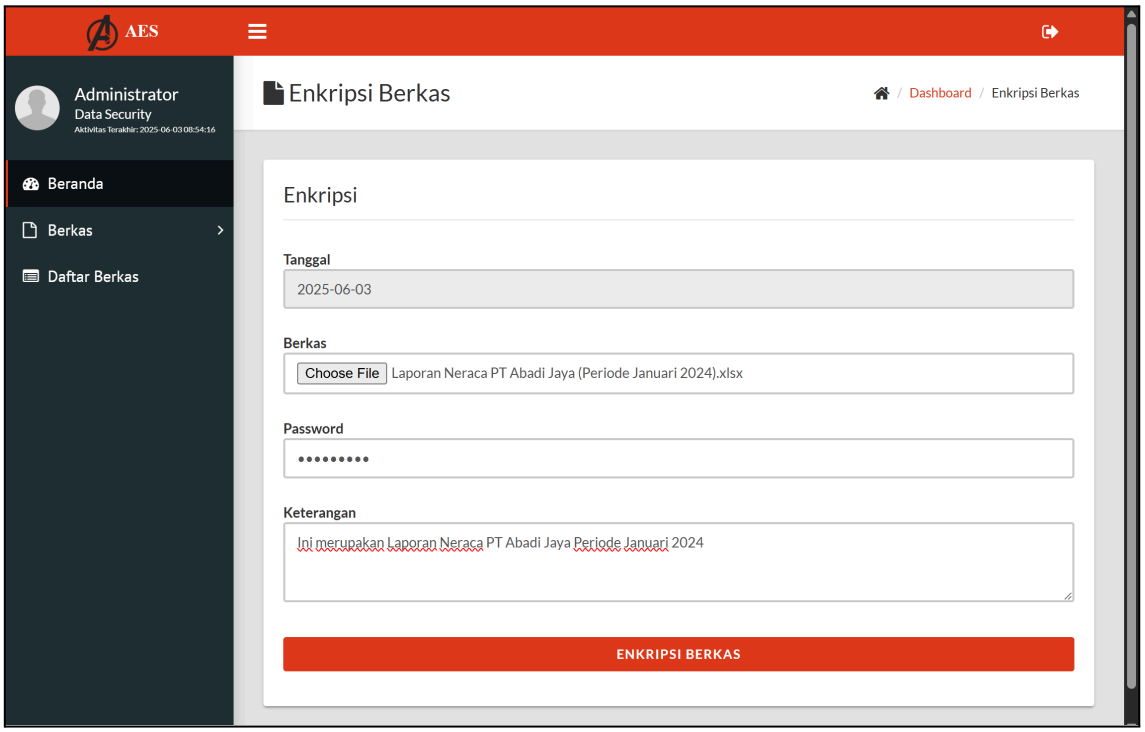
   Jika nilai $inputHash sama dengan $storedHash, maka proses dekripsi akan dijalankan. Jika tidak cocok, sistem akan menampilkan pesan bahwa *password* tidak valid.

Selanjutnya, langkah-langkah tersebut diimplementasikan dalam bentuk sistem berbasis web. Interaksi pengguna dimulai dari halaman *login*, di mana pengguna diminta untuk memasukkan *username* dan *password* yang telah terdaftar sebelumnya oleh administrator. Setelah proses autentikasi berhasil, pengguna diarahkan menuju halaman dashboard utama yang menampilkan fitur-fitur inti sistem, yaitu Enkripsi Berkas, Dekripsi Berkas, dan Daftar Berkas. Pada fitur Enkripsi Berkas, pengguna dapat melakukan proses enkripsi terhadap dokumen digital dengan cara mengisi sejumlah informasi yang tersedia dalam form, seperti tanggal enkripsi yang secara otomatis terisi oleh sistem, pemilihan berkas yang ingin dienkripsi melalui tombol *Choose File*, serta input *password* yang digunakan untuk proses *hashing* dan pembentukan kunci enkripsi. Di samping itu, pengguna juga dapat memberikan deskripsi atau keterangan tambahan mengenai isi *file*. Antarmuka fitur

JURNAL ILMIAH FIFO
Vol. 17 No. 1 (2025) 78-90
DOI: 10.22441/fifo.2025.v17i1.009

P-ISSN 2085-4315
E-ISSN 2502-8332

Enkripsi Berkas ini ditampilkan pada Gambar 3.



Gambar 3. Tampilan Fitur Enkripsi Berkas

Setelah semua data pada form enkripsi terisi dan tombol "Enkripsi Berkas" diklik, seperti yang ditampilkan pada Gambar 3, sistem akan mulai memproses *password* yang dimasukkan oleh pengguna. *Password* ini akan digabungkan dengan nilai *salt* yang dihasilkan secara acak, kemudian diproses menggunakan algoritma SHA-256 untuk menghasilkan *hash* yang unik. Hasil *hash* tersebut memiliki dua fungsi, yaitu: pertama, digunakan sebagai mekanisme autentikasi yang aman karena tidak menyimpan *password* dalam bentuk teks asli; dan kedua, 16 *byte* pertama dari *hash* digunakan untuk membentuk kunci enkripsi AES-128 yang kemudian dipakai untuk mengenkripsi isi *file*. *Password* yang telah melalui proses *hashing* dan penggabungan dengan *salt* tidak disimpan dalam bentuk asli, melainkan dalam format terenkripsi yang hanya dapat diverifikasi melalui proses *hash* ulang. Penyimpanan data *password* terenkripsi beserta informasi *file* lainnya dapat dilihat pada Gambar 4.

id_file	username	file_name_source	file_name_finish	file_url	file_size	tgl_upload	password	status	keterangan
44	admin	43385-laporan-rugi-laba-pt-abadi-jaya-2024.xlsx	96526-laporan-rugi-laba-pt-abadi-jaya-2024.rda	file_encrypt/96526-laporan-rugi-laba-pt-abadi-jaya...	8.61621	2025-06-02 21:03:56	50da4bb8b0c5051e	1	Laporan Rugi Laba pada PT Abadi Jaya
45	admin	14641-laporan-keuangan-perusahaan-dagang.xlsx	72640-laporan-keuangan-perusahaan-dagang.rda	file_encrypt/72640-laporan-keuangan-perusahaan-dag...	10.2334	2025-06-03 08:57:16	aee5d26984b6aef2	1	Laporan Keuangan Perusahaan Dagang PT Abadi Jaya
46	admin	81096-surat-tugas.pdf	70694-surat-tugas.rda	file_encrypt/70694-surat-tugas.rda	335.158	2025-06-03 09:00:19	31821ac94eae057c	1	Surat Tugas
47	admin	75519-58-surat-keterangan.docx	66060-58-surat-keterangan.rda	file_encrypt/66060-58-surat-keterangan.rda	262.836	2025-06-03 09:01:44	cd8e6a1ff3b3a653	1	Surat Keterangan
48	admin	3592-laporan-arus-kas-pt-abadi-jaya-(periode-janua...	8380-laporan-arus-kas-pt-abadi-jaya-(periode-janua...	file_encrypt/8380-laporan-arus-kas-pt-abadi-jaya-(...	9.04785	2025-06-03 09:02:23	d0cec2abaf1dfbb2	1	Laporan Arus Kas
49	admin	18127-laporan-neraca-pt-abadi-jaya-(periode-januar...	78326-laporan-neraca-pt-abadi-jaya-(periode-januar...	file_encrypt/78326-laporan-neraca-pt-abadi-jaya-(p...	8.83105	2025-06-03 09:02:42	4eb13516052cadb8	1	Laporan Neraca

Gambar 4. Tampilan Tabel *File* pada Database setelah Proses Enkripsi

Pada Gambar 4 terlihat bahwa setiap entri mencatat nama *file* asli, nama *file* hasil enkripsi dengan ekstensi .rda, lokasi penyimpanan *file* terenkripsi, ukuran *file* dalam KB, waktu unggah, serta kolom *password* yang telah disimpan dalam bentuk hash. Kolom *password* pada tabel ini berisi hasil penggabungan *salt* dan *hash* SHA-256

JURNAL ILMIAH FIFO
Vol. 17 No. 1 (2025) 78-90
DOI: 10.22441/fifo.2025.v17i1.009

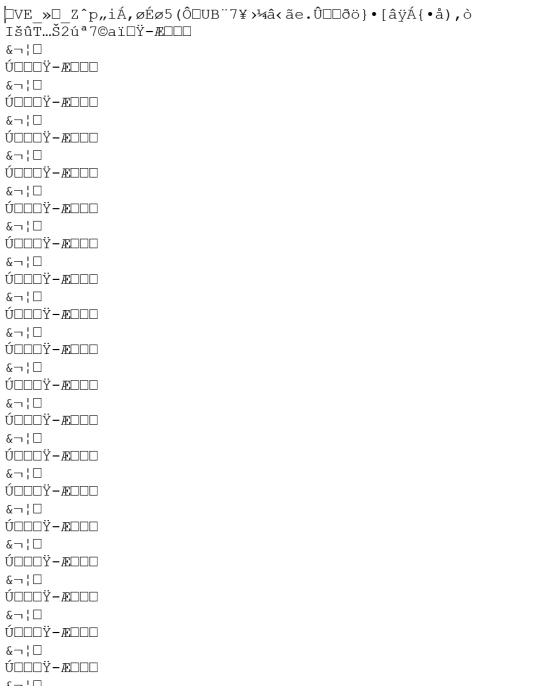
P-ISSN 2085-4315
E-ISSN 2502-8332

yang telah dihasilkan secara otomatis oleh sistem. Hal ini memastikan bahwa setiap *file* memiliki tingkat keamanan yang tinggi karena hanya dapat didekripsi menggunakan *password* asli yang benar, serta tidak dapat direkonstruksi balik dari database.

Hasil dari proses enkripsi adalah sebuah *file* dengan format khusus .rda yang berfungsi sebagai bentuk terenkripsi dari dokumen asli. *File* ini tidak dapat dibuka atau dibaca secara langsung oleh pengguna biasa karena seluruh isi *file* telah disamarkan melalui proses enkripsi. Dengan demikian, dokumen tidak dapat diakses oleh pihak yang tidak memiliki otorisasi, sehingga tingkat kerahasiaan dan integritas informasi tetap terjaga. Contoh perbandingan antara dokumen asli dan *file* hasil enkripsi ditampilkan pada Gambar 5.

PT. Abadi Jaya	
Neraca	
Periode 31 Januari 2024	
Aktiva	
Aktiva Lancar	
Kas dan bank	
Kas:	**74.812.500**
Kas IDR	30.090.000
Kas USD	33.900.000
Kas SGD	10.822.500
Bank:	**2.931.549.587,55**
Bank 1 (IDR)	319.434.889,2
Bank 2 (IDR)	2.378.089.691,55
Bank 3 (USD)	210.180.000
Bank 4 (SGD)	14.845.006,8
Jumlah Kas dan Bank	**3.006.362.090,55**
Piutang Dagang	
Jumlah Piutang Dagang	0
Persediaan	
Persediaan Barang Dagang:	**1.471.430.501,25**
Persediaan Bahan Tekstil	998.390.5-1,25
Persediaan Perkakas	56.166.000
Persediaan Elektronik	250.134.000
Persediaan Furniture	166.740.000
Jumlah Persediaan	1.471.430.501,25
Aktiva Lancar Lainnya:	
Biaya dibayar di muka	258.000.000
Sewa dibayar di muka	108.000.000
Asuransi dibayar di muka	150.000.000
PPn Masukan	289.170.656,4
Proyek dalam Proses	33.963.289,5
Jumlah Aktiva Lancar lainnya	**581.133.948,9**
Jumlah Aktiva Lancar	**5.058.926.534,7**

(a)



(b)

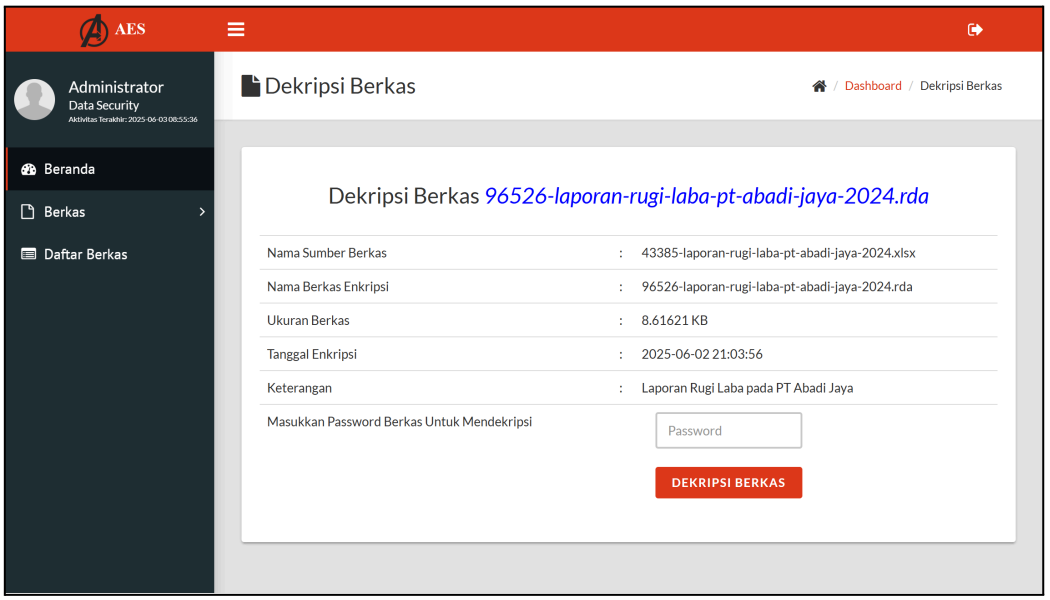
Gambar 5. (a) *File* Asli Format Excel, (b) *File* Hasil Enkripsi Format .rda

Gambar 5 memperlihatkan perbedaan mencolok antara isi *file* sebelum dan sesudah proses enkripsi. Gambar 5(a) menampilkan *file* Excel yang berisi laporan neraca PT Abadi Jaya dalam format yang dapat dibaca dan dipahami dengan mudah. Sebaliknya, Gambar 5(b) menunjukkan isi *file* setelah dienkripsi dengan AES-128, yang tampak sebagai rangkaian karakter acak dan simbol yang tidak memiliki makna secara visual. Hal ini menunjukkan bahwa algoritma AES-128 berhasil menyamarkan seluruh isi dokumen dengan efektif dan melindungi informasi dari akses tidak sah. Efektivitas dari proses enkripsi juga tercermin dalam peningkatan keamanan *file* tanpa menyebabkan perubahan signifikan pada ukuran *file*. Selain itu, sistem tetap mempertahankan struktur *file* sehingga memungkinkan proses dekripsi berjalan sempurna apabila *password* yang sesuai dimasukkan.

Proses dekripsi sendiri dapat dilakukan melalui fitur Dekripsi Berkas. Fitur ini memungkinkan pengguna untuk memilih *file* .rda yang telah terenkripsi, kemudian memasukkan *password* yang sama seperti saat proses enkripsi. Setelah *password* dimasukkan, sistem akan mengambil nilai *salt* dan *hash* yang telah disimpan dalam database, kemudian melakukan *hashing* ulang terhadap input pengguna. Jika nilai *hash* yang dihasilkan sesuai dengan yang tersimpan, maka sistem akan melanjutkan ke proses dekripsi dan mengembalikan *file* ke bentuk aslinya secara utuh dan dapat digunakan kembali. Dalam sistem ini, password login dan password enkripsi merupakan dua entitas yang terpisah. Password *login* digunakan untuk autentikasi, sementara *password* enkripsi menjadi input kunci AES-128 dan tidak disimpan oleh sistem. Jika *password* enkripsi hilang, dokumen tidak dapat didekripsi. Saat ini, sistem belum mendukung *reset password* enkripsi secara otomatis, namun *reset login* dapat dibantu oleh administrator melalui *database*. Tampilan antarmuka fitur dekripsi dapat dilihat pada Gambar 6, yang menampilkan informasi detail *file* seperti nama asli *file*, nama *file* terenkripsi, ukuran *file*, tanggal enkripsi, dan kolom input *password*.

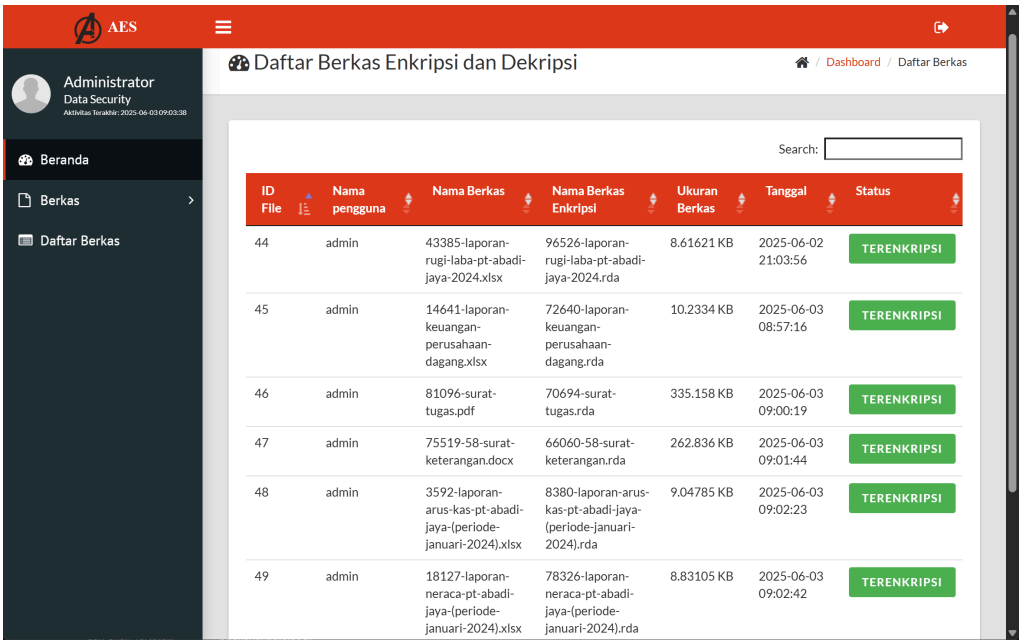
JURNAL ILMIAH FIFO
Vol. 17 No. 1 (2025) 78-90
DOI: 10.22441/fifo.2025.v17i1.009

P-ISSN 2085-4315
E-ISSN 2502-8332



Gambar 6. Tampilan Fitur Dekripsi Berkas

Tampilan antarmuka untuk melakukan dekripsi diperlihatkan pada Gambar 6, yang menyajikan informasi lengkap mengenai *file* seperti nama dokumen asli, nama *file* hasil enkripsi, ukuran, tanggal enkripsi, serta kolom input *password* untuk autentikasi. Sistem juga menyediakan fitur Daftar Berkas yang menampilkan seluruh *file* yang telah dienkripsi oleh pengguna. Fitur ini menyajikan informasi penting seperti ID *file*, nama pengguna, nama *file* asli, nama *file* hasil enkripsi, ukuran *file*, tanggal proses, serta status proses enkripsi. Fitur ini memudahkan pengguna maupun administrator dalam memantau histori dan status dokumen digital yang telah diamankan melalui sistem. Gambar 7 menunjukkan tampilan dari halaman daftar *file* yang telah dienkripsi dalam sistem.



Gambar 7. Tampilan Daftar Berkas Terenkripsi dalam Sistem

Proses selanjutnya yaitu pengujian sistem yang mencakup dua aspek utama, yaitu pengujian fungsionalitas dan pengujian performa enkripsi-dekripsi. Pengujian ini bertujuan untuk memastikan bahwa setiap fitur utama dalam sistem bekerja sesuai dengan yang diharapkan, serta mengevaluasi efisiensi proses enkripsi dan dekripsi

JURNAL ILMIAH FIFO
Vol. 17 No. 1 (2025) 78-90
DOI: 10.22441/fifo.2025.v17i1.009
P-ISSN 2085-4315
E-ISSN 2502-8332

terhadap berbagai jenis dokumen digital. Pengujian dilakukan dengan berbagai skenario mulai dari proses *login*, pengunggahan *file*, enkripsi, penyimpanan hasil enkripsi, hingga dekripsi dan verifikasi kesesuaian data. Hasil pengujian fungsionalitas ditunjukkan pada Tabel 1.

Tabel 1. Hasil Pengujian Fungsionalitas Sistem

No	Skenario Pengujian	Langkah yang Dilakukan	Hasil yang Diharapkan	Status
1	*Login* pengguna valid	Pengguna memasukkan *username* dan *password* yang benar	Berhasil masuk ke sistem	Berhasil
2	*Login* pengguna tidak valid	Pengguna memasukkan *password* salah	Gagal *login* dan muncul notifikasi kesalahan	Berhasil
3	Unggah dokumen	Pengguna mengunggah *file* .docx	*File* berhasil diunggah ke server	Berhasil
4	Enkripsi *file*	Klik tombol Encrypt pada dokumen yang telah diunggah	*File* terenkripsi dan disimpan dalam format .rda	Berhasil
5	Simpan hasil enkripsi	Sistem menyimpan hasil enkripsi dan metadata	*File* dan metadata tersimpan di basis data	Berhasil
6	Dekripsi dengan *password* valid	Pengguna memilih *file* terenkripsi dan memasukkan *password* yang benar	*File* berhasil didekripsi dan identik dengan *file* asli	Berhasil
7	Dekripsi dengan *password* salah	Pengguna memasukkan *password* salah saat dekripsi	Sistem menolak proses dekripsi dan menampilkan peringatan	Berhasil
8	*Hash password* pengguna	Simpan *password* pengguna baru dengan *hash* SHA-256 berbasis *salt*	*Hash* disimpan dalam format *salt*:hash	Berhasil

Tabel 1 menunjukkan bahwa seluruh skenario pengujian sistem berhasil dijalankan dengan output yang sesuai. Pengujian *login* valid dan tidak valid menghasilkan autentikasi yang akurat. Fitur unggah dokumen dapat menerima berbagai format *file* (.docx, .pdf, .xlsx), dan *file* berhasil dienkripsi menggunakan algoritma AES-128 serta disimpan dalam format khusus .rda. Seluruh metadata *file* juga berhasil direkam dalam database, termasuk *password* yang telah di-*hash* menggunakan SHA-256 berbasis *salt*. Saat proses dekripsi dilakukan, sistem mampu memverifikasi *password* pengguna dan mengembalikan *file* ke bentuk aslinya apabila *password* benar, serta menolak proses dekripsi jika *password* salah. Pengujian ini membuktikan bahwa mekanisme autentikasi berbasis *salt* dan SHA-256 bekerja sesuai harapan.

Selanjutnya, pengujian performa dilakukan terhadap sepuluh *file* uji coba dengan format dan ukuran yang bervariasi. Hasilnya ditunjukkan pada Tabel 2.

Tabel 2. Hasil Pengujian Performa Enkripsi dan Dekripsi *File*

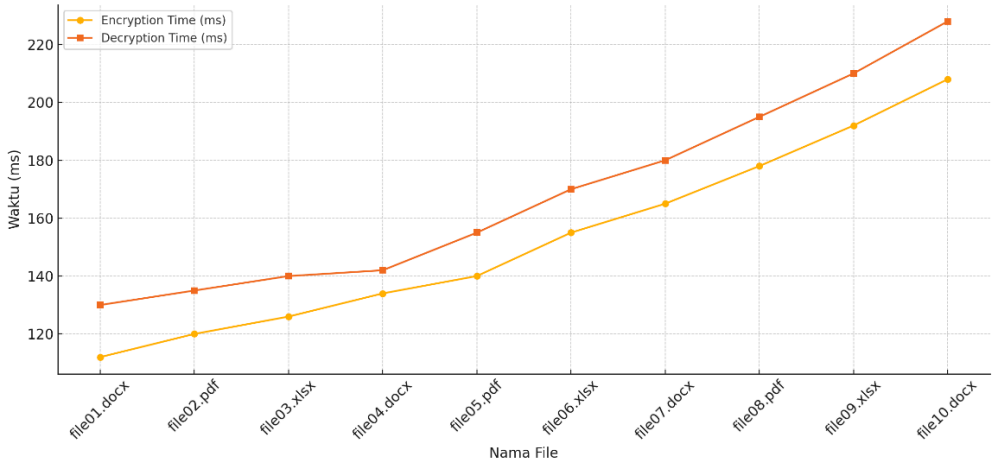
No	Nama *File*	Format	Ukuran Asli (KB)	Ukuran Enkripsi (KB)	Waktu Enkripsi (ms)	Waktu Dekripsi (ms)	Kesesuaian *File*
1	file01.docx	DOCX	128	132	112	130	Sesuai
2	file02.pdf	PDF	256	260	120	135	Sesuai
3	file03.xlsx	XLSX	512	516	126	140	Sesuai
4	file04.docx	DOCX	768	772	134	142	Sesuai
5	file05.pdf	PDF	1024	1028	140	155	Sesuai
6	file06.xlsx	XLSX	2048	2052	155	170	Sesuai

P-ISSN 2085-4315
E-ISSN 2502-8332

7	file07.docx	DOCX	3072	3076	165	180	Sesuai
8	file08.pdf	PDF	4096	4100	178	195	Sesuai
9	file09.xlsx	XLSX	5120	5124	192	210	Sesuai
10	file10.docx	DOCX	6144	6148	208	228	Sesuai

Tabel 2 menunjukkan bahwa proses enkripsi dan dekripsi berlangsung cukup cepat, dengan waktu enkripsi berkisar antara 112 ms hingga 208 ms, dan waktu dekripsi antara 130 ms hingga 228 ms. Ukuran *file* sebelum dan sesudah proses enkripsi hanya mengalami peningkatan yang sangat kecil, yaitu ±4 KB, yang disebabkan oleh struktur blok tetap dalam algoritma AES-128 (block padding). Semua *file* yang telah didekripsi menunjukkan hasil yang identik dengan *file* asli, baik dari segi ukuran maupun isi, sehingga dinyatakan sesuai.

Berdasarkan hasil pengujian terhadap performa enkripsi dan dekripsi terhadap sepuluh file dengan format dan ukuran yang berbeda, diperoleh waktu proses yang bervariasi sesuai dengan ukuran file yang diuji. Semakin besar ukuran file, maka waktu proses enkripsi maupun dekripsi cenderung meningkat. Grafik berikut menunjukkan perbandingan waktu rata-rata yang dibutuhkan sistem untuk melakukan enkripsi dan dekripsi terhadap masing-masing file. Hasil lengkap pengujian ini ditunjukkan pada Gambar 8.



Gambar 8. Perbandingan Waktu Enkripsi Dan Dekripsi (Dalam Milidetik) Terhadap Berbagai *File* Uji

Gambar 8 menunjukkan grafik perbandingan waktu enkripsi dan dekripsi untuk setiap file uji. Grafik ini menunjukkan bahwa waktu dekripsi secara konsisten sedikit lebih tinggi dibandingkan enkripsi, namun keduanya tetap berada dalam rentang waktu yang efisien (di bawah 230 ms).

Simulasi uji keamanan terhadap sistem autentikasi dilakukan sebagai pengujian lanjutan, dengan menggunakan pendekatan *dictionary attack* berbasis daftar password umum. Serangan ini meniru upaya peretas yang mencoba mencocokkan hash password menggunakan kombinasi antara nilai *salt* (yang didapatkan secara tidak sah dari database) dan beberapa password yang sering digunakan.

Tabel 3. Hasil Pengujian *Dictionary Attack*

Password yang Diuji	**Hash SHA-256 + Salt (disingkat)**	**Kecocokan dengan Hash Asli**	**Keterangan**
password	2deec334899dceaf...	Tidak cocok	Gagal login
123456	36cade4c3a9373a5...	Tidak cocok	Gagal login
admin	d03f9885dac6a16e...	Tidak cocok	Gagal login
welcome	6fae5c957d4aff09...	Tidak cocok	Gagal login
Rahasia123	de69e2ebc46c013f...	Tidak cocok	Gagal login
qwerty	41c306210a47ae0b...	Tidak cocok	Gagal login

Hasil simulasi menunjukkan bahwa tidak satu pun dari password dalam daftar tersebut berhasil menghasilkan hash yang cocok dengan hash asli yang disimpan dalam sistem. Hal ini menunjukkan bahwa

penggunaan *salt* yang unik untuk setiap pengguna berhasil memperkuat algoritma hash SHA-256 dan mencegah pemanfaatan ulang hash dari password umum. Salt mencegah dua pengguna dengan password yang sama menghasilkan hash yang sama, serta membuat metode precomputed seperti *rainbow table* menjadi tidak efektif. Dengan demikian, meskipun peretas memperoleh data *salt* dan *hash* dari database, mereka tetap harus melakukan brute-force secara individual terhadap setiap akun, yang secara komputasional jauh lebih berat.

Berdasarkan dari beberapa pengujian yang dilakukan, membuktikan bahwa algoritma AES-128 yang digunakan mampu melakukan enkripsi terhadap berbagai format *file* dengan efisien dan tetap menjaga integritas data. Begitu pula, penerapan SHA-256 berbasis *salt* berhasil memastikan keamanan *password* dan mencegah penggunaan kembali *hash* yang sama pada dua akun berbeda meskipun menggunakan *password* yang sama.

Namun demikian, terdapat beberapa keterbatasan yang perlu diperhatikan. Pertama, pengujian keamanan yang dilakukan dalam penelitian ini masih terbatas pada simulasi *dictionary attack* menggunakan daftar *password* umum, dan belum mencakup jenis serangan lain seperti *SQL injection*, *rainbow table*, atau *side-channel attack*. Kedua, waktu enkripsi dan dekripsi masih dipengaruhi oleh performa server lokal, sehingga performa sistem dapat berbeda jika dijalankan pada lingkungan server produksi dengan beban yang lebih besar. Selain itu, enkripsi dilakukan per blok tetap (16 *byte*) sehingga tidak optimal untuk *file* berukuran sangat kecil karena tetap mengalami *padding*.

4. KESIMPULAN

Penelitian ini berhasil mengembangkan sistem pengamanan dokumen digital berbasis web dengan mengombinasikan algoritma enkripsi simetris AES-128 dan *hashing* SHA-256 berbasis *salt*. Sistem ini dirancang untuk memberikan dua lapisan keamanan: pertama, enkripsi konten *file* dengan AES-128 untuk menjaga kerahasiaan informasi; dan kedua, perlindungan terhadap autentikasi pengguna melalui mekanisme *hash* SHA-256 yang disertai *salt*, guna mencegah serangan berbasis *rainbow table* maupun *precomputed hash*. Hasil implementasi menunjukkan bahwa sistem mampu mengenkripsi berbagai format dokumen seperti DOCX, PDF, dan XLSX secara efektif. Seluruh proses enkripsi menghasilkan *file* .rda yang tidak dapat dibaca secara langsung, dan *file* tersebut dapat dikembalikan ke bentuk aslinya melalui proses dekripsi yang valid. Pengujian fungsional dan performa menunjukkan bahwa sistem bekerja secara konsisten, dengan waktu enkripsi dan dekripsi yang efisien dan peningkatan ukuran *file* yang minimal. Seluruh skenario pengujian berhasil dijalankan, dan hasil dekripsi dinyatakan sesuai dengan *file* asli. Kontribusi utama dari penelitian ini adalah integrasi metode kriptografi modern dalam sistem berbasis web yang ramah pengguna, serta penggunaan teknik *salted hashing* yang memperkuat sistem autentikasi pengguna. Sistem ini juga menyediakan fitur manajemen dokumen yang terstruktur, memungkinkan pengguna untuk mengenkripsi, menyimpan, dan mendekripsi *file* dengan antarmuka yang intuitif. Namun demikian, sistem ini memiliki beberapa batasan, antara lain belum tersedianya fitur *reset password* enkripsi, belum dilakukan pengujian terhadap serangan lanjutan seperti SQL *injection* dan *side-channel attack*, serta ketergantungan pada infrastruktur server lokal. Untuk penelitian selanjutnya, sistem dapat dikembangkan dengan menambahkan fitur verifikasi integritas *file*, autentikasi dua faktor, serta pelacakan aktivitas pengguna untuk meningkatkan aspek keamanan dan forensik digital. Sistem juga dapat diperluas untuk mendukung dekripsi *batch*, pemantauan akses *file*, serta kompatibilitas dengan penyimpanan *cloud*.

DAFTAR PUSTAKA

[1] T. Darmansah, G. A. Pasaribu, D. Juliani, S. N. Pulungan, and C. A. Pangolangan, “Optimalisasi Sistem Informasi Administrasi Digital Untuk Meningkatkan Efisiensi Layanan dan Keamanan Informasi Organisasi,” *Socius J. Penelit. Ilmu-ilmu Sos.*, vol. 2, no. 11, pp. 108–112, 2025.

[2] A. Sulistiawati and R. Firdaus, “Perancangan Sistem Informasi Manajemen Berbasis Teknologi Blockchain Untuk Optimalisasi Keamanan Dokumen Digital,” *JICN J. Intelek dan Cendikiawan Nusant.*, vol. 1, no. 3, pp. 4142–4148, 2024.

[3] I. Priambudi and M. Mufti, “Implementasi Kriptografi Dengan Metode AES-128 Untuk Pengamanan File Berbasis Web Pada SMP Yapipa,” *SKANIKA Sist. Komput. dan Tek. Inform.*, vol. 6, no. 1, pp. 22–31, 2023, doi: 10.36080/skanika.v6i1.2997.

[4] A. Tumanggor, H. Rumapea, and A. Silalahi, “Implementasi Algoritma Advance Encryption Standard (AES) Pada Keamanan Dokumen Keuangan (Studi Kasus : CV.Multikreasi Bersama),” *Methotika J. Ilm. Tek. Inform.*, vol. 3, no. 1, pp. 83–90, 2023.

[5] H. D. Novianti and A. T. Hidayat, “Implementasi Kriptografi Advanced Encryption Standard 128 Bit Dalam Pengamanan Data Keuangan Kas (Studi Kasus: Masjid Al-Ikhlas Trini Sleman D.I.Yogyakarta),” *J. Komput. dan Teknol.*, vol. 2, no. 1, pp. 27–34, 2023.

[6] A. I. Suranta and D. V. S. Y. Sakti, “Penerapan Algoritma AES (Advance Encryption Standart) 128 untuk Enkripsi Dokumen di PT. Gunung Geulis Elok Abadi,” *SKANIKA Sist. Komput. dan Tek. Inform.*, vol. 5,

P-ISSN 2085-4315
E-ISSN 2502-8332

no. 1, pp. 1–10, 2022, doi: 10.36080/skanika.v5i1.2118.

[7] B. O. P. I. Irawan, M. Tahir, N. Ayu, W. Y. C. Windrastuti, D. Mulaikah, and A. B. M. S. Wachid, "Implementasi Kriptografi Pada Keamanan Data Menggunakan Algoritma Advance Encryption Standard (AES)," *J. Simantec*, vol. 11, no. 2, pp. 167–174, 2023.

[8] I. Saputra and S. D. Nasution, "Perbandingan Performa Algoritma Md5 Dan Sha-256 Dalam Membangkitkan Identitas File," *J. Sains Komput. Inform.*, vol. 6, no. 1, pp. 172–187, 2022.

[9] J. Hutagalung, P. S. Ramadhan, and S. J. Sihombing, "Keamanan Data Menggunakan Secure Hashing Algorithm (SHA)-256 dan Rivest Shamir Adleman (RSA) pada Digital Signature," *J. Teknol. Inf. dan Ilmu Komput.*, vol. 10, no. 6, pp. 1213–1222, 2023, doi: 10.25126/jtiik.1067319.

[10] I. Ahmad, A. T. Prastowo, E. Suwarni, and R. I. Borman, "Pengembangan Aplikasi Online Delivery Sebagai Upaya Untuk Membantu Peningkatan Pendapatan," *JMM (Jurnal Masy. Mandiri)*, vol. 5, no. 6, pp. 4–12, 2021.

[11] A. Aprizald, M. A. Hasan, and D. Setiawan, "Aplikasi Keamanan Data Berbasis Web Menggunakan Algoritma AES 128 Untuk Enkripsi Dan Dekripsi Data," *JEKIN - J. Tek. Inform.*, vol. 2, no. 2, pp. 85–95, 2023, doi: 10.58794/jekin.v2i2.225.

[12] M. B. Aryanto, M. Tahir, S. I. Devita, Z. N. Mustofa, Q. Ainiyah, and S. Sundoro, "Implementasi Enkrip Dan Dekrip File Menggunakan Metode Advance Encryption Standard (AES-128)," *J. Ilm. Sist. Inf. dan Ilmu Komput.*, vol. 3, no. 1, pp. 89–104, 2023, doi: 10.55606/juisik.v3i1.434.

[13] D. Widyawan and I. Imelda, "Pengamanan File Menggunakan Kriptografi Dengan Metode AES-128 Berbasis Web Di Komite Nasional Keselamatan Transportasi," *SKANIKA Sist. Komput. dan Tek. Inform.*, vol. 4, no. 1, pp. 15–22, 2021, doi: 10.36080/skanika.v4i1.2216.

[14] N. Sitorus, J. S. G. Sinaga, and S. L. Samosir, "Analisis Kinerja Algoritma Hash pada Keamanan Data: Perbandingan Antara SHA-256, SHA-3, dan Blake2," *J. Quancom*, vol. 2, no. 2, pp. 9–16, 2024.

[15] N. A. Yulianto, M. Ridwan, and A. T. Wibowo, "Analisis Kinerja Algoritma MD5, SHA-256, dan Base62 dalam Sistem Pemendekan URL," *JUMISTIK J. Manaj. Inform. Sist. Inf. dan Teknol. Inform.*, vol. 3, no. 2, pp. 270–276, 2024, doi: 10.70247/jumistik.vi2.113.

[16] S. Nainggolan, "Rekayasa Teknik Informatika dan Informasi Implementasi Algoritma SHA-256 Pada Aplikasi Duplicate Document Scanner," *RESOLUSI Rekayasa Tek. Inform. dan Inf.*, vol. 2, no. 5, pp. 201–213, 2022.

[17] R. F. Maulana, A. Y. Wicaksono, K. M. D. Pertiwi, and M. D. Fauzi, "Kombinasi Timestamp Nonce dan Nilai Salt pada Autentikasi Single Sign-On," *Briliant J. Ris. dan Konseptual*, vol. 8, no. 4, pp. 1049–1061, 2023, doi: 10.28926/briliant.v8i4.1389.

[18] D. Ramalinda, J. Jayadi, and A. R. Raharja, "Strategi Perlindungan Data Menggunakan Sistem Kriptografi Dalam Keamanan Informasi," *J. Int. Multidiscip. Res.*, vol. 2, no. 6, pp. 665–671, 2024, doi: 10.62504/jimr679.

[19] C. Agusniar, K. M. Siregar, and R. Rasyid, "Sistem Booking Makeup Artist Berbasis Web Menggunakan Algoritma SHA-256," *JIP (Jurnal Inform. Polinema)*, vol. 11, no. 2, pp. 185–194, 2024.

[20] Y. Fernando, R. Napianto, and R. I. Borman, "Implementasi Algoritma Dempster-Shafer Theory Pada Sistem Pakar Diagnosa Penyakit Psikologis Gangguan Kontrol Impuls," *Insearch Inf. Syst. Res. J.*, vol. 2, no. 2, pp. 46–54, 2022.