Jurnal Elektronik Ilmu Komputer Udayana
Volume 13, No 3. February 2025

p-ISSN: 2301-5373
e-ISSN: 2654-5101

Perbandingan Algoritma Blowfish dan Gost Berbasis CriptoForge untuk Enkripsi Deskripsi File Teks

Ilham Bisma Akbar Maolana[1], Alam Rahmatulloh[2]

Program Studi Informatika, Fakultas Teknik, Universitas Siliwangi
Jalan Siliwangi No.24, Tasikmalaya, Jawa Barat, Indonesia 46115
[1]ilhambisma09@gmail.com
[2]alam@unsil.ac.id

Abstrak

Kemajuan teknologi mengakibatkan meningkatnya jumlah data yang perlu diolah dan disimpan. Namun, tidak semua data dapat diakses secara publik karena adanya informasi sensitif yang harus dirahasiakan. Kriptografi hadir sebagai solusi untuk mengamankan informasi, sehingga pemilihan algoritma yang efisien menjadi hal yang krusial. Penelitian ini bertujuan untuk menguji dan membandingkan efisiensi algoritma Blowfish dan GOST dalam proses enkripsi dan dekripsi data teks (.txt). Blowfish menggunakan panjang kunci 448 bit, sedangkan GOST menggunakan panjang kunci 256 bit, dengan keduanya diimplementasikan menggunakan perangkat lunak CriptoForge. Hasil simulasi menunjukkan bahwa meskipun panjang kunci berbeda, ciphertext yang dihasilkan memiliki ukuran yang sama karena kedua algoritma mengenkripsi data dengan membaginya menjadi blok berukuran tetap, yaitu 64 bit. Dalam hal kecepatan enkripsi, Blowfish lebih unggul dengan kecepatan sekitar 124–125 MB/detik dibandingkan dengan GOST yang hanya mencapai 59–63 MB/detik. Perbedaan ini disebabkan oleh desain Blowfish yang lebih efisien, sementara GOST memiliki proses enkripsi yang lebih kompleks sehingga mempengaruhi kecepatan pemrosesan. Dengan demikian, Blowfish lebih direkomendasikan untuk aplikasi yang membutuhkan kecepatan tinggi, sedangkan GOST dapat digunakan untuk kebutuhan yang mengutamakan aspek keamanan lebih lanjut.*

Kata kunci: *Informasi, Blowfish, Gost, Enkripsi, Deskripsi*

1. Pendahuluan

Keamanan data teks merupakan aspek krusial dalam teknologi informasi modern, terutama dalam menjaga integritas dan keaslian data dari akses yang tidak sah [1]. Seiring dengan meningkatnya jumlah data yang dikirim dan disimpan secara digital, risiko kebocoran dan penyalahgunaan data juga semakin tinggi [2]. Kebocoran data dapat menyebabkan kerugian finansial yang signifikan bagi perusahaan serta merusak reputasi dan kepercayaan pelanggan [3]. Oleh karena itu, diperlukan metode keamanan yang efektif untuk melindungi data dari ancaman tersebut.

Kriptografi merupakan salah satu solusi utama dalam menjaga kerahasiaan dan integritas data [4]. Teknik ini memastikan bahwa data yang dikirimkan diamankan agar hanya dapat diakses oleh penerima yang berwenang [5]. Berbagai algoritma kriptografi telah dikembangkan untuk memenuhi kebutuhan keamanan informasi, termasuk kriptografi simetris dan asimetris.

Kriptografi simetris, seperti algoritma Blowfish dan GOST, menggunakan satu kunci yang sama untuk proses enkripsi dan dekripsi. Keunggulan utama dari algoritma ini adalah kecepatan dan efisiensinya dalam pengolahan data. Namun, kelemahan utama kriptografi simetris terletak pada risiko keamanan jika kunci enkripsi terekspos [6]. Sebaliknya, kriptografi asimetris, seperti algoritma RSA dan ECC, menggunakan pasangan kunci publik dan privat untuk meningkatkan keamanan. Kunci privat tidak perlu dibagikan, sehingga lebih sulit untuk diretas. Namun, kriptografi asimetris memiliki kecepatan pemrosesan yang lebih rendah dibandingkan dengan kriptografi simetris karena melibatkan perhitungan matematis yang kompleks [7].

Beberapa penelitian telah membandingkan berbagai algoritma kriptografi berdasarkan efisiensi dan tingkat keamanannya. Misalnya, kombinasi algoritma simetris Blowfish dengan algoritma asimetris RSA dapat memberikan keamanan lebih tinggi dengan tetap mempertahankan efisiensi proses enkripsi dan dekripsi. Perbandingan antara algoritma AES dan RC4 menunjukkan bahwa meskipun RC4 memiliki kecepatan enkripsi yang lebih baik, AES menghasilkan ciphertext dengan ukuran yang berbeda dari ukuran data asli, sedangkan ciphertext RC4 tetap sama. Hal ini menunjukkan bahwa pemilihan algoritma harus mempertimbangkan keseimbangan antara efisiensi dan keamanan [8].

Selain itu, beberapa algoritma klasik seperti One-Time Pad, Vigenère Cipher, dan Hill Cipher memiliki karakteristik unik dalam keamanan dan efisiensi. One-Time Pad memiliki tingkat keamanan yang tinggi karena menggunakan kunci unik yang tidak dapat dipecahkan jika tetap dirahasiakan. Namun, tantangan utama algoritma ini adalah kebutuhan akan kunci yang sama panjang dengan data serta pengelolaan distribusi kunci yang aman [9]. Algoritma Vigenère Cipher menawarkan fleksibilitas penggunaan kunci, tetapi rentan terhadap analisis frekuensi jika kunci terlalu pendek atau berulang [10]. Hill Cipher, yang berbasis operasi matriks, memiliki ketergantungan pada matriks kunci dengan determinan tidak nol, sehingga kurang efisien untuk pesan pendek [11].

Algoritma vernam menggunakan operasi XOR antara teks asli (plaintext) dan kunci, menghasilkan teks terenkripsi (ciphertext) yang hanya dapat didekripsi dengan kunci yang sama, ketergantungan pada kunci yang harus acak dan digunakan sekali (one-time pad). Jika kunci tidak acak atau digunakan berulang, enkripsi bisa ditembus [12]. Algoritma Hill Chiper metode kriptografi simetris yang mengandalkan operasi matriks dan modulo untuk mengenkripsi dan mendekripsi pesan ketergantungan pada matriks kunci dengan determinan tidak nol, kurang efisien untuk pesan pendek [13]. Algoritma aesar Cipher dengan integrasi karakter kontrol ASCII non-printable, yang meningkatkan keamanan data secara signifikan, ketergantungan pada karakter kontrol ASCII non-printable yang kurang umum dan tidak selalu kompatibel dengan berbagai sistem atau aplikasi [14].

Algoritma modern seperti RC5 dan Caesar Cipher juga banyak digunakan untuk mengamankan data. RC5 merupakan algoritma simetris berbasis blok yang cepat dan ringan, serta memiliki tingkat keamanan yang baik [15]. Sementara itu, Caesar Cipher dan metode transposisi sering digunakan dalam sistem sederhana untuk mengamankan data dari pencurian atau penyalahgunaan oleh pihak yang tidak berwenang [16]. Algoritma Base64 juga digunakan untuk pengamanan pesan teks pada perangkat seluler dengan metode encoding yang mengubah teks menjadi ciphertext berbasis ASCII [17].

CriptoForge merupakan perangkat lunak enkripsi yang digunakan untuk melindungi data sensitif melalui proses enkripsi. Perangkat lunak ini dirancang untuk memberikan perlindungan data yang mudah digunakan, baik untuk individu maupun organisasi [18].

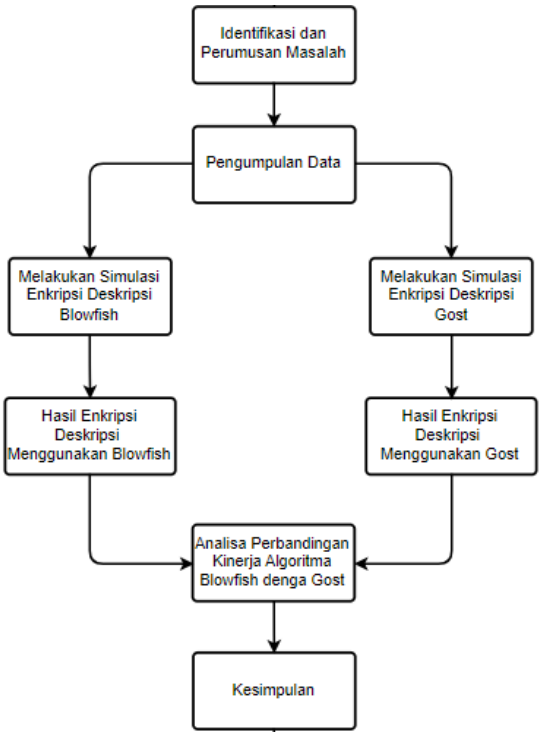
Dalam penelitian ini, algoritma Blowfish dan GOST dipilih untuk dibandingkan karena keduanya merupakan algoritma simetris yang dikenal cepat dan kuat. Perbandingan ini bertujuan untuk mengetahui algoritma mana yang lebih efisien dalam mengamankan data teks. Proses enkripsi dan dekripsi akan dilakukan menggunakan perangkat lunak CriptoForge untuk mengukur kecepatan dan efisiensi masing-masing algoritma. Hasil penelitian ini diharapkan dapat memberikan wawasan tentang pemilihan algoritma kriptografi yang optimal dalam menjaga keamanan data teks.

2. Metode

2.1. Tahapan Penelitian

Penelitian ini mengikuti langkah-langkah yang ditunjukkan dalam Gambar 1. Langkah pertama adalah mengidentifikasi dan merumuskan masalah untuk menentukan solusi yang tepat. Selanjutnya, dilakukan pengumpulan data yang diperlukan sebagai dasar analisis. Setelah data terkumpul, proses simulasi enkripsi dan dekripsi dilakukan menggunakan algoritma Blowfish dan GOST. Hasil simulasi kemudian dianalisis untuk membandingkan efisiensi kedua algoritma berdasarkan kecepatan dan ukuran ciphertext yang dihasilkan. Penelitian ini diakhiri dengan penyusunan kesimpulan berdasarkan hasil analisis yang telah dilakukan.

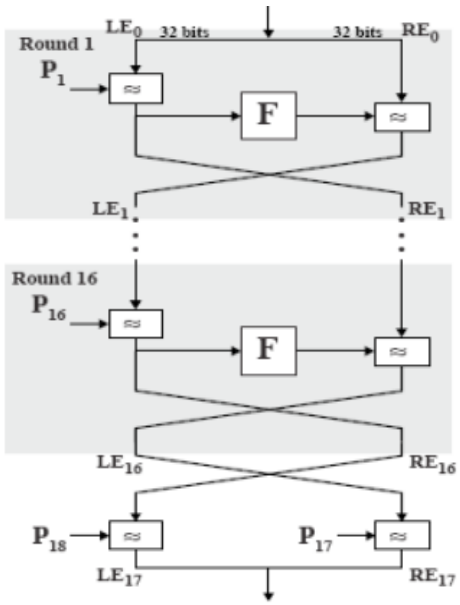
p-ISSN: 2301-5373
e-ISSN: 2654-5101



Gambar 1. Tahapan Penelitian

2.2. Skema Enkripsi Deskripsi Blowfish

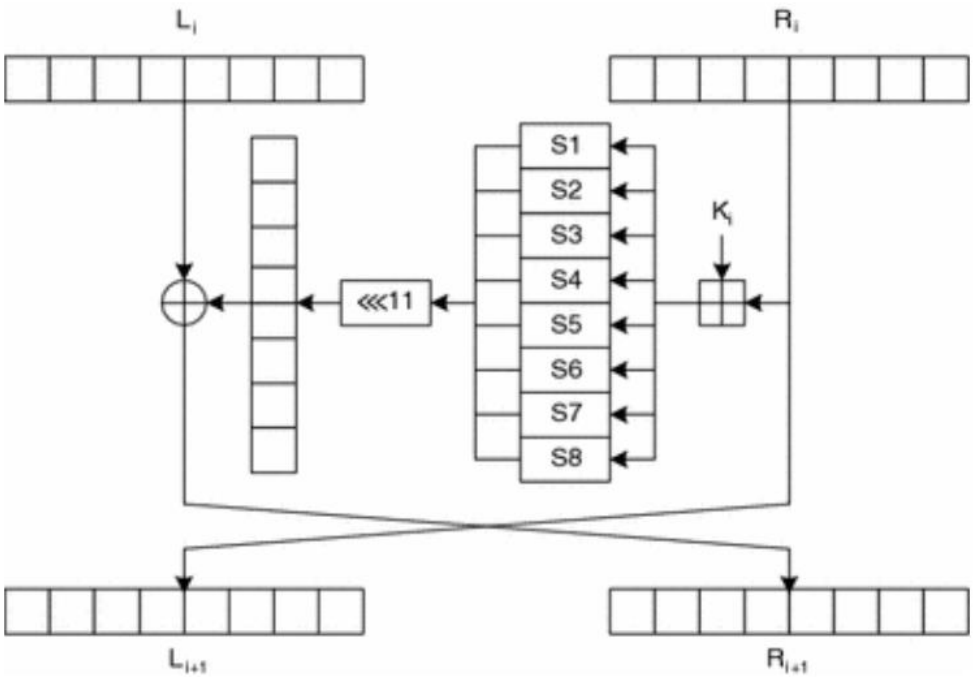
Algoritma Blowfish berbasis jaringan Feistel yang memproses data dalam blok 64-bit dengan panjang kunci variabel hingga 448 bit. Proses enkripsi melibatkan 16 putaran, di mana setiap putaran terdiri dari operasi XOR, substitusi non-linear menggunakan fungsi F berbasis S-box, dan pertukaran posisi antara dua bagian data, xL dan xR. Kunci yang digunakan diperluas melalui P-array untuk setiap putaran. Setelah putaran selesai, hasil akhirnya digabungkan kembali untuk menghasilkan ciphertext, dengan desain yang memastikan efek avalanche untuk keamanan tinggi terhadap serangan kriptografi. Gambar 2. menggambarkan enkripsi dan dekripsi Blowfish dengan 16 putaran jaringan Feistel menggunakan operasi XOR dan S-box untuk mengubah dan mengembalikan data 64-bit [19].



Gambar 2. Proses Enkripsi Deskripsi Blowfish

2.3. Skema Enkripsi Deskripsi Gost

Algoritma GOST mengenkripsi data 64-bit dengan membaginya menjadi dua bagian: blok kiri (L) dan blok kanan (R), masing-masing 32-bit. Prosesnya terdiri dari 32 putaran, di mana blok kanan diolah menggunakan fungsi mangler yang mencakup operasi XOR dengan kunci 32-bit, penambahan modular, substitusi melalui S-Box, dan rotasi kiri siklik. Hasilnya di-XOR dengan blok kiri untuk menghasilkan blok kanan baru, sementara blok kanan awal menjadi blok kiri baru. Dekripsi dilakukan dengan urutan kunci yang dibalik, menggunakan langkah yang sama. GOST menggunakan kunci 256-bit dan menyembunyikan nilai S-Box untuk keamanan maksimal. Gambar 3. Menunjukan satu putaran algoritma GOST dalam proses enkripsi [20].



Gambar 3. Proses Enkripsi Dekripsi Gost

3. Hasil dan Pembahasan

Penelitian ini mengandalkan sejumlah berkas yang mengandung teks biasa (dalam format *.txt) untuk melakukan pengujian terhadap proses enkripsi dan deskripsi menggunakan algoritma Blowfish dan Gost. Berikut adalah beberapa data teks (dalam format *.txt) yang akan diterapkan terlampir pada Tabel 1.

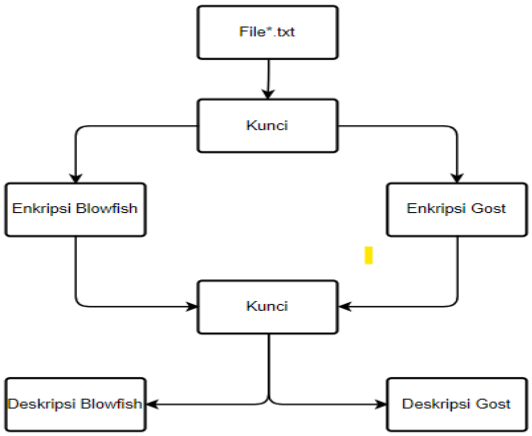
Tabel 1. file yang akan diujikan

No	File [*.txt]	Ukuran Asli
1.	Sample 1	481 bytes
2.	Sample 2	289 bytes
3.	Sample 3	309 bytes
4.	Sample 4	254 bytes
5.	Sample 5	230 bytes

3.1. Simulasi Enkripsi dan Deskripsi

Simulasi enkripsi dan deskripsi yang untuk mengamankan data teks dalam penelitian ini memanfaakan CryptoForge, seperti yang terlihat pada Gambar 4. Menunjukan alur proses pengujian dari simulasi enkripsi dan deskprisi pada data teks (dalam format *.txt) dengan menggunakan Algoritma Blowfish dan Gost melalui CryptoForge.

p-ISSN: 2301-5373
e-ISSN: 2654-5101



Gambar 4. Alur Enkripsi Deskrpsi

3.2. Perbandingan Ukuran Enkripsi Blowfish dan Gost

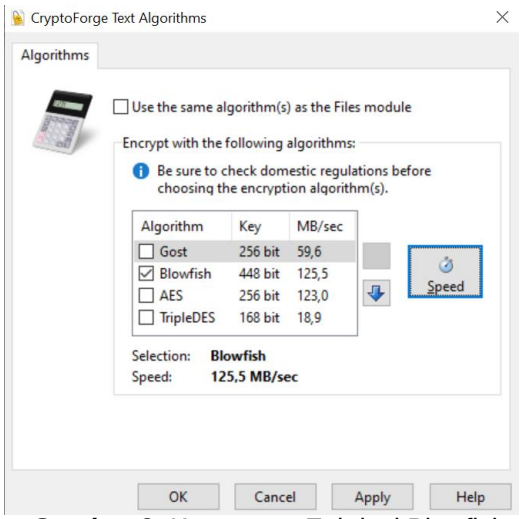
Hasil simulasi algoritma Blowfish dan GOST ditampilkan secara lengkap dalam Tabel 3. Proses enkripsi dan dekripsi dengan kunci 448 untuk algoritma Blowfish da 256 bit untuk algoritma Gost menunjukkan bahwa ukuran file setelah proses tersebut berbeda dari ukuran aslinya. Terlihat juga ukuran file hasil enkripsi dan dekripsi menggunakan algoritma GOST dengan Blowfish menghasilkan ukuran chipertext yang sama. Ini dikarenakan kenyataan bahwa baik Blowfish maupun GOST menggunakan pendekatan pemrosesan data dalam blok berukuran tetap, yaitu 64 bit. Karena keduanya membagi data menjadi blok-blok berukuran serupa sebelum proses enkripsi, hasil akhirnya adalah ciphertext dengan ukuran yang sama, meskipun panjang kunci berbeda

Tabel 2. Perbandingan Kinerja Ukuran Blowfish dengan Gost

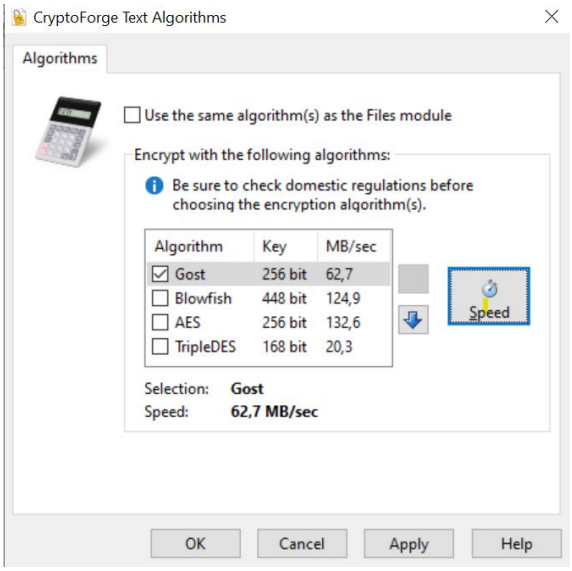
No	File (*.txt)	Ukuran Asli	Blowfish 448 bit		Gost 256 bit	
			Enkripsi	Deskripsi	Enkripsi	Deskripsi
1	Sample 1	481 bytes	782 bytes	481 bytes	782 bytes	481 bytes
2	Sample 2	289 bytes	380 bytes	289 bytes	380 bytes	289 bytes
3	Sample 3	309 bytes	424 bytes	309 bytes	424 bytes	309 bytes
4	Sample 4	254 bytes	412bytes	254 bytes	412 bytes	254 bytes
5	Sample 5	230 bytes	400 bytes	230 bytes	400 bytes	230 bytes

3.3. Perbandingan Kecepatan Enkripsi Blowfish dan Gost

Gambar 9. dan 10. menunjukkan bahwa algoritma Blowfish dengan kunci 448-bit memiliki kecepatan enkripsi lebih tinggi (sekitar 124–125 MB/sec) dibandingkan dengan Gost berukuran kunci 256 bit (sekitar 59–63 MB/sec). Blowfish lebih cepat karena desainnya yang efisien dengan struktur Feistel, menggunakan operasi sederhana seperti XOR yang mempercepat pemrosesan data. Sementara itu, Gost memiliki proses enkripsi yang lebih kompleks, sehingga membutuhkan waktu komputasi lebih lama.



Gambar 9. Kecepatan Enkripsi Blowfish



Gambar 10. Kecepatan Enkripsi Gost

4. Kesimpulan

Mengacu pada hasil simulasi data teks berformat *.txt digunakan algoritma Blowfish dengan panjang kunci 448-bit dan algoritma Gost dengan kunci 256 bit, keduanya mampu menjalankan enkripsi dan dekripsi data teks. Dalam simulasi menggunakan Blowfish, ukuran byte hasil enkripsi tidak sama dengan ukuran data teks aslinya. Hal serupa terjadi pada algoritma Gost, di mana ukuran hasil enkripsinya juga berbeda dari ukuran teks aslinya Akan tetapi ukuran yang dihasilkan algoritma Blowfish maupun Gost menghasilkan ukuran yang sama meskipun menggunakan panjang kunci yang tidak setara dikarenakan kedua algoritma tersebut sama sama memproses data dalam blok yang sama 64 bits. Kecepatan enkripsi algoritma Blowfish lebih tinggi daripada algoritma Gost, untuk blowfish memiliki kecepatan (sekitar 124–125 MB/sec) sedangkan Gost (sekitar 59–63 MB/sec). Meskipun ukuran kunci Blowfish lebih besar, algoritma ini lebih efisien untuk enkripsi cepat, menjadikannya pilihan yang lebih baik untuk aplikasi yang membutuhkan kecepatan tinggi, sementara Gost dapat digunakan sebagai alternatif dengan kunci yang lebih kecil.

Penelitian selanjutnya dapat melakukan perbandingan dengan algoritma kriptografi lainya untuk dapat hasil yang lebih baik dan juga mempertimbangkan pengujian dengan berbagai jenis file lain, seperti docx, pdf, atau jpeg, untuk mengevaluasi bagaimana kedua algoritma tersebut menangani berbagai format data.

Daftar Pustaka

[1] M. F. Mushtaq, S. Jamel, A. H. Disina, Z. A. Pindar, and N. S. Ahmad, “Survei tentang Enkripsi Kriptografi Algoritma,” pp. 333–344, 2017.

[2] A. C. Kusuma and A. D. Rahmani, “Analisis Yuridis Kebocoran Data Pada Sistem Perbankan Di Indonesia (Studi Kasus Kebocoran Data Pada Bank Indonesia),” *SUPREMASI : Jurnal Hukum*, vol. 5, no. 1, pp. 46–63, 2022. DOI: 10.36441/supremasi.v5i1.721

[3] Putra Rahmadi and Hilda Dwi Yunita, “Implementasi Pengamanan Basis Data Dengan Teknik Enkripsi,” *Jurnal Cendikia*, vol. 19, no. 1, pp. 413–418, 2020.

[4] A. Hamzah and B. Kumar, “Standar Enkripsi RSA,” 2021.

[5] D. Kumar Sharma, N. Chidananda Singh, D. A. Noola, A. Nirmal Doss, and J. Sivakumar, “A review on various cryptographic techniques & algorithms,” *Materials Today: Proceedings*, vol. 51, no. xxxx, pp. 104–109, 2021. DOI: 10.1016/j.matpr.2021.04.583

[6] M. A. Al-Shabi, “A Survey on Symmetric and Asymmetric Cryptography Algorithms in information Security,” *International Journal of Scientific and Research Publications (IJSRP)*, vol. 9, no. 3, p. p8779, 2019. DOI: 10.29322/ijsrp.9.03.2019.p8779

[7] B. Dan and A. Rsa, “ANALISIS PERFORMA KRIPTOGRAFI HYBRID ALGORITMA PENDAHULUAN Keamanan informasi merupa-kan salah satu masalah penting , seiring

dengan perkembangan software dan pengguna internet . Keamanan komputer berhubungan dengan pencegahan dari pencurian data atau inf," vol. VI, no. 1, 2019.

[8] R. N. Fitriana and D. Djuniadi, "Analisis Perbandingan Algoritma AES Dan RC4 Pada Enkripsi dan Dekripsi Data Teks Berbasis CrypTool 2," *Systemic: Information System and Informatics Journal*, vol. 7, no. 2, pp. 1–7, 2022. DOI: 10.29080/systemic.v7i2.1263

[9] F. Fitriyadi and A. Ahwan, "Implementasi Algoritma One Time Pad Untuk Enkripsi dan Dekripsi pada Peresepan Data Obat di Puskesmas Purwodiningratan Surakarta," *Smart Comp: Jurnalnya Orang Pintar Komputer*, vol. 12, no. 4, 2023. DOI: 10.30591/smartcomp.v12i4.3958

[10] D. Astuti and C. Sundari, "Implementasi Algoritma Vigenere Cipher Untuk Enkripsi Dan Dekripsi Pada Peresepan Data Obat Di Puskesmas Mertoyudan 1 Kabupaten Magelang," *Jurnal Teknik Informasi dan Komputer (Tekinkom)*, vol. 5, no. 2, p. 341, 2022. DOI: 10.37600/tekinkom.v5i2.534

[11] M. A. Wijaya, W. Kurniawan, and A. Kusyanti, "Perancangan dan Implementasi Algoritma Enkripsi Idea pada Perangkat Kriptografi Berbasis FPGA," *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer (J-PTIIK) Universitas Brawijaya*, vol. 2, no. 12, pp. 6973–6981, 2018.

[12] Yusuf Ramadhan Nasution, Heri Santoso, and S. W. Amalia, "Penerapan Algoritma Vernam Dalam Mengamankan Dokumen Pdf," *Jurnal Informatika dan Rekayasa Elektronik*, vol. 6, no. 1, pp. 37–46, 2023. DOI: 10.36595/jire.v6i1.804

[13] R. Wardhani, S. R. Nurshiami, and N. Larasati, "Komputasi Enkripsi Dan Dekripsi Menggunakan Algoritma Hill Cipher," *Jurnal Ilmiah Matematika dan Pendidikan Matematika*, vol. 14, no. 1, p. 45, 2022. DOI: 10.20884/1.jmp.2022.14.1.5727

[14] N. P. S. Winarno and T. A. Cahyanto, "Penggunaan Karakter Kontrol ASCII Untuk Integrasi Data Pada Hasil Enkripsi Algoritma Caesar Cipher," *INFORMAL: Informatics Journal*, vol. 6, no. 3, p. 197, 2021. DOI: 10.19184/isj.v6i3.21091

[15] I. N. Purnama, "Implementasi Algoritma Enkripsi Rc5 Untuk Mengamankan Gambar Pada Perangkat Android," *Jurnal Informatika dan Rekayasa Elektronik*, vol. 2, no. 2, p. 1, 2019. DOI: 10.36595/jire.v2i2.108

[16] A. B. Nasution, "Implementasi Pengamanan Data Dengan Menggunakan Algoritma Caesar Cipher Dan Transposisi Cipher," *Jurnal Teknologi Informasi*, vol. 3, no. 1, p. 1, 2019. DOI: 10.36294/jurti.v3i1.680

[17] R. Minarni, "Implementasi Algoritma Base64 Untuk Mengamakan Sms Pada Smartphone," *Building of Informatics, Technology and Science (BITS)*, vol. 1, no. 1, pp. 28–33, 2019. DOI: 10.47065/bits.v1i1.3

[18] Cryptoforge.com, "cryptoforge.com." [Online]. Available: https://www.cryptoforge.com/

[19] S. Manku and K. Vasanth, "Blowfish encryption algorithm for information security," *ARPN Journal of Engineering and Applied Sciences*, vol. 10, no. 10, pp. 4717–4719, 2015.

[20] A. Claude, B. Linwa, and G. Bassam, "Survey of Cryptography Algorithms for Sub- Saharan Countries," *African Journals Online*, vol. 17, no. January, p. 2, 2021.

This page is intentionally left blank.