Penerapan Keamanan File Pada Shared Data Storage Dengan Kombinasi Chacha20 Dan Modifikasi AES

1st Erica Dyah Ayu Septiani
Informatika
Telkom University
Surabaya, Indonesia
ericadyah@student.telkomuniversity.ac.id

2nd Rizky Fenaldo Maulana
Informatika
Telkom University
Surabaya, Indonesia
rizkyfenaldo@telkomuniversity.ac.id

3rd Tanzilal Mustaqim
Informatika
Telkom University
Surabaya, Indonesia
tanzilal@telkomuniversity.ac.id

***Abstrak* — Dalam era digital, kebutuhan akan sistem berbagi file yang aman semakin penting, terutama pada lingkungan jaringan berbasis Linux. Penelitian ini mengusulkan sistem keamanan file menggunakan kombinasi algoritma enkripsi ChaCha20 dan modifikasi Advanced Encryption Standard (AES) melalui metode AddRoundKey. Sistem diimplementasikan pada protokol Server Message Block (SMB) versi 1 berbasis Samba dan dilengkapi antarmuka web client berbasis Flask untuk memfasilitasi unggah, unduh, serta pengelolaan file secara terenkripsi. Proses enkripsi dilakukan secara berlapis: data pertama-tama dienkripsi menggunakan AES modifikasi, kemudian hasilnya dienkripsi kembali dengan ChaCha20 untuk memperkuat keamanan terhadap serangan dan penyadapan. Pengujian dilakukan terhadap berbagai jenis file teks dan gambar berukuran 10KB hingga 100MB untuk mengevaluasi performa sistem dari segi waktu enkripsi-dekripsi, kecepatan transfer, dan integritas data. Pengujian keamanan dilakukan melalui checksum dan analisis lalu lintas menggunakan Wireshark guna memastikan bahwa data tetap terlindungi selama proses transfer. Hasil pengujian menunjukkan bahwa rata-rata waktu enkripsi mencapai 0,02583 detik dan waktu dekripsi sebesar 0,02271 detik untuk file berukuran 1MB, serta file hasil dekripsi terbukti identik dengan file asli berdasarkan nilai hash checksum. Selain itu, histogram hasil enkripsi menunjukkan pola distribusi acak yang membuktikan efektivitas pengacakan data. Hasil tersebut menunjukkan bahwa kombinasi algoritma ini mampu memberikan perlindungan yang lebih baik dibandingkan metode Samba standar, serta mempertahankan integritas dan kerahasiaan file selama proses berbagi data. Sistem ini membuktikan efektivitas penerapan kriptografi hibrida dalam meningkatkan keamanan pada lingkungan file sharing berbasis protokol SMBv1.**

***Kata kunci*— ChaCha20, AES, addroundkey, kriptografi hibrida, samba, keamanan file**

I. PENDAHULUAN

Dengan berkembangnya teknologi dan semakin populernya informasi, kebutuhan akan media penyimpanan data yang besar juga semakin meningkat. Oleh karena itu, muncul masalah terkait bagaimana cara mendapatkan media penyimpanan yang lebih besar dan aman untuk menampung file, salah satunya melalui teknologi berbagi file dengan menggunakan protokol Server Message Block (SMB)(Wijaya et al., 2022). Protokol SMB memungkinkan layanan berbagi file yang memudahkan pengelolaan penyimpanan data, sementara Samba berfungsi sebagai perangkat lunak untuk implementasi SMB, yang menawarkan beberapa keunggulan, seperti sifatnya yang gratis, kemampuannya untuk terhubung langsung ke jaringan, serta kompatibilitas dengan berbagai platform yang menjadikannya pilihan populer (Insanudin et al., 2024).

Alasan utama penggunaan Linux dalam berbagi file, termasuk menggunakan protokol SMB melalui perangkat lunak Samba, adalah stabilitasnya yang tinggi, Linux dirancang sebagai sistem operasi multiuser, yang memastikan bahwa hanya pengguna dengan akses khusus seperti "root" atau "administrator" yang dapat mengakses kernel dan melakukan perubahan pada sistem. Hal ini meningkatkan tingkat keamanannya. Linux juga memiliki perlindungan yang baik terhadap virus dan malware. Meskipun tidak ada sistem yang sepenuhnya aman, sifat open source Linux memungkinkan komunitas pengguna untuk memperbaiki celah keamanan (vulnerabilities) dengan cepat. Semua proses, folder, dan file di Linux dapat dikontrol dan diawasi, sehingga transparansi ini menjadikannya pilihan yang sangat baik bagi para administrator. Dengan memanfaatkan protokol SMB, pengguna dapat mengelola dan membagikan data dengan mudah, aman, dan kompatibel dengan berbagai sistem operasi (Rezaldy & Ropianto, 2023).

Dalam penelitian ini, saya menggunakan protokol SMBv1. Meskipun protokol SMBv1 menawarkan banyak keunggulan, akan tetapi SMBv1 mempunyai kelemahan yang signifikan, terutama terkait dengan keamanan dan efisiensi jaringan. Protokol ini tidak menyediakan mekanisme enkripsi bawaan, sehingga data yang dikirimkan melalui SMBv1 dapat dengan mudah disadap atau dimodifikasi (Vinodhini et al., 2021). Kriptografi menjadi solusi untuk melindungi kerahasiaan data yang dikirim, dengan mengubah pesan asli (plaintext) menjadi bentuk yang tidak dapat dibaca (ciphertext) melalui enkripsi menggunakan kunci yang hanya diketahui oleh pengirim dan penerima. Dengan demikian, pihak lain yang tidak memiliki kunci tidak akan dapat membaca isi pesan tersebut (Anggraeni Eka Putri et al., 2021). Untuk mengatasi masalah keamanan ini, kombinasi algoritma enkripsi seperti AES (Advanced Encryption Standard) dan ChaCha20 digunakan untuk meningkatkan keamanan data selama transfer file.

AES merupakan algoritma enkripsi block cipher yang efektif dan efisien dalam mengenkripsi data dengan panjang kunci yang bervariasi (128 bit, 192 bit, dan 256 bit). Pemilihan panjang kunci ini mempengaruhi jumlah putaran

enkripsi dan tingkat keamanannya (Azhari et al., 2022). Di sisi lain, ChaCha20, yang merupakan varian dari Salsa20, telah terbukti memiliki tingkat keamanan yang tinggi dan digunakan secara luas oleh Google dalam berbagai aplikasinya. Algoritma ini menggunakan keystream yang dihasilkan dari kombinasi kunci rahasia dan nonce, memastikan bahwa proses enkripsi dan dekripsi tetap aman dari potensi ancaman (Lima et al., 2022). Kombinasi kedua algoritma ini memungkinkan perlindungan data yang lebih kuat selama pertukaran informasi.

Sebagai solusi terhadap permasalahan keamanan file pada sistem penyimpanan berbagi menggunakan protokol SMB, penerapan sistem enkripsi yang menggabungkan AES Modifikasi AddRoundKey dan ChaCha20 memberikan perlindungan yang lebih tinggi. Dalam sistem ini, file yang diunggah ke Samba akan dienkripsi terlebih dahulu di server menggunakan AES untuk mengenkripsi file AES dan ChaCha20 untuk menghasilkan keystream yang aman. Setelah file terenkripsi, hanya pihak yang memiliki kunci yang sesuai yang dapat mengunduh dan membuka file tersebut. Dengan cara ini, file yang dipertukarkan melalui protokol SMB tetap terlindungi dari ancaman seperti peretasan, virus, atau malware. Solusi ini memberikan tingkat keamanan yang lebih baik untuk file pada proses unggah dan unduh melalui protokol SMBv1 menggunakan Samba. Maka dari itu file yang diunggah akan dienkripsi setelah proses upload selesai, dan file yang diunduh akan didekripsi sebelum diterima pengguna menggunakan kombinasi algoritma ChaCha20 dan AES Modifikasi. Dengan cara ini, keamanan data selama transfer dapat lebih terjamin, memberikan perlindungan tambahan terhadap risiko penyadapan atau manipulasi data serta dapat menyimpan file di file sharing dengan aman bagi pengguna dalam berbagi data di platform Linux.

II. KAJIAN TEORI

A. Protokol SMBv1

SMB(Server Message Block) adalah protokol jaringan yang digunakan untuk berbagi file, printer, dan komunikasi lainnya antara komputer di dalam jaringan. Versi pertama dari protokol ini, yang dikenal sebagai SMBv1, telah digunakan secara luas pada sistem operasi Windows yang lebih lama. Berikut merupakan karakteristik dari SMBv1:

a. Pendekatan Monolitik: SMBv1 dirancang sebagai protokol yang monolitik, yang berarti bahwa setiap operasi file dilakukan dalam beberapa langkah yang berurutan. Ini mengakibatkan peningkatan jumlah lalu lintas jaringan, membuat protokol ini dikenal sebagai “chattier” dibandingkan dengan versi yang lebih baru.
b. Transaksi File Terstruktur: Protokol ini menggunakan paket-paket yang sangat terstruktur untuk melakukan operasi seperti membuka, membaca, menulis, dan menutup file. Meskipun strukturnya membantu interoperabilitas, kompleksitas ini justru memperlambat proses di jaringan modern.
c. Kurangnya Enkripsi: SMBv1 tidak mendukung enkripsi end-to-end, sehingga data yang dikirimkan rentan terhadap penyadapan dan serangan man-in-the-middle.

SMBv1 mempunyai kelemahan yang signifikan, terutama terkait dengan keamanan dan efisiensi jaringan. Protokol ini tidak menyediakan mekanisme enkripsi bawaan, sehingga data yang dikirimkan melalui SMBv1 dapat dengan mudah disadap atau dimodifikasi. Selain itu, metode autentikasi yang lebih lemah membuat SMBv1 menjadi target yang rentan bagi peretas. Salah satu contoh mencolok dari kerentanannya adalah eksploitasi EternalBlue, yang digunakan dalam serangan ransomware besar seperti WannaCry. Selain itu, SMBv1 tidak dirancang untuk menangani volume data yang besar dan kebutuhan transfer yang lebih cepat di jaringan modern. Banyak operasi di SMBv1 membutuhkan permintaan dan respons berganda, yang mengakibatkan peningkatan latensi dan performa yang kurang optimal, terutama dalam lingkungan jaringan yang sibuk (Vinodhini et al., 2021).

B. Samba

Samba merupakan perangkat lunak dari Unix dan Linux, yang dikenal sebagai protokol Server Message Block (SMB). Banyak sistem operasi seperti Windows dan Linux menggunakan Samba untuk membuat jaringan klien-server. Protokol SMB memungkinkan server Unix dan Linux berkomunikasi dengan protokol Windows dalam suatu jaringan, dengan sistem operasi Linux Samba diperlakukan sebagai pengontrol domain utama seperti yang dilakukan NT dalam jaringan Microsoft Windows. Dengan menggunakan teknologi klien Samba pada Windows, server Linux dapat berkomunikasi satu sama lain, misalnya dengan berbagi file satu atau lebih (Ardiansyah et al., 2022).

Samba memiliki berbagai fungsi, mulai dari berbagi file, berbagi perangkat, Pengontrol Domain Utama (PDC), firewall, DNS, DHCP, FTP, server web, gateway, server email, proxy, dan lain-lain. Fitur jarak jauh seperti telnet dan SSH juga tersedia. Salah satu kelebihan Samba adalah aplikasi instalasinya yang tidak hanya berbasis teks, tetapi juga berbasis grafis, terutama melalui SWAT. Samba dapat mengatur mesin Linux/UNIX sebagai PDC, seperti yang dilakukan oleh NT pada jaringan Windows.

Samba mencakup dua program yang berjalan di background, yaitu SMBD (Server Message Block Daemon) dan NMBD (NetBIOS Name Block Daemon). Secara singkat, SMBD adalah program yang akan membuat proses baru untuk setiap klien yang aktif, sementara NMBD bertanggung jawab untuk memetakan nama komputer (NetBIOS) ke alamat IP (Internet Protocol) dan memantau berbagi file di jaringan. SMBD dikelola melalui file konfigurasi, yang memungkinkan Samba digunakan sebagai server file, server cetak, pengontrol domain, dan berbagai fungsi lainnya. Salah satu keuntungan utama menggunakan Samba sebagai pengontrol domain adalah biayanya yang jauh lebih murah dibandingkan dengan Windows NT/200x. Harga lisensi Windows NT/200x dan biaya akses per pelanggan sangat mahal, sementara dengan Linux dan Samba, semuanya dapat diperoleh hampir secara gratis. Meskipun demikian, Samba bukanlah perangkat lunak yang murahan. Sebuah penelitian menunjukkan bahwa kinerja server Samba dua setengah kali lebih cepat dibandingkan dengan server Windows NT 2003.

C. WireShark

Wireshark adalah alat analisis jaringan yang digunakan untuk menangkap dan memeriksa data yang berjalan di dalam jaringan komputer. Aplikasi ini mendukung berbagai protokol komunikasi dan memberikan informasi rinci tentang paket-paket data, memungkinkan penggunanya untuk memahami perilaku jaringan, mendeteksi anomali, serta menganalisis kinerja.

Wireshark sering digunakan dalam berbagai penelitian dan proyek pengembangan jaringan. Aplikasi ini berfungsi sebagai platform yang mendukung pembelajaran dan eksperimen dalam bidang telekomunikasi, khususnya untuk memvisualisasikan bagaimana data diproses dan diteruskan melalui jaringan. Fitur utamanya termasuk kemampuannya untuk menampilkan data paket dalam format yang mudah dipahami, mendukung filterisasi untuk memfokuskan analisis pada jenis data tertentu, serta menyediakan alat decoding untuk membantu memahami detail protokol yang kompleks.

Wireshark juga mendukung berbagai protokol jaringan, memungkinkan penggunanya untuk melihat struktur data mulai dari lapisan fisik hingga lapisan aplikasi. Dengan antarmuka yang intuitif, alat ini digunakan secara luas oleh profesional TI, insinyur jaringan, serta peneliti keamanan siber untuk memecahkan masalah jaringan, menguji konfigurasi keamanan, dan mengidentifikasi ancaman. Artikel ini menunjukkan bahwa Wireshark bukan hanya alat pemantauan biasa, melainkan juga menjadi media pembelajaran yang sangat bermanfaat dalam memahami dinamika jaringan modern(Jain & Anubha, 2021).

D. Kriptografi

Kriptografi merupakan teknik yang bertujuan untuk menjaga privasi dan keamanan suatu pesan hingga mencapai penerimanya, sekaligus mencegah kemungkinan penyadapan. Terdapat tiga konsep utama dalam kriptografi, yaitu enkripsi, dekripsi, dan kunci. Fungsi utama dari kriptografi adalah melindungi kerahasiaan kunci dan mengubah teks asli (plaintext) menjadi teks terenkripsi (ciphertext). Dengan demikian, teks asli berubah menjadi bentuk yang tidak dapat dimengerti tanpa kunci tertentu, tanpa perlu merahasiakan algoritma yang digunakan. Namun, jika kunci yang digunakan untuk menjaga privasi dan keamanan dapat diakses oleh pihak lain, maka keamanan pesan menjadi terancam. Oleh karena itu, kriptografi dianggap aman untuk diterapkan pada berbagai jenis media, seperti pesan teks, gambar, audio, video, maupun media lainnya (Sari et al., 2022).

Salah satu konsep utama dalam kriptografi adalah enkripsi, yaitu proses mengubah data yang dapat dipahami menjadi kode yang tidak dapat dimengerti. Tujuan utama dari enkripsi adalah untuk menjaga keamanan data. Metode ini digunakan sebagai upaya perlindungan terhadap kejahatan siber, seperti peretasan email, phishing, pencurian data, dan penyalahgunaan informasi kartu. Dalam prosesnya, data sensitif diacak sedemikian rupa sehingga berbeda dari bentuk aslinya. Dengan demikian, meskipun data tersebut berhasil diakses oleh peretas, data tersebut tidak dapat digunakan dengan mudah. Oleh karena itu, banyak platform digital seperti situs web dan media sosial saat ini mengadopsi enkripsi untuk melindungi kerahasiaan data pengguna. Data yang telah dienkripsi tetap dapat dikembalikan ke bentuk aslinya melalui proses dekripsi, yang hanya dapat dilakukan oleh pihak yang memiliki akses atau merupakan pemilik sah dari data tersebut (Wulandari & Hwihanus, 2023). Dekripsi, yang merupakan proses kebalikan dari enkripsi, bertujuan mengembalikan pesan terenkripsi ke bentuk aslinya (Azhari et al., 2022).

E. Chacha20

Menurut Bernstein, ChaCha20 adalah algoritma stream cipher yang dirancang oleh Daniel J. Bernstein sebagai varian dari algoritma Salsa20. Algoritma ini dirancang untuk meningkatkan keamanan, kinerja, dan kompatibilitas perangkat keras dibandingkan algoritma enkripsi lainnya. Algoritma ChaCha20 telah terbukti fleksibel dan efektif dalam berbagai pengaturan keamanan, termasuk sistem tertanam, jaringan sensor nirkabel, dan keamanan untuk mesin virtual di pusat data. Algoritma ini juga digunakan dalam protokol SSL/TLS oleh perusahaan seperti Google dan Cloudflare. ChaCha20 meningkatkan difusi data di setiap putaran sambil tetap mempertahankan performa yang tinggi. Menggunakan operasi Add-Rotate-XOR, ChaCha20 bekerja dengan matriks awal 4x4 yang terdiri dari 16 kata 32-bit, total berukuran 512-bit, untuk menghasilkan aliran kunci (keystream). Input utama algoritma ini meliputi kunci 256-bit, konstanta 128-bit, nonce 96-bit, dan counter 32-bit. ChaCha20 mendukung kunci dengan panjang 128-bit atau 256-bit, menggunakan konstanta "expand 32-byte k" dan nonce sebagai bilangan acak yang digunakan sekali saja. Dalam ChaCha20, kata-kata disimpan dalam format little-endian, dan susunan matriksnya terstruktur dengan rapi(Fitra Rahim & Supiyandi, 2024. ChaCha20 yang beroperasi pada matriks 4×4 yang terdiri dari 16 kata 32-bit (512 bit total), mencakup:

- 4 kata konstanta (128 bit): Biasanya string ASCII "expand 32-byte k".
- 8 kata kunci (256 bit): Kunci utama untuk proses enkripsi.
- 1 kata penghitung (32 bit): Mengindikasikan nomor blok, memastikan keystream unik.
- 3 kata nonce (96 bit): Sebuah nilai unik untuk setiap enkripsi.

Adapun susunan matriks-nya disusun seperti pada TABEL 1:

TABEL 1
(Matriks Enkripsi ChaCha20)

Const	Const	Const	Const
Key	Key	Key	Key
Key	Key	Key	Key
Counter	Nonce	Nonce	Nonce

Setelah 20 putaran selesai, matriks yang dihasilkan ditambahkan elemen per elemen dengan status matriks awal menggunakan penjumlahan modulo 2^{32}.

Proses Enkripsi:

- Matriks inisialisasi diatur dengan kunci, nonce, penghitung, dan konstanta.
- Matriks diproses melalui 20 putaran quarter-round.
- Hasil akhir matriks digunakan untuk menghasilkan keystream.
- Keystream dioperasikan dengan plaintext menggunakan XOR untuk menghasilkan.

Keystream yang dihasilkan dari proses matriks internal digunakan untu mengenkripsi *plaintext* dengan operasi XOR:

$$cipher\ text\ [i] = Plaintext[i] \oplus Keystream[i] \quad (1)$$

Setiap blok data menggunakan nonce dan penghitung yang unik untuk memastikan tidak ada blok data yang berbagi keystream.

Fungsi Quarter-Round adalah inti dari ChaCha20 yang memproses empat elemen dalam matriks internal. Quarter-Round bekerja dengan menggunakan operasi ARX untuk menghasilkan difusi yang kuat. Fungsi ini dijelaskan melalui rumus:

$$a \mathrel{+}= b;\ d \mathrel{\hat{}}= a;\ d = ROTL(d,16);$$

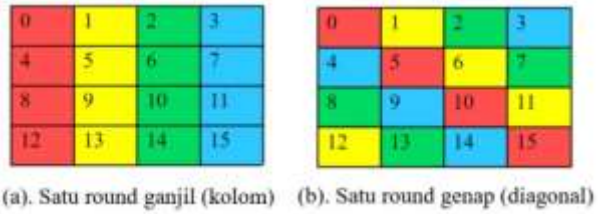
$$c \mathrel{+}= d;\ b \mathrel{\hat{}}= c;\ b = ROTL(b,12); \quad (2)$$

$$a \mathrel{+}= b;\ d \mathrel{\hat{}}= a;\ d = ROTL(d,8);$$

$$c \mathrel{+}= d;\ b \mathrel{\hat{}}= c;\ b = ROTL(b,7);$$

Keterangan:
- += adalah penjumlahan modulo 2^{32} untuk menambahkan data secara siklis.
- ^= adalah operasi XOR.
- ROTL(x, n) adalah rotasi bit ke kiri sebanyak n posisi untuk mempercepat distribusi bit data.

Dalam penelitian ini, saya menggunakan 20 putaran yang terdiri dari 10 double round. Pada GAMBAR 1 setiap double round meliputi 2 putaran: putaran ganjil yang terdiri dari 4 quarter round kolom dan putaran genap dengan 4 quarter round diagonal. ChaCha20 melakukan 10 putaran ganda, setara dengan 20 putaran atau 80 aplikasi quarter round.

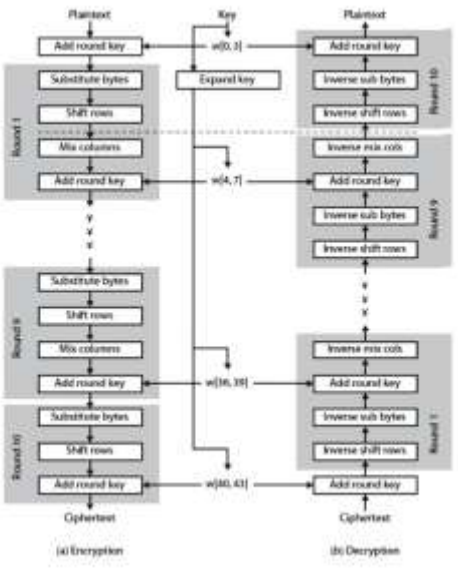


GAMBAR 1
(Double Round (Fitra Rahim & Supiyandi, 2024))

F. Advanced Encryption Standard (AES)

Advanced Encryption Standard (AES) adalah algoritma enkripsi blok yang dirilis oleh National Institute of Standards and Technology (NIST) pada tahun 2000. Algoritma ini dikembangkan untuk menggantikan DES yang mulai dianggap rentan terhadap berbagai jenis serangan. Untuk menemukan pengganti DES, NIST mengundang pakar keamanan data dan enkripsi dari seluruh dunia untuk mengusulkan algoritma cipher blok yang kuat dan inovatif untuk mengenkripsi dan mendekripsi data.

Dari seluruh algoritma yang diajukan, NIST menerima lima algoritma untuk dievaluasi secara menyeluruh. Setelah melalui berbagai kriteria dan parameter keamanan, algoritma yang diusulkan oleh kriptografer Belgia Joan Daemen dan Vincent Rijmen terpilih sebagai pemenang. Algoritma ini awalnya dikenal dengan nama Rijndael sebelum resmi dinamai AES.



GAMBAR 2
(Arsitektur AES(Maulana Anidita A.A, 2023))

Algoritma AES menggunakan beberapa proses iteratif untuk menghasilkan *ciphertext*, dengan tingkat kekuatan enkripsi yang berbeda:
1. 10 iterasi untuk kekuatan enkripsi 128 bit.
2. 12 iterasi untuk kekuatan enkripsi 192 bit.
3. 14 iterasi untuk kekuatan enkripsi 256 bit.

Proses iteratif ini memastikan AES memiliki struktur yang kompleks seperti pada GAMBAR 1 dan tahan terhadap berbagai jenis serangan, menjadikannya salah satu algoritma enkripsi paling andal hingga saat ini (Ibtihaji Ilham et al., 2021.

Dalam sebuah perulangan proses enkripsi setiap blok, terdapat empat sub-proses yang berlaku:
1. *SubBytes* dari data masukan disubstitusikan menggunakan byte yang sesuai berdasarkan substitution box (S-box) yang bisa ditunjukkan pada GAMBAR 2 Tahap ini membentuk nonlinearitas dan kebingungan (confusion) dalam data hasil enkripsi.

	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	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

GAMBAR 3
(S-box Algoritma AES(Maulana Anidita A.A, 2023))

2. *ShiftRows*: Setiap baris data digeser secara siklis. Baris pertama tetap pada posisinya, baris kedua digeser ke kiri satu posisi, baris ketiga digeser ke kiri dua posisi, dan baris keempat digeser ke kiri tiga posisi. Tahap ini menciptakan diffusion dan menyebarkan data di setiap baris.
3. *MixColumns*: Setiap kolom data diacak menggunakan matriks pengacakan. Tahap ini menambah diffusion

yang lebih dalam dan memastikan bahwa setiap byte dalam kolom tergantung pada empat byte lainnya.

4. *AddRoundKey*: Pada tahap ini, dilakukan operasi XOR antara data dan round key. Round key diperoleh dari kunci utama melalui algoritma penjadwalan kunci (Key-Scheduling Algorithm / KSA). Tahap ini memberikan kunci yang unik pada data, sehingga hasil enkripsi bergantung pada kunci tersebut.

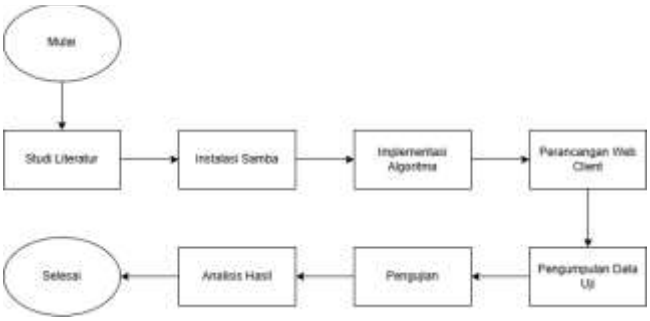
Keempat proses ini akan dijalankan berulang pada setiap iterasi, kecuali pada iterasi terakhir, di mana proses MixColumns akan dilewati. Hasil dari iterasi terakhir ini adalah data yang telah terenkripsi(Maulana Anidita A.A, 2023).

III. METODE

Penelitian ini menggunakan pendekatan kuantitatif untuk menganalisis penerapan keamanan file pada shared data storage berbasis server Samba dengan kombinasi algoritma enkripsi ChaCha20 dan modifikasi AES melalui AddRoundKey. Penelitian dimulai dengan instalasi server Samba yang dikonfigurasi untuk mendukung protokol SMB sebagai media penyimpanan berbagi. Kemudian, algoritma ChaCha20 dan modifikasi AES diimplementasikan untuk melakukan proses enkripsi dan dekripsi file. Selanjutnya, dirancang antarmuka web client yang memungkinkan pengguna untuk mengupload file, mendownload file, dan melihat daftar file yang tersedia di server. Data uji berupa file teks dan gambar disiapkan, dienkripsi menggunakan modifikasi AES untuk menghasilkan ciphertext awal, kemudian diproses lebih lanjut dengan ChaCha20 untuk meningkatkan keamanan, sebelum dikirim ke server Samba untuk disimpan secara terenkripsi. Saat file diunduh, data diterima dalam bentuk terenkripsi dan didekripsi kembali ke bentuk aslinya menggunakan kombinasi algoritma yang sama. Pengujian dilakukan untuk mengukur kecepatan transfer file (upload dan download), memverifikasi integritas data melalui perbandingan checksum sebelum dan sesudah transfer, serta memastikan keamanan data selama proses transfer menggunakan Wireshark. Hasil pengujian dianalisis untuk mengevaluasi efektivitas kombinasi algoritma ChaCha20 dan modifikasi AES dalam meningkatkan keamanan data serta kinerja sistem secara keseluruhan.

A. Prosedur Penelitian

Prosedur penelitian ini mencakup setiap tahap, mulai dari instalasi perangkat lunak hingga analisis hasil. Prosedur dimulai dengan tahap Instalasi Samba, yang mungkin merujuk pada persiapan lingkungan jaringan atau server. Selanjutnya, Implementasi Algoritma menunjukkan pengembangan inti dari sistem, di mana algoritma spesifik dirancang dan dikembangkan. Setelah itu, Perancangan Web Client mencakup pengembangan antarmuka pengguna yang akan berinteraksi dengan sistem. Pengumpulan Data Uji dan Pengujian menunjukkan fase pengujian di mana data diuji dan sistem diuji untuk validitas dan efisiensinya. Proses berakhir dengan Analisis Hasil, dimana hasil pengujian dianalisis untuk mendapatkan wawasan dan kesimpulan. Flowchart alur penelitian bisa dilihat pada GAMBAR 4.



GAMBAR 4
(Flowchart Rancangan Penelitian)

IV. HASIL DAN PEMBAHASAN

Penelitian ini menghasilkan sebuah sistem keamanan file berbasis web yang menggunakan kombinasi algoritma enkripsi AES Modifikasi AddRoundKey dan ChaCha20, serta diintegrasikan dengan layanan berbagi file melalui protokol SMBv1 (Server Message Block) menggunakan Samba. Sistem ini mampu melakukan proses unggah dan unduh file yang terenkripsi secara otomatis dengan mempertahankan integritas serta meningkatkan keamanan terhadap penyadapan.

A. Hasil Implementasi Sistem

Sistem yang diimplementasikan memiliki dua antarmuka utama, yaitu Samba Standard (tanpa enkripsi) dan Samba Modified (dengan enkripsi). Keduanya diakses melalui web interface berbasis framework Flask yang dirancang untuk memudahkan pengguna dalam mengunggah dan mengunduh file, baik secara standar maupun dengan lapisan keamanan tambahan.

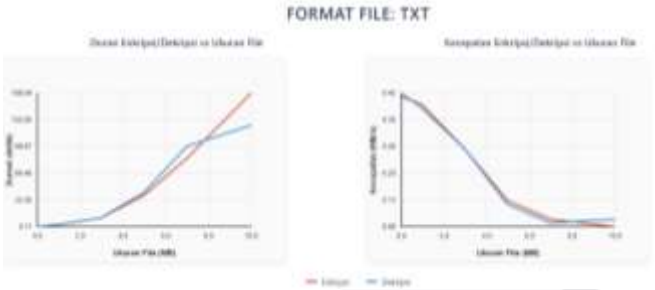
GAMBAR 5
(Samba Modified)

Antarmuka Samba Modified - Encrypted menyediakan fitur unggah dan unduh file dengan perlindungan enkripsi ganda menggunakan kombinasi algoritma ChaCha20 dan Modified AES. Pengguna dapat memilih file, memasukkan kunci enkripsi, lalu mengeksekusi proses unggah terenkripsi. Untuk mengakses kembali file yang telah terenkripsi, pengguna cukup memasukkan nama file serta kunci dekripsi yang sesuai. Implementasi ini dirancang untuk meningkatkan keamanan data yang dibagikan melalui protokol SMBv1.

GAMBAR 6
(Samba Standard)

Sementara itu, antarmuka Samba Standard - No Encryption menyediakan layanan unggah dan unduh file tanpa mekanisme enkripsi. Fitur ini digunakan untuk membandingkan performa sistem berbagi file standar tanpa perlindungan kriptografi, seperti kecepatan transfer yang tinggi namun tanpa jaminan kerahasiaan data.
Kedua antarmuka dilengkapi tombol navigasi yang memungkinkan pengguna beralih antar mode (Standard ke Modified) dengan mudah, menjadikan sistem fleksibel baik dari sisi efisiensi maupun keamanan.

B. Tabel Hasil Waktu Enkripsi dan Dekripsi



GAMBAR 7
(Analisis Performa Enkripsi/ Dekripsi Format TXT)

Pengujian performa terhadap file .txt pada GAMBAR 7 menunjukkan bahwa durasi enkripsi dan dekripsi meningkat seiring bertambahnya ukuran file. Untuk file kecil (±49 KB), proses berlangsung sangat cepat (±0,114 detik), namun durasi meningkat signifikan pada file besar, mencapai 166 detik untuk enkripsi dan 125 detik untuk dekripsi pada file 10 MB. Menariknya, dekripsi cenderung sedikit lebih cepat dibandingkan enkripsi mulai dari ukuran 3 MB ke atas.
Grafik kecepatan menunjukkan tren penurunan tajam, dari ±0,42 MB/s pada file kecil menjadi ±0,06 MB/s pada file besar. Penurunan ini menunjukkan beban komputasi yang meningkat secara non-linear terhadap ukuran file. Temuan ini menegaskan bahwa sistem bekerja sangat efisien pada file kecil, namun perlu optimasi lebih lanjut agar tetap responsif saat menangani file berukuran besar.

TABEL 2
(Uji Coba Parameter TXT)

Jenis *File*	Size File (MB)	Waktu Upload (s)	Waktu Download (s)	Kecepatan Upload (MB/s)	Kecepatan Download (MB/s)
Txt	1	2.560	2.611	0.39	0.38
	3	10.922	11.015	0.27	0.27
	5	38.941	42.526	0.13	0.12
	7	84.706	99.941	0.08	0.07
	10	166.040	125.964	0.06	0.08

Berdasarkan hasil pengujian terhadap file TXT pada tabel 2 dengan ukuran 1 MB hingga 10 MB, dapat disimpulkan bahwa sistem enkripsi bekerja stabil dan berhasil memproses seluruh file tanpa error. Namun, performa sistem menunjukkan bahwa waktu proses meningkat signifikan seiring bertambahnya ukuran file, sementara kecepatan transfer menurun secara bertahap. Ini menandakan bahwa beban komputasi sistem meningkat secara non-linear pada file berukuran besar. Oleh karena itu, meskipun sistem terbukti andal, diperlukan optimasi performa agar efisiensi tetap terjaga pada file berukuran besar.

C. Analisis Checksum untuk Integritas Data

Pengujian keamanan file diukur menggunakan web MD5 File Checksum dan hasil ditunjukkan pada table IV.15 untuk memastikan integritas data selama proses enkripsi, upload, download, dan dekripsi.

TABEL 3
(Pengujian Keamanan File)

Tahap	Checksum	Keterangan
File Asli	b0260a8bea3ab5bfa32ce27be4c35ccd	Validasi integritas awal.
Setelah *Upload*	557c1d73d83860593b53f69ad4124a29	Hasil enkripsi harus unik.
Setelah Enkripsi	557c1d73d83860593b53f69ad4124a29	Validasi integritas *file* di *server*.
Setelah *Download*	b0260a8bea3ab5bfa32ce27be4c35ccd	Verifikasi tidak ada perubahan selama proses.
Setelah Dekripsi	b0260a8bea3ab5bfa32ce27be4c35ccd	*File* kembali ke bentuk asli tanpa perubahan.
Setelah Dekripsi	b0260a8bea3ab5bfa32ce27be4c35ccd	*File* kembali ke bentuk asli tanpa perubahan.

Pengujian integritas data yang dilakukan pada TABEL 3 menggunakan metode checksum MD5 pada lima tahap proses, yaitu sebelum dan sesudah enkripsi, unggah, unduh, dan dekripsi. Hasil menunjukkan bahwa file asli memiliki nilai checksum b0260a8bea3ab5bfa32ce27be4c35ccd, yang berubah menjadi 557c1d73d83860593b53f69ad4124a29 setelah proses enkripsi dan tetap konsisten selama penyimpanan di server. Setelah file diunduh dan didekripsi, nilai checksum kembali ke bentuk awal, menandakan bahwa tidak ada perubahan pada isi file sepanjang proses berlangsung. Temuan ini membuktikan bahwa sistem mampu menjaga integritas file dari awal hingga akhir secara

menyeluruh, dan mekanisme enkripsi-dekripsi berjalan sempurna tanpa menyebabkan kerusakan data.

D. Analisis Kinerja Sistem

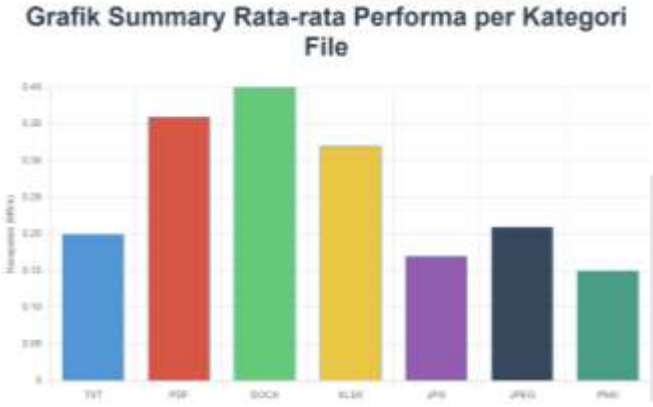
Pada bagian ini, dilakukan analisis terhadap kinerja sistem dengan menggunakan dua metode, yaitu Samba Standar dan Samba Modifikasi. Hasil pengujian ini bertujuan untuk membandingkan efisiensi dan keamanan antara metode Samba Standar dan Samba Modifikasi, serta mengevaluasi kelebihan yang ditawarkan oleh metode modifikasi dalam meningkatkan performa sistem.

TABEL 4
(Analisis Hasil Pengujian)

Metode	EU	ED	ES	WU (s)	WD (s)	KU (MB/s)	KD (MB/s)
Samba Standart	-	-	-	0.0209	0	85.01	2046.48
Samba Modifikasi	Berhasil	Berhasil	Berhasil	5.1384	5.3723	0.34	0.33

Analisis performa pada TABEL 4 menunjukkan bahwa Samba standar tanpa enkripsi memiliki waktu unggah dan unduh sangat cepat (WU: 0,0209 detik; WD: 0 detik) dengan kecepatan transfer tinggi (upload: 85,01 MB/s; download: 2046,48 MB/s). Sebaliknya, sistem dengan enkripsi ChaCha20 dan Modified AES mengalami penurunan performa (WU: 5,1384 detik; WD: 5,3723 detik; kecepatan transfer ±0,34 MB/s). Meskipun demikian, penambahan lapisan enkripsi terbukti memberikan perlindungan data yang jauh lebih tinggi. Trade-off antara keamanan dan efisiensi ini dinilai wajar, terutama untuk lingkungan yang memprioritaskan kerahasiaan data.

E. Analisis Performa per Kategori File



GAMBAR 8
(Grafik Rata-Rata Performa per Kategori File)

Analisis performa berdasarkan jenis file pada GAMBAR 8 menunjukkan bahwa file dokumen cenderung memiliki kecepatan proses yang lebih tinggi dibandingkan file gambar. File DOCX mencatat kecepatan rata-rata tertinggi (0,40 MB/s), diikuti oleh PDF (0,36 MB/s), dan TXT (0,20 MB/s), menunjukkan efisiensi tinggi pada struktur file teks. Sementara itu, file XLSX mengalami penurunan performa (0,32 MB/s) akibat kompleksitas struktur datanya.

Pada kategori gambar, performa cenderung lebih rendah. File JPEG menunjukkan kecepatan paling stabil (0,21 MB/s), diikuti oleh JPG (0,17 MB/s) dan PNG (0,15 MB/s). Variasi ini menunjukkan bahwa sistem lebih optimal dalam menangani dokumen dibandingkan file visual, yang umumnya lebih kompleks dan berukuran besar.

TABEL 5
(Analisis Performa per Kategori File)

Kategori File	Kecepatan Upload Rata-rata(MB/s)	Kecepatan Download Rata-rata(MB/s)
Dokumen	0.32	0.34
Gambar	0.18	0.19
Keseluruhan	0.25	0.27

Berdasarkan TABEL 5, sistem menunjukkan rata-rata kecepatan upload sebesar 0,32 MB/s dan download 0,34 MB/s untuk file dokumen, mencerminkan efisiensi yang baik terutama pada format sederhana seperti TXT dan DOCX. Untuk file gambar, kecepatan rata-rata upload dan download masing-masing tercatat 0,18 MB/s dan 0,19 MB/s, menunjukkan efisiensi sedang akibat kompleksitas struktur data visual.

Secara keseluruhan, rata-rata kecepatan sistem adalah 0,25 MB/s (upload) dan 0,27 MB/s (download). Meskipun tergolong cukup efisien, peningkatan performa masih diperlukan, khususnya untuk file besar dan gambar. Implementasi teknik kompresi disarankan guna mengoptimalkan kecepatan pemrosesan tanpa mengurangi tingkat keamanan sistem.

F. Performa Samba Modifikasi vs Samba Standar

TABEL 6
(Analisis Dampak Performa Samba Modifikasi vs Samba Standar)

Metode	Kecepatan Upload(MB/s)	Kecepatan Download(MB/s)	Dampak Performa Upload	Dampak Performa Download
Samba Standar	85.01	2046.48	-	-
Samba Modifikasi	0.34	0.33	-99.6%	-99.98%

Pada TABEL 6, dampak performa antara Samba Modifikasi dan Samba Standar menunjukkan perbedaan yang mencolok.

- Kecepatan Upload pada Samba Standar mencapai 85.01 MB/s, sedangkan pada Samba Modifikasi, kecepatan upload turun drastis menjadi 0.34 MB/s, mencerminkan penurunan performa sebesar 99.6%.
- Kecepatan Download juga mengalami penurunan yang signifikan, dari 2046.48 MB/s pada Samba Standar menjadi 0.33 MB/s pada Samba Modifikasi, yang berarti penurunan performa sebesar 99.98%.

Penurunan performa ini menunjukkan bahwa meskipun sistem enkripsi memberikan perlindungan keamanan yang komprehensif, ada trade-off yang signifikan antara keamanan dan efisiensi.

V. KESIMPULAN

Penelitian ini berhasil mengimplementasikan sistem keamanan file pada shared data storage berbasis Linux dengan kombinasi algoritma ChaCha20 dan AES yang dimodifikasi pada tahap AddRoundKey. Sistem diterapkan melalui protokol SMBv1 menggunakan Samba dan diuji melalui antarmuka web client berbasis Flask. Proses enkripsi

berlapis terbukti mampu menjaga kerahasiaan dan integritas data, dibuktikan dengan nilai checksum identik antara file asli dan hasil dekripsi. Analisis lalu lintas jaringan menggunakan Wireshark menunjukkan bahwa file tetap terenkripsi selama transmisi, mencegah potensi penyadapan. Sistem juga menunjukkan performa efisien dengan rata-rata waktu enkripsi 0,02583 detik dan dekripsi 0,02271 detik untuk file 1MB, serta distribusi data terenkripsi yang acak berdasarkan histogram.

Untuk pengembangan lebih lanjut, sistem disarankan menggunakan protokol SMBv2/v3 guna meningkatkan keamanan. Penambahan fitur manajemen kunci berbasis PKI dan autentikasi multi-faktor juga direkomendasikan. Selain itu, pengujian perlu diperluas ke format file lain (audio, video, arsip), dan antarmuka web dapat ditingkatkan dengan fitur seperti enkripsi sisi klien, log aktivitas, dan sistem backup otomatis.

REFERENSI

Ardiansyah, M. F., Diansyah, T. M., Liza, R., & Redaksi, D. (2022). Penggunaan Set top box Bekas untuk Dimanfaatkan sebagai Cloud Server. *Blend Sains Jurnal Teknik*, *1*(2), 88–96. https://doi.org/10.56211/BLENDSAINS.V1I2.115

Azhari, M., Mulyana, D. I., Perwitosari, F. J., & Ali, F. (2022). Implementasi Pengamanan Data pada Dokumen Menggunakan Algoritma Kriptografi Advanced Encryption Standard (AES). *Jurnal Pendidikan Sains Dan Komputer*, *2*(01), 163–171. https://doi.org/10.47709/JPSK.V2I01.1390

Fitra Rahim, Y. R. N., & Supiyandi. (2024). View of Implementasi Algoritma ChaCha20 Pada Pengamanan File Citra Bitmap. *FASILKOM*, *14*, 615–626. https://www.ejurnal.umri.ac.id/index.php/JIK/article/view/7956/3266

Ibtihaji Ilham, L., Pramita Widyassari, A., Sekolah Tinggi Teknologi Ronggolawe Cepu, ab, & Teknik Elektro, J. (2021). Pengembangan Aplikasi Pesan Instan Terenkripsi Menggunakan Algoritma Kriptografi AES (Advanced Encryption Standard). *JES (Jurnal Elektro Smart)*, *1*(1), 1–6. https://www.jurnal.sttrcepu.ac.id/index.php/jes/article/view/157

Insanudin, E., Sularsa, A., & Soegiarto, D. (2024). Design Of Web-Based Cloud Drive Application As Online Storage Media Using Virtual Private Server. *Jurnal Ilmiah Teknologi Infomasi Terapan*, *11*(1). https://doi.org/10.33197/JITTER.VOL11.ISS1.2024.2389

Jain, G., & Anubha. (2021). Application of SNORT and Wireshark in Network Traffic Analysis. *IOP Conference Series: Materials Science and Engineering*, *1119*(1), 012007. https://doi.org/10.1088/1757-899X/1119/1/012007

Lima, P. M., da Silva, C. K. P., de Farias, C. M., Carvalho, L. K., & Moreira, M. V. (2022). Event-based cryptography for automation networks of cyber-physical systems using the stream cipher ChaCha20. *IFAC-PapersOnLine*, *55*(28), 58–65. https://doi.org/10.1016/J.IFACOL.2022.10.324

Maulana Anidita A.A. (2023). *Analisis Perbandingan Algoritma AES dan ChaCha20-Poly1305 dalam Enkripsi Konten Livestreaming*. https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi-dan-Koding/2022-2023/Makalah2023/Makalah-KriptoKoding-2023%20(10).pdf

Rezaldy, A., & Ropianto, M. (2023). Mengkonfigurasi Samba Server Di Sistem Operasi Linux. *Jurnal Ilmiah Sistem Informasi*. https://d1wqtxts1xzle7.cloudfront.net/78655758/Aditya_Rezaldy_Mengkonfigurasi_Samba_Server_di_OS_Linux-libre.pdf?1642149947=&response-content-disposition=inline%3B+filename%3DMENGKONFIGURASI_SAMBA_SERVER_DI_SISTEM_O.pdf&Expires=1736608135&Signature=DcAGbYpEhMYPR21B087i2OhmrWWQQ-2AXJjTHWK-dwclYdqSr1VWTRYQ07dQjzMGzxXMD98iUrdk2ZQMhw-GvZwlh4GQAJEwJNIEtMjzepx3fCsijdfV-aQItASsULCekMQdcllKvOx~PuHab2XJrqJWFPQCQMOo~baKpw-~B9OQ9oHrWkUGP82DDmHySIFe3xrbEdYLqeWXKZfi6nDzwXGbsWk9IzRczTBm1oYfslB162E8Rbe~rK8JGYy

Sari, M., Kriptografi Keamanan, A., Dwi Purnomo, H., Sembiring, I., Sistem Informasi, M., Teknologi Informasi, F., Kristen Satya Wacana, U., & Notohamidjojo, J. O. (2022). Review : Cryptographic Algorithm for SMS Security System on Android. *Journal of Information Technology*, *2*(1), 11–15. https://doi.org/10.46229/JIFOTECH.V2I1.292

Vinodhini, V., Kumuthini, C., & Santhi, K. (2021). *A Study on Behavioural Analysis of Specific Ransomware and its Comparison with DBSCAN-MP*. https://doi.org/10.32628/CSEIT206670

Wijaya, E., Purwantoro ESGS, S., Caltex Riau, P., & Umban Sari No, J. (2022). Perbandingan Kinerja Clustered File System pada Cloud Storage menggunakan GlusterFS dan Ceph. *INOVTEK Polbeng - Seri Informatika*, *7*(2), 319–333. http://103.174.114.133/index.php/ISI/article/view/2753

Wulandari, I. W., & Hwihanus, H. (2023). Peran Sistem Informasi Akuntansi Dalam Pengaplikasian Enkripsi Terhadap Peningkatan Keamanan Perusahaan. *Jurnal Kajian Dan Penalaran Ilmu Manajemen*, *1*(1), 11–25. https://doi.org/10.59031/JKPIM.V1I1.46