Terakreditasi SINTA Peringkat 2
Surat Keputusan Dirjen Penguatan Riset dan Pengembangan Ristek Dikti No. 10/E/KPT/2019
masa berlaku mulai Vol. 1 No. 1 tahun 2017 s.d Vol. 5 No. 3 tahun 2021

Terbit online pada laman web jurnal: **http://jurnal.iaii.or.id**

JURNAL RESTI
(Rekayasa Sistem dan Teknologi Informasi)
Vol. 4 No. 6 (2020) 1209 – 1215 ISSN Media Elektronik: 2580-0760

Peningkatan Keamanan Kriptografi *Caesar Cipher* dengan Menerapkan Algoritma Kompresi "*Stout Codes*"

Mesran[1], Surya Darma Nasution[2]
[1,2]Prodi Teknik Informatika, Fakultas Ilmu Komputer & Teknologi Informasi, Universitas Budi Darma, Medan, Indonesia
[1]mesran.skom.mkom@gmail.com, [2]surya.darma.nasution1@gmail.com

Abstract

The development of applications that can crack passwords or data that has been secured by cryptographic algorithms has made researchers try various ways to further secure the data they have. Even though the method used is quite modern, the algorithm for breaking ciphertext in cryptography can still be created. For this reason, in this study the authors develop cryptographic security by combining it with data compression. In this study, the algorithm used is the classic cryptographic algorithm, namely Caesar Cipher, combined with the Stout Codes compression algorithm. The results of the Caesar Cipher ciphertext are then compressed by applying the Stout Codes algorithm. From the test results using the Brute Force Atack model, the results are quite good, that the security of the encrypted data is quite good and not easily broken.

Keywords: security, criptography, Caesar Cipher, compression, Stout Codes

Abstrak

Berkembangnya aplikasi yang dapat memecahkan sandi atau data yang telah diamankan dengan algoritma kriptografi, membuat peneliti mencoba beragam cara untuk lebih mengamankan data yang dimiliki. Walaupun metode yang digunakan sudah cukup modern, namun algoritma pemecah chiperteks pada kriptografi tetap saja bisa di ciptakan. Untuk itu pada penelitian ini penulis melakukan pengembangan keamanan kriptografi dengan mengkombinasikan dengan kompresi data. Dalam penelitian ini algoritma yang digunakan algoritma kriptografi klasik yaitu *Caesar Cipher* yang dikombinasikan dengan algoritma kompresi *Stout Codes*. Hasil chiperteks *Caesar Cipher* kemudian dilakukan kompresi dengan menerapkan algoritma *Stout Codes*. Dari hasil pengujian dengan menggunakan model *Brute Force Atack* mendapati hasil yang cukup baik, bahwa keamanan terhadap data yang di enkripsi cukup baik dan tidak mudah dipecahkan.

Kata kunci: keamanan, kriptografi, *Caesar Cipher*, kompresi, *Stout Codes*.

1. Pendahuluan

Keamanan data menjadi aspek yang harus diperhatikan didalam perkembangan teknologi, dimana data yang diolah atau yang dikirim dalam suatu transmisi data bisa saja jatuh ke pihak-pihak yang tidak bertanggung jawab. Pengamanan terhadap suatu data semakin hari juga semakin berkembang, hal ini dikarenakan banyak algoritma kriptografi telah dapat dipecahkan oleh piranti lunak yang memang dibuat untuk memecahkan algoritma-algoritma kriptografi. Selain menerapkan algoritma kriptografi yang baru, biasanya banyak peneliti memodifikasi suatu algoritma kriptografi. Tentu saja hal ini bukanlah suatu pekerjaan yang mudah.

Dalam penelitian terdahulu yang pernah dilakukan oleh Benni Purnama dan Hetty Rohayani.AH dengan memodifikasi algoritma kriptografi *Caesar Cipher* yang menghasilkan suatu cipherteks yang bisa terbaca sehingga pasti pihak lain atau kriptanalis tidak akan curiga dengan pesan yang telah dienkripsi [1].

Priya Verma, dkk juga melakukan penelitian tentang pengembangan *Caesar Cipher* untuk keamanan yang lebih baik, hasil yang didapat dalam penelitian ini yaitu dalam teknik yang dikembangkan berisi simbol, angka, huruf besar dan karakter huruf kecil yang meningkatkan kinerja sandi Caesar. Ukuran kunci yang digunakan 82, sehingga semakin sulit untuk mendekripsi pesan dan terbukti lebih aman [2].

Kemudian Anupama Mishra juga melakukan penelitian tentang peningkatan keamanan *Caesar Cipher* dengan menggunakan metode kriptografi yang berbeda, hasil yang dicapai dalam penelitian tersebut yaitu dengan mengkombinasikan dari dua teknik klasik ini

Diterima Redaksi : 07-12-2020 | Selesai Revisi : 19-12-2020 | Diterbitkan Online : 30-12-2020

menghasilkan sandi yang lebih aman dan kuat sehingga sangat sulit untuk dipecahkan[3].

Penelitian lainnya juga dilakukan oleh Fahrul Ikhsan Lubis, dkk tentang kombinasi dari modifikasi algoritma *Caesar Cipher* digabungkan dengan cipher transposisi, dalam tiga kali enkripsi pada percobaan yang dilakukan yaitu modifikasi Caesar pada awalnya kemudian ciphertext yang dihasilkan akan dienkripsi dengan transposisi, dan terakhir, hasil dari transposisi akan dienkripsi lagi dengan modifikasi Caesar kedua, demikian pula pada proses dekripsi tetapi prosesnya dilakukan secara terbalik. Hasil yang dicapai sangat bagus dan untuk mengatasinya ada banyak kemungkinan yang harus dilakukan oleh kriptanalis [4].

Enas Ismael Imran dan Farah abdulameerabdulkareem pada penelitiannya tentang peningkatan *Caesar Cipher* untuk Keamanan Lebih Baik menunjukkan bahwa *Caesar Cipher* menjadi salah satu algoritma untuk mengenkripsi yang paling sederhana dan banyak teknik yang dapat digunakan untuk memperkuat bahkan melebihi apa yang dapat dicapai oleh algoritma *Caesar Cipher* tersebut[5].

O.E. Omolara, dkk dalam penelitiannya tentang pengembangan *Hybrid Caesar Cipher* dan *Vigenere Cipher* yang dimodifikasi untuk Komunikasi Data, pengujian yang dilakukan menghasilkan persentase keamanan yang tinggi sehingga menjadi sandi yang sangat kuat dan sulit dipecahkan [6].

Atish Jain, dkk dalam penelitiannya untuk meningkatkan keamanan penggantian *Caesar Cipher* menggunakan pendekatan acak, penelitian ini bertujuan untuk mengusulkan sebuah versi yang disempurnakan dari teknik substitusi sandi Caesar yang dapat mengatasi segala keterbatasan yang dihadapi klasik *Caesar Cipher* [7].

Imam Wahyu Utomo, dkk telah melakukan penelitian mengenai aplikasi kriptografi yang berbasis android dimana penelitian ini menggunakan penggabungan antara algoritma *Caesar Cipher* dan *Vigenere Cipher*. Hasil yang dicapai dalam penelitian tersebut adalah dalam melakukan proses enkripsi dan dekripsi tingkat kesuksesannya 75% [8].

Penelitian lainnya juga dilakukan oleh Primaningtyas Nur Arifah dan Windi Agustiar Basuki mengenai pengimplementasian algoritma kriptografi Caesar Chiper dengan menggunakan Matlab R2013a, di penelitian ini dilakukan sedikit modifikasi dimana karakter yang diamankan selain huruf A-Z juga menambahkan karakter .,!? dan juga angka 0-9. Hasil dari penelitian ini dinilai dapat mempersulit pihak-pihak yang kurang bertanggung jawab untuk memecahkan sistem keamanan yang dibuat [9].

Sementara Muhammad Lutfi Wijaya, dkk melakukan penelitian tentang kriptografi dengan komposisi *Caesar Cipher* dan *Affine Cipher* untuk mengamankan pesan rahasia. Pada penelitian ini dilakukan kombinasi dengan melakukan dua kali proses enkripsi untuk mengamankan pesan rahasia, serta melakukan dua kali proses dekripsi untuk mengembalikan ke pesan semula [10].

Dari penelitian-penelitian yang pernah dilakukan sebelumnya dapat disimpulkan bahwa dengan mengembangkan atau memodifikasi suatu algoritma kriptografi dapat meningkatkan level keamanan data, walaupun algoritma yang dikembangkan berupa algoritma *Caesar Cipher*. Seperti diketahui bahwa *Caesar Cipher* merupakan algoritma kriptografi klasik yang level keamanannya sangat rendah dan mudah dipecahkan.

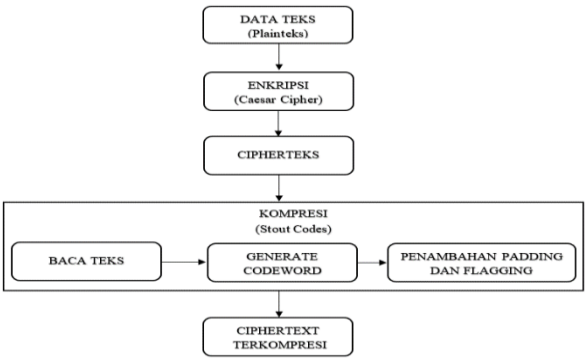
Dalam penelitian ini dilakukan kombinasi algoritma yang bertujuan untuk meningkatkan kemanan data. Peningkatan kemanan data dilakukan dengan menggunakan algoritma yang digunakan untuk kompresi data, walaupun algoritma yang digunakan berbeda tujuannya tetapi hal ini bisa saja menjadi solusi yang tepat dalam mengamankan data karena hasil dari pengompresian suatu data berupa karakter yang berbeda dari karakter sebelum dilakukan proses kompresi.

Penelitian yang mengkombinasikan algoritma kriptografi dengan algoritma kompresi juga sudah pernah dilakukan sebelumnya oleh Ruchita Sharma dan Swarnalata Bollavarapu, kesimpulan yang dihasilkan yaitu jika kombinasi yang digunakan membuat lebih mudah, menghemat waktu dan menjadi lebih aman. Dipenelitian tersebut juga mengharapkan di masa mendatang, teknik kriptografi yang berbeda dapat diterapkan dengan kombinasi teknik kompresi [11].

Berdasarkan dari penelitian-penelitian terdahulu yang sudah melakukan pengembangan terhadap algoritma kriptografi, dan bahkan mengkombinasikan dengan algoritma kompresi, maka dalam penelitian ini algoritma kriptografi yang akan dicoba untuk dikombinasikan adalah *Caesar Cipher*. Sedangkan untuk algortima kompresi yang digunakan yaitu *Stout Codes*.

2. Metode Penelitian

Alur metode penelitian dari penelitian yang akan dilakukan dapat dilihat pada Gambar 1.

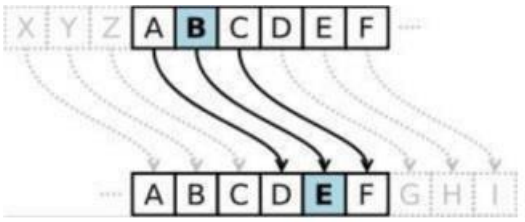


Gambar 1. Alur Metode Penelitian

Berdasarkan informasi yang ditunjukan pada Gambar 1, maka dapat dilihat bahwa terdapat dua proses utama yang dilakukan. Proses pertama yaitu melakukan proses enkripsi menggunakan *Caesar Cipher*, lalu hasil dari proses tersebut akan diproses lagi menggunakan *Stout Codes* sehingga menghasilkan cipherteks hasil kompresi.

2.1.*Caesar Cipher*

Metode penyandian dalam kriptografi klasik yang paling terkenal yaitu *Caesar Chiper*, pertama sekali digunakan oleh Julius Caesar untuk melakukan komunikasi dengan panglimanya [12],[13]. *Caesar Chiper* merupakan teknik kriptografi yang paling sederhana dan banyak digunakan, dimana setiap huruf pada plainteksnya digantikan dengan huruf lain dengan pergeseran sebanyak nilai kunci [14]–[16]. Dalam penelitian ini, nilai kunci dari *Caesar Cipher* adalah 3. Untuk proses pergeseseran karakter di *Caesar Cipher* dapat dilihat pada Gambar 2.



Gambar 2. Proses Pergeseran dalam *Caesar Cipher*

Contoh :

Plainteks : AYO KITA PERGI

Cipherteks : DBR NLWD SHUJL

Untuk proses dekripsi dari *Caesar Cipher*, hanya perlu melakukan pergeseran kebalikan sebanyak 3.

2.2. *Stout Codes*

Kompresi memiliki arti mengompresi atau mengecilkan ukuran. Kompresi data adalah proses pengkodean informasi menggunakan bit lain yang memiliki nilai lebih rendah daripada representasi data yang tidak dikodekan [17]–[20]. Pada algoritma kompresi *Stout Codes*, sangat mirip dengan kode *Elias omega* dan *Even-Rodeh*. *Codeword* yang dihasilkan oleh algoritma *stout code* bergantung pada pilihan parameter l yang lebih besar dari atau sama dengan 2. Algoritma *Stout Code* diperkenalkan oleh Quentin Stout dengan dua keluarga yaitu rekursi [21][18]. Pada rumpun R_l, semakin banyak kelompok panjang yang dibaca sampai kelompok tersebut ditemukan diikuti oleh 0. Gunakan notasi L = 1 +[Log_2 n] dan dilambangkan dengan representasi biner B (n, l) dari l bit dari bilangan bulat n. Jadi, B (12,5) = 01100. Untuk $l \geq 2$ prefiks defenisikan dengan:

$$R_l\, n = \begin{cases} B(n,L), & for\ 0 \leq n \leq 2^l - 1, \\ R_l\,(L)B(n,L), & for\ n \geq 2^l \end{cases} \quad (1)$$

Keluarga kedua dari kode *Stout* memiliki metode yang serupa, tetapi dengan awalan berbeda yang dilambangkan dengan S_l(n). Untuk nilai kecil l, keluarga ini menawarkan beberapa peningkatan dibandingkan dengan kode R_l. Khususnya, menghilangkan sedikit redundansi di R_l kode karena grup panjang tidak boleh 0 (itulah sebabnya kelompok panjang dalam kode omega menyandikan Li-1 dan bukan Li). Awalan S_l mirip dengan awalan R_l dengan perbedaan bahwa kelompok panjang untuk Li menyandikan Li −1 - l. Awalan S_l (n) didefinisikan secara rekursif oleh:

$$S_l\, n = \begin{cases} B(n,L), & for\ 0 \leq n \leq 2^l - 1, \\ R_l\,(L-1-l)B(n,L), & for\ n \geq 2^l \end{cases} \quad (2)$$

3. Hasil dan Pembahasan

3.1. Pengembangan Teknik Keamanan Yang Diusulkan

Dalam penelitian ini, nilai kunci dari *Caesar Cipher* adalah 3, dan *Stout Codes* yang digunakan yaitu dengan menggunakan pengkodean dari S_l (n). Sebagai contoh untuk proses enkripsi menggunakan *Caesar Cipher* dan pengompresian menggunakan *Stout Codes*, disini dicontohkan dengan plainteks sebagai berikut :

Plainteks : AYO KITA PERGI

Cipherteks : DBR NLWD SHUJL

Setelah dilakukan proses enkripsi menggunakan *Caesar Cipher*, maka dilakukan proses kompresi menggunakan *Stout Codes*. Langkah-langkah yang dilakukan untuk pengompresiannya yaitu :

1. Mengurutkan karakter berdasarkan frekuensinya.
   Melakukan pengurutan dari teks "DBR NLWD SHUJL", pengurutan dilakukan berdasarkan frekuensi dari yang paling besar ke frekuensi yang paling kecil. Pengurutan dapat dilihat pada Tabel 1.

Tabel 1. Pengurutan Berdasarkan Frekuensi

n	Karakter	Frek
1	D	2
2	Spasi	2
3	L	2
4	B	1
5	R	1
6	N	1
7	W	1
8	S	1
9	H	1
10	U	1
11	J	1

2. Pembentukan *Codeword* dan membuat stringbit.

Dalam melakukan pembentukan *codeword*, terlebih dahulu menentukan nilai *l*. Disini kita mengambil nilai *l*=2.

a. Untuk nilai $0 \le n \le 2^l-1$
   Karena nilai *l*=2, maka $0 \le n \le 2^2-1= 3$
   1) n = 1, maka *codeword* yang dihasilkan berupa nilai biner dari n yaitu 1 dan diambil sebanyak *l* yaitu 2 sehingga *codeword* yang diperoleh adalah 01.
   2) n = 2, maka *codeword* yang dihasilkan berupa nilai biner dari n yaitu 10 dan diambil sebanyak *l* yaitu 2 sehingga codeword yang diperoleh adalah 10.
   3) n = 3, maka *codeword* yang dihasilkan berupa nilai biner dari n yaitu 11 dan diambil sebanyak *l* yaitu 2 sehingga *codeword* yang diperoleh adalah 11.

b. Untuk nilai $n= 2^l$
   Karena nilai *l*=2, maka $n \ge 2^2 = 4$
   1) n = 4 dengan nilai biner 100, maka nilai L = 3 diambil dari panjang bit biner nilai n dan langkah selanjutnya mencari nilai R.
      R_2(3-1-2) = 0, nilai binernya adalah 0 dan diambil sebanyak nilai *l* yaitu 2 sehingga nilainya dari R = 00.
      R_l(L-1-l) B (n, L) = 00 100
   2) n = 5 nilai binernya 101, maka nilai L = 3.
      R_2(3-1-2) = 0, nilai binernya adalah 0 dan diambil sebanyak nilai *l* yaitu 2 sehingga nilainya dari R = 00.
      R_l(L-1-l) B (n, L) = 00 101
   3) n = 6 nilai binernya 110, maka nilai L = 3.
      R_2(3-1-2) = 0, nilai binernya adalah 0 dan diambil sebanyak nilai *l* yaitu 2 sehingga nilainya dari R = 00.
      R_l(L-1-l) B (n, L) = 00 110
   4) n = 7 nilai binernya 111, maka nilai L = 3.
      R_2(3-1-2) = 0, nilai binernya adalah 0 dan diambil sebanyak nilai *l* yaitu 2 sehingga nilainya dari R = 00.
      R_l(L-1-l) B (n, L) = 00 111
   5) n = 8 nilai binernya 1000, maka nilai L = 4.
      R_2(4-1-2) = 1, nilai binernya adalah 1 dan diambil sebanyak nilai *l* yaitu 2 sehingga nilainya dari R = 01.
      R_l(L-1-l) B (n, L) = 01 1000
   6) n = 9 nilai binernya 1001, maka nilai L = 4.
      R_2(4-1-2) = 1, nilai binernya adalah 1 dan diambil sebanyak nilai *l* yaitu 2 sehingga nilainya dari R = 01.
      R_l(L-1-l) B (n, L) = 01 1001
   7) n = 8 nilai binernya 1010, maka nilai L = 4.
      R_2(4-1-2) = 1, nilai binernya adalah 1 dan diambil sebanyak nilai *l* yaitu 2 sehingga nilainya dari R = 01.
      R_l(L-1-l) B (n, L) = 01 1010
   8) n = 8 nilai binernya 1011, maka nilai L = 4.
      R_2(4-1-2) = 1, nilai binernya adalah 1 dan diambil sebanyak nilai *l* yaitu 2 sehingga nilainya dari R = 01.
      R_l(L-1-l) B (n, L) = 01 1011

Setelah selesai membentuk *codeword*, berikutnya mengganti karakter dengan *codeword* lalu menghasilkan *string bit*. Tabel 2 menunjukan penggantian karakter dengan *codeword*.

Tabel 2. Mengganti Karakter dengan *Codeword*

n	Karakter	Frek	*Codeword*
1	D	2	01
2	Spasi	2	10
3	L	2	11
4	B	1	00100
5	R	1	00101
6	N	1	00110
7	W	1	00111
8	S	1	011000
9	H	1	011001
10	U	1	011010
11	J	1	011011

Setelah mengganti karakter dengan *codeword*, maka yang harus dilakukan berikutnya menyusun sesuai kalimat yang akan dikompresi.

D	B	R	Spasi	
01	00100	00101	10	
N	L	W	D	Spasi
00110	11	00111	01	10
S	H	U	J	L
011000	011001	011010	011011	11

Dari proses yang dilakukan diatas, maka hasil dari string bitnya, yaitu : 0100100001011000110110011101100110000110010110100110111 1.

3. Penambahan *padding* dan *flagging*.

Penambahan *padding* dan *flagging* dibutuhkan agar jumlah stringbit yang dihasilkan dari proses kompresi habis dibagi 8. Jumlah bit dari stringbit yaitu 56 dan habis dibagi 8, sehingga tidak perlu ditambahkan padding, tetapi tetap diperlukan flagging. Jika jumlah bit dari hasil string bit tidak habis dibagi 8, maka tambahkan 0 sebanyak 7 - n + "1" di ujung bit string, dimana n= sisa hasil bagi tersebut.

Penambahan *flagging* dilakukan dengan menggunakan rumus 9-n. Karena tidak ada penambahan padding, maka nilai n=8, sehingga 9-8 = 1 dinyatakan dalam bentuk bilangan biner 8-bit sehingga menghasilkan 00000001. Hasil dari string bit yang sudah ditambahkan *flagging* yaitu : 010010000101100011011001110110011000011001 011010011011110000001.

Dari string bit tersebut dibagi per 8 bit, lalu dirubah ke karakter, lihat Tabel 3.

Tabel 3. Hasil Kompresi

Bit	Karakter
01001000	H
01011000	X
11011001	Ù
11011001	Ù
10000110	†
01011010	Z
01101111	o
00000001	

Dari proses enkripsi dan proses kompresi, maka karakter yang dihasilkan yaitu : HXÙÙ†Zo .

3.2. Proses Dekompresi dan Dekripsi

Proses merubah chiperteks yang sudah terkompresi tersebut yaitu dengan melakukan dekompresi, lalu diikuti dengan proses dekripsi. Dekompresi dilakukan dengan merubah cipherteks hasil kompresi menjadi bilangan biner, seperti diperlihatkan pada tabel 4.

Setelah diubah ke biner, susun menjadi sebuah string bit. String bit yang dihasilkan yaitu : 0100100001011 000110110011101100110000110010110100110111110 0000001. Langkah berikutnya yaitu menghilangkan *padding* dan *flagging*, dengan cara mengambil 8 bit diakhir string bit dan merubahnya menjadi bilangan desimal. Dari *string bit* yang dihasilkan di atas, maka 8 bit terakhirnya yaitu "00000001" dan diubah menjadi desimal yaitu 1, dan nyatakan dengan n. Gunakan rumus 7+n untuk dapat mengetahui berapa bit yang harus dihilangkan pada string bit, 7+1 = 8 maka hilangkan 8 bit terakhir untuk menghasilkan string bit tanpa adanya padding dan flagging. Hasil dari string bit tanpa padding dan flagging yaitu : 010010000101100011011001110 110011000011001011010011011111.

Tabel 4. Merubah Karakter ke Biner

Karakter	Bit
H	01001000
X	01011000
Ù	11011001
Ù	11011001
†	10000110
Z	01011010
o	01101111
	00000001

Untuk mendapatkan cipherteks awal sebelum dikompresi, maka perlu dilakukan pengecekan string bit yang disesuaikan dengan Tabel 2. Lakukan pembacaan bit dari kiri ke kanan, jika ketemu dengan *codeword* pada Tabel 2, maka langsung diganti dengan karakter dengan *codeword* tersebut. Hasil dari proses pengecekan bit yaitu kalimat : DBR NLWD SHUJL.

Langkah selanjutnya untuk mendapatkan kembali plainteks yaitu dengan melakukan dekripsi. Dekripsi menggunakan pergeseran karakter ke belakang sebanyak 3 karakter.

Cipherteks : DBR NLWD SHUJL

Plainteks : AYO KITA PERGI

3.3. Perbandingan dengan *Caesar Cipher* Biasa

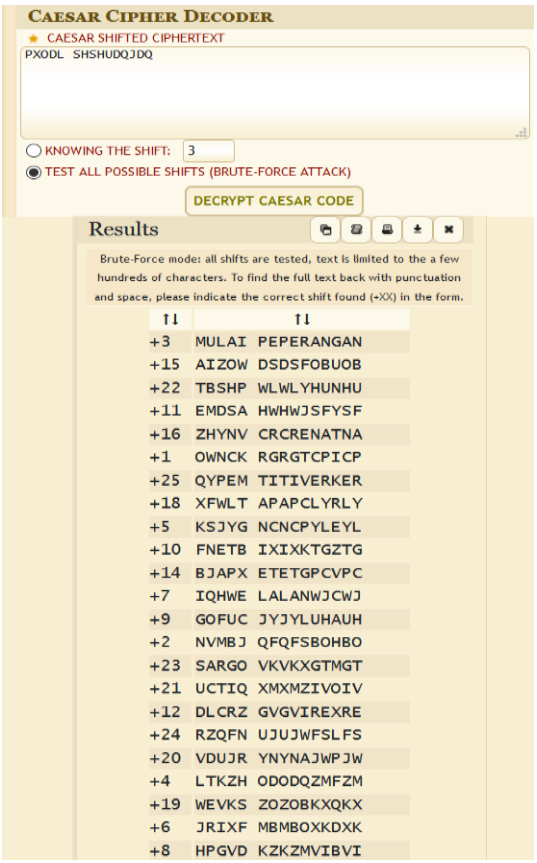
Jika dibandingkan dengan algoritma *Caesar Cipher* biasa, maka hasil yang diperoleh dapat dilihat pada Tabel 5.

Tabel 5. Perbandingan dengan *Caesar Cipher* Biasa

Plainteks	*Caesar Cipher*	Modifikasi Dengan *Stout Codes*
AYO KITA PERGI	DBR NLWD SHUJL	HXÙÙ†Zo
JUMPA DI MANHATTAN	MXPSD GL PDQKDWWDQ	1æ –¹#iJ‘
SIDANG MEJA HIJAU	VLGDQJ PHMD KLMDX	,dìl´Gn"á
MULAI PEPERANGAN	PXODL SHSHUDQJDQ	)Ž°ÍÛI I

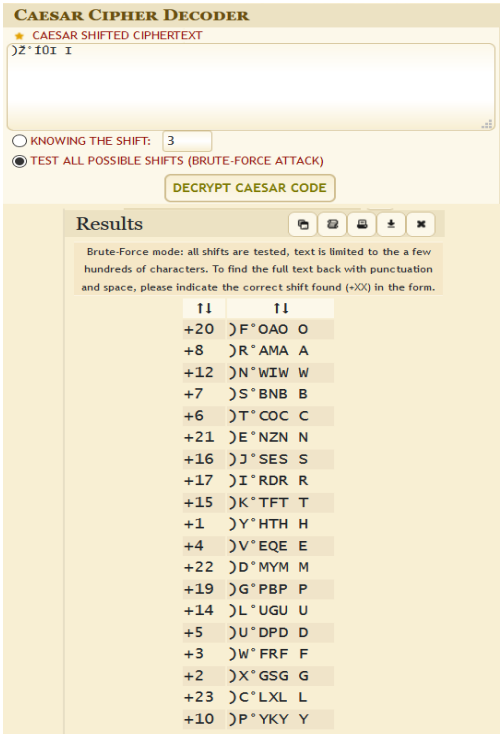
Dari Tabel 5 dapat kita lihat hasil perbandingan dari *Caesar Cipher* biasa dengan *Caesar Cipher* yang telah dimodifikasi dengan *Stout Codes*. Tampak jelas bahwa dengan memodifikasi dengan *Stout Codes*, hasil pengamanan teks seperti menggunakan algoritma kriptografi modern. Untuk menguji keamanan dari algoritma yang digunakan, maka dalam penelitian ini dilakukan uji coba untuk mengetahui apakah algoritma yang digunakan benar-benar aman sesuai dengan prinsip umum kriptografi yaitu *confidentiality*. Dalam aspek confidentiality, data atau informasi yang dikirim, diterima dan disimpan harus terjamin kerahasiaannya. Untuk menjamin kerahasiaan suatu data atau informasi dilakukan dengan menggunakan algoritma kriptografi

yang modern atau dengan memodifikasi algoritma kriptografi, hal ini perlu dilakukan karena ada suatu teknik untuk memecahkan sandi terhadap algoritma kriptografi yaitu *brute-force attack*. *Brute-force attack* yang dilakukan dengan tujuan mengetahui isi dari suatu data atau informasi. Pengujian yang dilakukan dalam penelitian ini diharapkan dapat mengatasi *brute-force attack*, sehingga bisa dikatakan data tersebut menjadi lebih aman. Pengujian dilakukan dengan melakukan *decoder* terhadap algoritma-algoritma kriptografi. Alamat dari laman tersebut yaitu https://www.dcode.fr/Caesar-cipher. Sebagai bahan pengujian, yang akan dicoba terlebih dahulu adalah cipherteks dari *Caesar Cipher* biasa tanpa menggunakan kunci, hanya menggunakan *brute-force attack* dengan metode *reverse brute force* sampai menemukan kombinasi yang tepat. untuk memecahkannya, dan hasilnya dapat dilihat pada Gambar 3.



Gambar 3. Pengujian Cipherteks *Caesar Cipher*

Dari Gambar 3 dapat dilihat bahwa dengan menggunakan *brute-force attack*, walaupun kunci tidak diketahui, tetap bisa dipecahkan dengan menemukan kalimat "MULAI PEPERANGAN". Dalam kasus ini kriptanalis dianggap sudah mengetahui bahwa algoritma yang digunakan adalah *Caesar Cipher*, atau anggap saja kriptanalis menguji ke semua algoritma kriptografi. Pengujian berikutnya akan dilakukan dengan cipherteks yang sudah dikompresi menggunakan *Stout Codes*. Hasil pengujiannya dapat dilihat pada Gambar 4.



Gambar 4. Pengujian Cipherteks *Caesar Cipher*

Dari Gambar 4 dapat dilihat bahwa dengan menggunakan *brute-force attack*, tetap tidak bisa memecahkan cipherteks yang sudah terkompresi dengan *Stout Codes*.

4. Kesimpulan

Berdasarkan pengujian yang telah dilakukan, maka dapat diambil kesimpulan bahwa kombinasi dari *Caesar Cipher* dan algoritma *Stout Codes* dapat meningkatkan keamanan data. Hasil cipherteks dari *Caesar Cipher* yang telah dikompresi terlihat seperti hasil enkripsi dari algoritma kriptografi modern. Pengembangan yang dilakukan yaitu mengkombinasikan teknik kriptografi *Caesar chiper* dengan algoritma kompresi *Stout Codes*, dapat menjadi solusi untuk mengamankan data yang penting.

Daftar Rujukan

[1] B. Purnama and A. H. H. Rohayani, "A New Modified Caesar Cipher Cryptography Method with LegibleCiphertext from a Message to Be Encrypted," *Procedia Comput. Sci.*, vol. 59, no. Iccsci, pp. 195–204, 2015.

[2] P. Verma, G. S. Gaba, and R. Miglani, "Diversified Caesar Cipher for Impeccable Security," *Int. J. Secur. Its Appl.*, vol. 11, no. 2, pp. 33–40, 2017.

[3] A. Mishra, "Enhancing Security of Caesar Cipher Using Different Methods," *Int. J. Res. Eng. Technol.*, vol. 02, no. 09, pp. 327–332, 2013.

[4] F. I. Lubis, H. F. S. Simbolon, T. P. Batubara, and R. W. Sembiring, "Combination of Caesar Cipher modification with transposition cipher," *Adv. Sci. Technol. Eng. Syst.*, vol. 2, no. 5, pp. 22–25, 2017.

[5] P. E. Ismael Imran and P. F. abdulameerabdulkareem, "Enhancement Caesar Cipher for Better Security," *IOSR J. Comput. Eng.*, vol. 16, no. 3, pp. 01–05, 2014.

[6] O. E. Omolara, A. I. Oludare, and S. E. Abdulahi, "Developing a Modified Hybrid Caesar Cipher and Vigenere Cipher for Secure Data Communication," *Issn*, vol. 5, no. 5, pp. 2222–1719, 2014.

[7] A. Jain, R. Dedhia, and A. Patil, "Enhancing the Security of Caesar Cipher Substitution Method using a Randomized Approach for more Secure Communication," *Int. J. Comput. Appl.*, vol. 129, no. 13, pp. 6–11, 2015.

[8] I. W. Utomo, R. Latifah, and D. Risanty, "Aplikasi Kriptografi Berbasis Android Menggunakan Algoritma Caesar Cipher dan Vigenere Cipher," *J. Sist. Informasi, Teknol. Inf. dan Komput.*, vol. 9, no. 2, pp. 142–149, 2018.

[9] P. N. Arifah and W. A. Basuki, "Implementasi Kriptografi Caesar Cipher," *Semin. Mat. DAN Pendidik. Mat. UNY 2017*, pp. 297–304, 2017.

[10] M. L. Wijaya, K. Yulianti, and H. S. Husain, "Caesar Cipher Dan Affine Cipher Untuk Mengubah Pesan Rahasia," *E u r e k a M a t i k a*, vol. 5, no. 1, pp. 30–45, 2017.

[11] R. Sharma and S. Bollavarapu, "Data Security using Compression and Cryptography Techniques," *Int. J. Comput. Appl.*, vol. 117, no. 14, pp. 15–18, 2015.

[12] S. K. Verma and D. B. Ojha, "An innovative Enciphering Scheme based on Caesar Cipher," *Int. J. Innov. Sci. Eng. Technol.*, vol. 1, no. 5, pp. 25–27, 2014.

[13] F. N. Nife, "A New Modified Caesar Cipher Cryptographic Method Along With Rail Fence to Encrypt Message 1-Dynamic Key Generation," *Int. J. Adv. Res.*, vol. 3, no. 2, pp. 331–335, 2015.

[14] Y. Dwi Putri, R. Rosihan, and S. Lutfi, "Penerapan Kriptografi Caesar Cipher Pada Fitur Chatting Sistem Informasi Freelance," *JIKO (Jurnal Inform. dan Komputer)*, vol. 2, no. 2, pp. 87–94, 2019.

[15] B. Oktaviana and A. P. Utama Siahaan, "Three-Pass Protocol Implementation in Caesar Cipher Classic Cryptography," *IOSR J. Comput. Eng.*, vol. 18, no. 04, pp. 26–29, 2016.

[16] S. Y. Wulandari, "Cryptography : A Combination of Caesar and Affine Cipher to Conceal the Message," *PROC. INTERNAT. CONF. SCI. ENGIN.*, vol. 3, no. April, pp. 741–744, 2020.

[17] L. Marlina, A. Putera, U. Siahaan, H. Kurniawan, and I. Sulistianingsih, "Data Compression Using Elias Delta Code," *Int. J. Recent Trends Eng. Res.*, vol. 3, no. 8, pp. 210–217, 2017.

[18] S. D. Nasution, "Data Compression Using Stout Codes," *IJICS (International J. Informatics Comput. Sci.*, vol. 3, no. 1, pp. 28–33, 2019.

[19] S. D. Nasution, G. L. Ginting, M. Syahrizal, and R. Rahim, "Data Security Using Vigenere Cipher and Goldbach Codes Algorithm," *Int. J. Eng. Res. Technol.*, vol. 6, no. 01, pp. 360–363, 2017.

[20] A. Singh and Y. Bhatnagar, "Enhancement of Data Compression Using Incremental Encoding," *Int. J. Sci. Eng. Res.*, vol. 3, no. 5, pp. 1457–1465, 2012.

[21] D. Salomon, D. Bryant, and G. Motta, "Handbook of Data Compression (Google eBook)", Fifth. Springer, 2010.