Penggunaan Karakter Kontrol ASCII Untuk Integrasi Data Pada Hasil Enkripsi Algoritma Caesar Cipher

Nur Putrananda Setyapuji Winarno*, Triawan Adi Cahyanto**
*Program Studi Teknik Informatika, Fakultas Teknik, Universitas Muhammadiyah Jember
**Program Studi Teknik Informatika, Fakultas Teknik, Universitas Muhammadiyah Jember
* yamie.rez@gmail.com, **triawanac@unmuhjember.ac.id

ABSTRACT

Cryptography is a technique or method for securing data from other unauthorized parties. The substitution algorithm is the simplest algorithm and is classified as a classic in the field of cryptography, for example the Caesar cipher algorithm. ASCII code (American Standard Code for Information Interchange) is a code that contains characters that can be processed by a computer. By type, not all ASCII characters can be printed by the computer. Some characters are not printed or illegible as usual. These unreadable characters are called control characters. The control character can be used to improve the performance of the Caesar Cipher algorithm, because it focuses on processing text data. The application of control characters uses a simple method but has a complex solution. The results of this study are in the form of a new method with the Caesar cipher algorithm as a classical cryptographic method or technique and ASCII characters as the basis for the development of the ciphertext performance resulting from the encryption process. In testing this method, the success rate reaches 100% in securing the contents of the document with a sample of 500 letters. While the possibility of solving the ciphertext results is classified as difficult because the control characters of the ciphertext results that are illegible will make the decryption result multiple interpretations.

Keyword: Control Characters, ASCII, Caesar Cipher, Cryptography, Substitution Algorithm

1. Introduction

Kriptografi adalah salah satu teknik atau metode untuk mengamankan data dari pihak lain yang tidak berkepentingan atau yang tidak memiliki akses [1]. Terdapat dua proses utama pada kriptografi yaitu enkripsi dan dekripsi. Enkripsi adalah proses merubah data menjadi kode-kode rahasia, sedangkan dekripsi adalah kode-kode rahasia yang diproses kembali menjadi data atau informasi. Beberapa contoh algoritma kriptografi klasik diantaranya *Caesar cipher, Vigenere cipher*, *Playfair cipher* dan lain-lain [2]. *Caesar cipher* adalah salah satu contoh kriptografi klasik yang menggunakan teknik substitusi *monoalphabet*, dimana setiap karakter enkripsi menggantikan satu karakter tertentu [3]. Algoritma *Caesar cipher* merupakan salah satu jenis algoritma kriptografi klasik yang memiliki mode simetris.

American Standard Code for Information Interchange (ASCII) merupakan tabel karakter *printable* maupun *non-printable* [4]. Karakter ASCII ini nantinya akan terintegrasi dengan pesan yang dihasilkan dari proses enkripsi menggunakan algoritma *Caesar cipher*. ASCII adalah sebuah kode standar yang digunakan dalam pertukaran informasi pada komputer [5]. Jumlah karakter ASCII adalah sebanyak 255, dimana karakter ASCII urutan 0 sampai 127 merupakan karakter ASCII untuk manipulasi teks, karakter ASCII urutan 128 sampai 255 merupakan karakter ASCII untuk manipulasi grafik [6]. ASCII memiliki karakter kontrol yang dibedakan menjadi 5 kelompok sesuai penggunaan berturut-turut yaitu *Logical Communication, Device Control, Information Separator, Code Extension,* dan *Physical Communication* [7]. Karakter ASCII ini banyak dijumpai pada papan ketik komputer atau instrumen-instrumen digital.

Salah satu bagian *Unicode* pada ASCII adalah karakter *non-printable* kontrol ASCII atau sering disebut "ASCII Character Control". Karakter ini jarang dipergunakan secara umum, namun sering digunakan untuk kontrol pada komputer. Sehingga dalam pengimplementasiannya karakter ini merupakan karakter *non-printable* serta sangat cocok dan tepat untuk diintegrasikan dalam proses enkripsi data dengan algoritma *Caesar cipher*. Pada kenyataannya, algoritma *Caesar cipher* mudah dipecahkan dengan metode *exhaustive key search* karena jumlah kuncinya sangat sedikit (hanya ada 26 kunci) [8]. Namun dengan adanya karakter ASCII *non-printable* yang diterapkan, tingkat kesulitan dalam proses dekodenya akan lebih rumit. Dikatakan lebih rumit karena karakter *non-printable* ini akan memperoleh *ciphertext* yang memiliki makna yang relatif sulit dipahami (bersifat multitafsir) dan menghasilkan karakter acak berdasarkan dari karakter *non-printable* yang

diintegrasikan. Penelitian terkait perpaduan data ASCII pernah dilakukan oleh M. Fadlan, et al. yang membahas terkait "Pengamanan Data Teks melalui Perpaduan Algoritma Beaufort dan Caesar Cipher" [9]. Hasil penelitian tersebut menerapkan perpaduan algoritma beaufort dengan caesar cipher dengan memberikan karakter penentuan sebanyak 94 yang dapat digunakan dalam melakukan proses enkripsi dan dekripsi data teks. Karakter sebanyak 94 terdiri dari huruf L – Z (14 karakter huruf) dan sisanya karakter berupa simbol. Sedangkan pada penelitian ini, fokus karakter yang digunakan adalah karakter yang bersifat *non printable*, karena diharapkan hasil enkripsinya akan lebih sulit untuk dipecahkan.

2. Research Method

Tahapan penelitian digambarkan dalam bagan berikut ini :

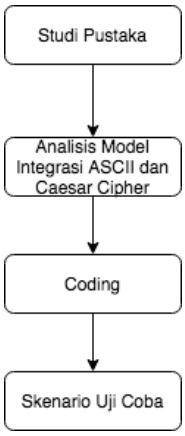

Figure 1. Tahapan Penelitian

a. Studi Pustaka

Studi pustaka dilakukan untuk mencari literatur terhadap bahan-bahan materi yang dibutuhkan yang berhubungan dengan topik yang diambil sebagai dasar pembahasan, serta untuk memperoleh landasan-landasan teori untuk menggali lebih jauh lagi tentang metodologi dari Kriptografi Substitusi. Adapun materi yang diperoleh adalah diktat materi kriptografi, artikel dari jurnal penelitian yang membahas kriptografi klasik dan model pengayaannya, dan materi terkait pengujian/evaluasi kriptografi klasik.

b. Analisis Model Integrasi ASCII dan Caesar Cipher

Analisis model integrasi ASCII dan Caesar Cipher merupakan analisa yang dilakukan tentang metode yang dipakai untuk memodifikasi bentuk luaran ciphertext yang disusun dengan cara substitusi. Tahap ini disusun untuk memastikan karakter kontrol yang diintegrasikan dengan *ciphertext* hasil dari enkripsi dengan *Caesar Cipher* dapat berjalan dengan baik.

c. Pembuatan Aplikasi (*coding*)

Pembuatan aplikasi uji meliputi perancangan sistem serta desain dan *coding* aplikasi

d. Skenario Uji

Skenario Uji yaitu pengujian yang dilakukan untuk menguji hasil penelitian. Skenario uji dilakukan dengan cara mengenkripsi *file* data berupa teks dengan menggunakan aplikasi yang telah dibuat.

2.1 Proses Enkripsi

Proses enkripsi dengan mengombinasikan antara data teks dengan karakter ASCII *non printable* terdiri dari beberapa tahap seperti berikut:

1. Setelah memasukkan kata atau kalimat yang berbentuk *string*, selanjutnya akan melalui proses *split* yang berfungsi mengubah bentuk *string* menjadi deret *array*.
2. Menggunakan proses *looping* untuk menghitung dan memproses setiap indeks *array*. Pada proses ini pengulangan terjadi sampai seluruh indeks pada *array* tersimpan dalam deret *array* baru.
3. Proses enkripsi pertama dikhususkan untuk mendapatkan karakter *cipher* pertama dari indeks karakter asli yang pertama dengan dasar penghitungan nilai desimal dari karakter indeks dibagi 30 karena karakter kontrol ASCII terbatas pada nilai desimal 0 – 30. Namun NULL dengan nilai desimal 0 tidak digunakan dalam proses ini. Setelah mendapatkan nilai bagi hasilnya akan dilakukan proses pembulatan nilai ke bawah. Nilai yang didapatkan akan disimpan dalam *array* baru.
4. Proses Enkripsi kedua dikhususkan untuk mendapatkan karakter *cipher* kedua dan sekaligus menjadi *key* untuk karakter *cipher* pertama. Cara mendapatkan karakter *cipher* kedua menggunakan penghitungan nilai desimal indeks *array* di *modulus* 30. Sisa bagi ditambahkan satu untuk

menghindari karakter kontrol netral atau *NULL*. Hasil yang didapatkan akan disimpan pada indeks selanjutnya dan pada *array* yang sama dengan proses enkripsi pertama.

5. Proses enkripsi yang menghasilkan dua karakter *cipher* tersebut dilakukan pada satu karakter teks asli atau indeks pertama dari deret *array* yang didapat pada proses *split*.
6. Proses tersebut akan berulang seperti yang dijelaskan pada poin 2.
7. Setelah terbentuk deret *array* baru, maka sistem akan melakukan *join* untuk menjadikannya *string* kembali dan dicetak.

Berikut ini merupakan diagram alur proses enkripsi secara lebih rinci:

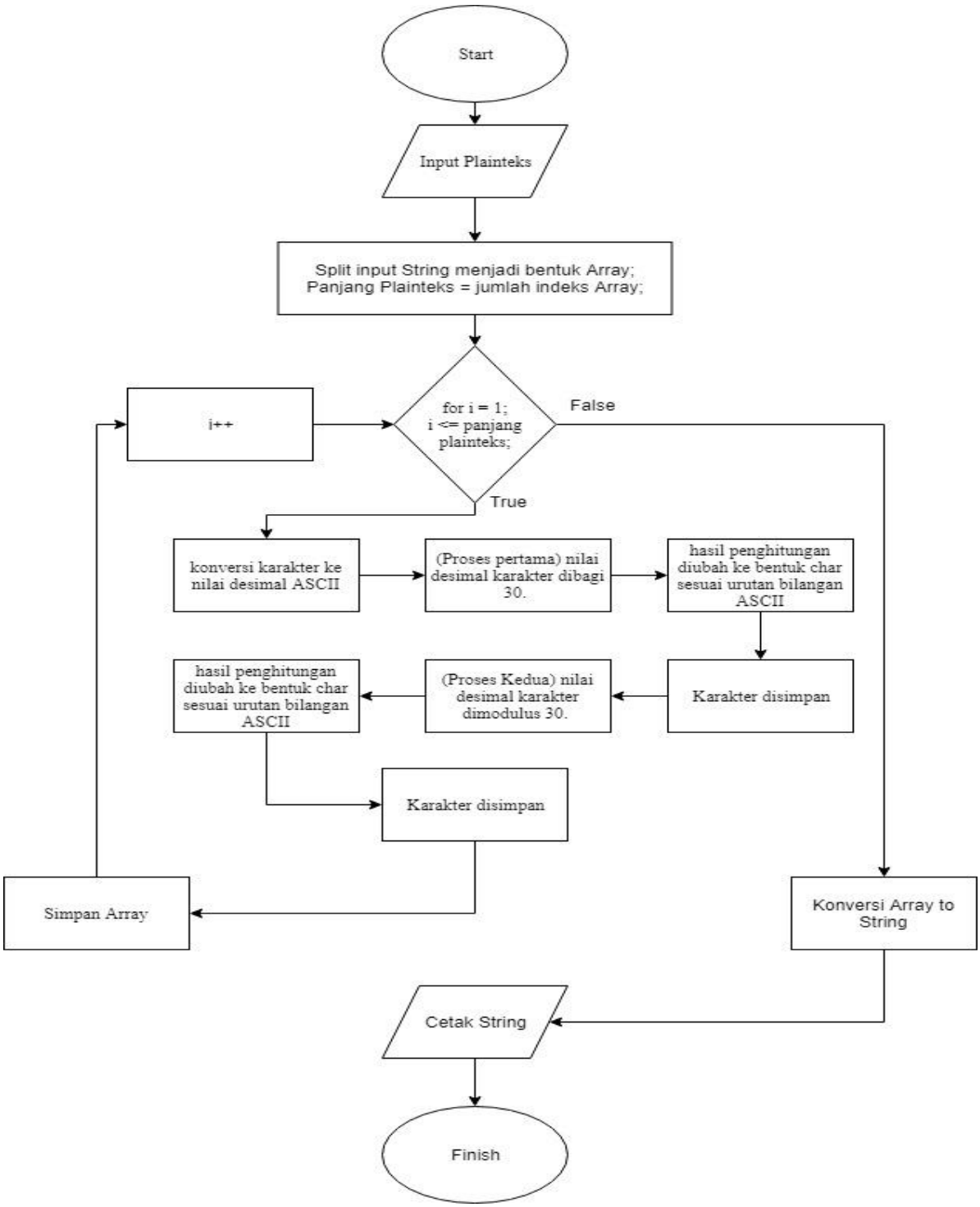

Figure 2. Diagram Proses Enkripsi

2.2 Uji Proses Enkripsi

Uji proses akan menggunakan kata **AKU** setiap kata akan di pisahkan menjadi deret *array*, dan menggunakan penghitungan rumus Enkripsi yang dimodifikasi dengan perpaduan antara karakter huruf (26 huruf) dengan karakter ASCII *non printable* (4 huruf) sebagai berikut [8] :

Untuk proses pertama menggunakan

$$\text{Enc1} = \text{char}(\frac{\text{ASCII (Plaintext)}}{30}) \quad \dots (1)$$

Untuk proses kedua menggunakan

$$Enc2 = char((ASCII(Plaintext)\ mod\ 30) + 1) \quad \dots (2)$$

Proses enkripsi secara manual dalam bentuk matematis, dijelaskan pada tahapan berikut ini:

Iterasi I "**AKU**"

- Deret *Array* ke-1 = A;
- Konversi nilai desimal; A = 65
- $\boldsymbol{Enc1 = char(\frac{65}{30})}$
- $\boldsymbol{Enc1 = char(2,1)}$**;** *floor* proses
- $\boldsymbol{Enc1 = char\ (2)}$
- $\boldsymbol{Enc1 = STX}$
- $\boldsymbol{Enc2 = char((65\ mod\ 30) + 1)}$
- $\boldsymbol{Enc2 = char(5 + 1)}$
- $\boldsymbol{Enc2 = char\ (6)}$
- $\boldsymbol{Enc2 = ACK}$

Hasil dari enkripsi huruf **A** pada iterasi pertama adalah : *STX ACK*

Iterasi II "**AKU**"

- Deret *Array* ke-2 = K
- Konversi nilai desimal; K = 75
- $\boldsymbol{Enc1 = char(\frac{75}{30})}$
- $\boldsymbol{Enc1 = \ char(2,5)}$**;** *floor* proses
- $\boldsymbol{Enc1 = char\ (2)}$
- $\boldsymbol{Enc1 = STX}$
- $\boldsymbol{Enc2 = char((75\ mod\ 30) + 1)}$
- $\boldsymbol{Enc2 = char\ (15 + 1)}$
- $\boldsymbol{Enc2 = char\ (16)}$
- $\boldsymbol{Enc2 = DLE}$

Hasil dari enkripsi huruf **K** pada iterasi pertama adalah : *STX DLE*

Iterasi III "**AKU**"

- Deret *Array* ke-3 = U
- Konversi nilai desimal; U = 85
- $\boldsymbol{Enc1 = char(\frac{85}{30})}$
- $\boldsymbol{Enc1 = char(2,8)}$**;** *floor* proses
- $\boldsymbol{Enc1 = char(2)}$
- $\boldsymbol{Enc1 = STX}$
- $\boldsymbol{Enc2 = char((85\ mod\ 30) + 1)}$
- $\boldsymbol{Enc2 = char\ (25 + 1)}$
- $\boldsymbol{Enc2 = char\ (26)}$
- $\boldsymbol{Enc\ 2 = SUB}$

Hasil dari enkripsi huruf **U** pada iterasi pertama adalah : *STX SUB*
Hasil dari tiga iterasi tersebut menghasilkan deret *array* baru yang berupa hasil enkripsi dari kata **AKU**. Sehingga dihasilkan hasil enkripsi sebagai berikut:

AKU = *STX ACK STX DLE STX SUB*

2.3 Proses Dekripsi

Setelah data *ciphertext* terbentuk, maka agar data *ciphertext* dapat dipahami oleh pihak yang menerima data, harus dilakukan proses dekripsi. Proses dekripsi yang dilakukan adalah sebagai berikut:

1. Setelah *ciphertext* berbentuk *string* dimasukkan, dilakukan proses *split* untuk diubah menjadi bentuk *array*.
2. Menggunakan proses looping untuk menghitung dan memproses setiap indeks *array*. Pada proses ini pengulangan terjadi sampai seluruh indeks pada *array* tersimpan dalam deret *array* baru.
3. Proses dekripsi untuk mendapatkan satu karakter asli melalui dua tahap. Tahap pertama untuk menentukan kondisi ganjil genap, sehingga pada proses ini diperlukan dua indeks *array* agar kondisi ganjil genap terpenuhi. Tahap kedua adalah proses penghitungan. Nilai yang didapatkan pada tahap pertama baik dari kondisi ganjil maupun genap akan dijumlahkan sesuai rumus penghitungan. Hasil dari penghitungan akan disimpan ke bentuk *array* baru.
4. Proses tersebut akan berulang seperti yang dijelaskan pada poin 2.
5. Setelah terbentuk deret *array* baru, maka sistem akan melakukan *join* untuk menjadikannya *string* kembali dan dicetak.

Berikut ini merupakan diagram alur proses dekripsi:

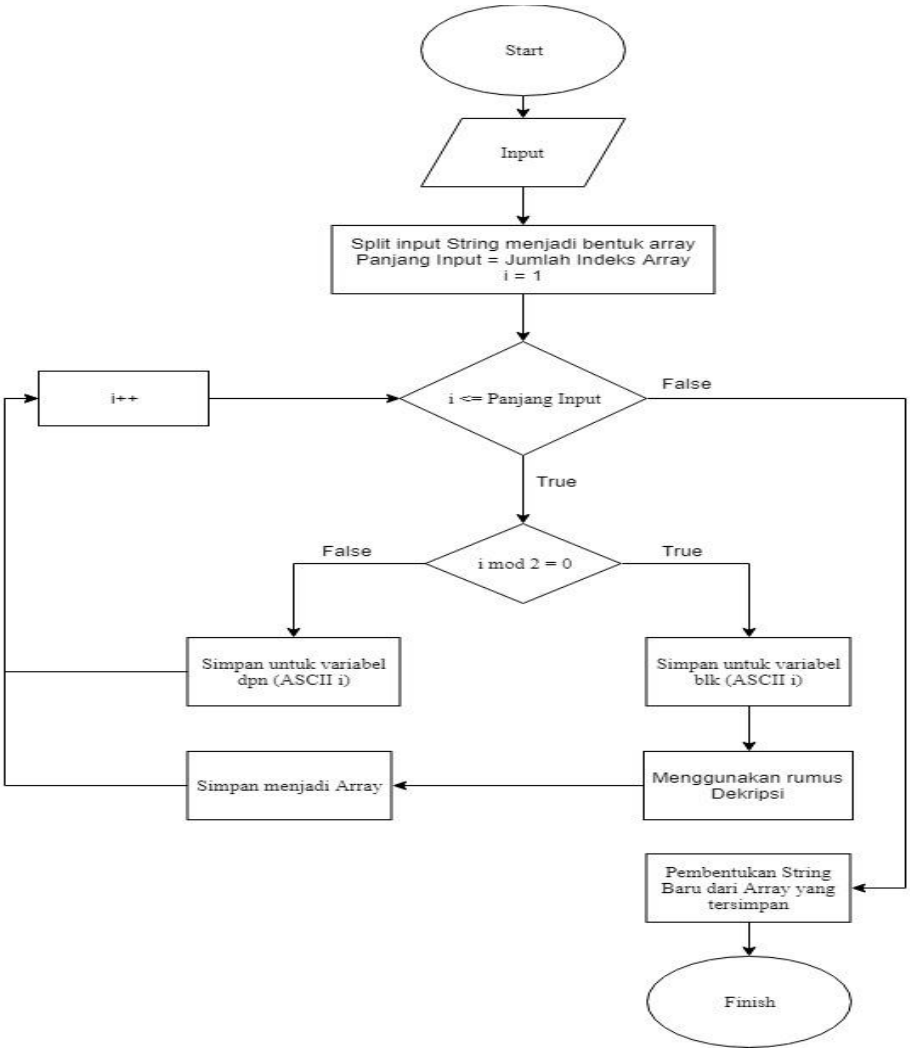

Figure 3. Diagram Proses Dekripsi

2.4 Uji Proses Dekripsi

Uji proses akan menggunakan kata **"STX ACK STX DLE STX SUB"** setiap karakter ASCII akan ditentukan kondisi terpenuhi dan dihitung dengan rumus penghitungan sebagai berikut [8]:

$$Dec = char(\ ASCII(Indeks\ Ganjil) * 30) + (\ ASCII(Indeks\ Genap) - 1) \quad \ldots (3)$$

Kondisi terpenuhi apabila terdapat dua nilai yang dibutuhkan, yakni nilai indeks ganjil dan nilai indeks genap. Untuk mendapatkan kondisi terpenuhi dibutuhkan dua iterasi yang berurutan. Dua karakter *ciphertext* yang berurutan akan membentuk satu karakter *plaintext*.

Penjelasan manual terkait proses dekripsi adalah sebagai berikut:

Iterasi I **"STX ACK STX DLE STX SUB"**

- Indeks *ciphertext* ke-1; i = 1
- Konversi nilai desimal; STX = 2
- Menentukan kondisi ganjil genap, i *mod* 2 = 1; *false*
- Mendapatkan kondisi iterasi ganjil;

Iterasi II **"STX ACK STX DLE STX SUB"**

- Deret *ciphertext* ke-2; i = 2
- Konversi nilai desimal; ACK = 6
- Menentukan kondisi ganjil genap, i *mod* 2 = 0; *true*
- Mendapatkan kondisi iterasi genap;

Kondisi terpenuhi dari iterasi 1 dan 2.

- $\boldsymbol{Dec = (\ 2 * 30) + (\ 6 - 1)}$
- $\boldsymbol{Dec = (\ 60) + (\ 5)}$
- $\boldsymbol{Dec = 65}$
- **Konversi nilai ke karakter ; 65 = A**

Hasil dari 2 iterasi dengan kondisi ganjil genap membentuk 1 karakter yakni **A**.

Iterasi III **"STX ACK STX DLE STX SUB"**

- Deret *ciphertext* ke-3; i = 3
- Konversi nilai desimal; STX = 2
- Menentukan kondisi ganjil genap, i *mod* 2 = 1; *false*
- Mendapatkan kondisi iterasi ganjil;

Iterasi IV **"**STX **ACK STX DLE STX SUB"**

- Deret *ciphertext* ke-4; i = 4
- Konversi nilai desimal; DLE = 16
- Menentukan kondisi ganjil genap, i *mod* 2 = 0; *true*
- Mendapatkan kondisi iterasi genap;

Kondisi terpenuhi, mendapatkan nilai depan dan belakang dari iterasi 3 dan 4.

- $Dec = (\ 2 * 30) + (\ 16 - 1)$
- $Dec = (\ 60) + (\ 15)$
- $Dec = 75$
- Konversi nilai ke karakter ; 75 = K

Hasil dari 2 iterasi dengan kondisi ganjil genap membentuk 1 karakter yakni **K**

Iterasi V **"STX** ACK **STX DLE STX SUB"**

- Deret *ciphertext* ke-5; i = 5
- Konversi nilai desimal; STX = 2
- Menentukan kondisi ganjil genap, i *mod* 2 = 0; *false*
- Mendapatkan kondisi iterasi ganjil;

Iterasi VI **"**STX **ACK STX DLE STX SUB"**

- Deret *ciphertext* ke-6; i = 6
- Konversi nilai desimal; SUB = 26
- Menentukan kondisi ganjil genap, i *mod* 2 = 0; *true*
- Mendapatkan kondisi iterasi genap;

Kondisi terpenuhi, mendapatkan nilai depan dan belakang dari iterasi 5 dan 6.

- $Dec = (2 * 30) + (26 - 1)$
- $Dec = (60) + (25)$
- $Dec = 85$
- Konversi nilai ke karakter ; 85 = U

Hasil dari 2 iterasi dengan kondisi ganjil genap membentuk 1 karakter yakni **U**

Hasil dari 6 iterasi tersebut menghasilkan deret *array* baru yang berupa hasil enkripsi dari deret **"*STX ACK STX DLE STX SUB*"**. Sehingga dihasilkan hasil enkripsi sebagai berikut
***STX ACK STX DLE STX SUB* = AKU**

3. Result and Analysis

3.1 Pengujian dengan Tools secara Daring

Pengujian menggunakan *tools decoding online* yang bisa di akses secara umum. *Tools* dapat diakses pada url: www.dcode.fr. Ada 20 berkas yang akan diujicoba dengan jumlah kata dan kapasitas yang bervariatif.

Pengujian Enkripsi dan *Ciphertext*, metode yang dipakai untuk pengujian menggunakan metode *bruteforce*.

Table 1. Pengujian Sampel Teks dan *Ciphertext*

No	File Sampel	Jumlah Kata (kapasitas file)	Keberhasilan enkripsi dan dekripsi	Pengujian encoding ciphertext menggunakan tools online
1	File 1	458 (15 KB)	Berhasil	Tidak Terbaca
2	File 2	411 (15 KB)	Berhasil	Tidak Terbaca
3	File 3	409 (16 KB)	Berhasil	Tidak Terbaca
4	File 4	372 (15 KB)	Berhasil	Tidak Terbaca
5	File 5	265 (14 KB)	Berhasil	Tidak Terbaca
6	File 6	332 (16 KB)	Berhasil	Tidak Terbaca
7	File 7	504 (16 KB)	Berhasil	Tidak Terbaca
8	File 8	448 (16 KB)	Berhasil	Tidak Terbaca
9	File 9	737 (17 KB)	Berhasil	Tidak Terbaca
10	File 10	3 (13 KB)	Berhasil	Tidak Terbaca
11	File 11	6 (13 KB)	Berhasil	Tidak Terbaca
12	File 12	15 (13 KB)	Berhasil	Tidak Terbaca
13	File 13	666 (16 KB)	Berhasil	Tidak Terbaca
14	File 14	86 (14 KB)	Berhasil	Tidak Terbaca
15	File 15	237 (15 KB)	Berhasil	Tidak Terbaca
16	File 16	143 (14 KB)	Berhasil	Tidak Terbaca
17	File 17	279 (16 KB)	Berhasil	Tidak Terbaca
18	File 18	163 (15 KB)	Berhasil	Tidak Terbaca
19	File 19	224 (15 KB)	Berhasil	Tidak Terbaca
20	File 20	407 (16 KB)	Berhasil	Tidak Terbaca

Berdasarkan hasil pengujian pada tabel di atas, dapat dilihat tidak ada proses enkripsi maupun dekripsi yang mengalami kegagalan. Sehingga tingkat kesuksesan dalam proses enkripsi dan dekripsi mencapai 100%. Dari tabel di atas, diketahui bahwa *ciphertext* yang dihasilkan tidak dapat dipecahkan (data *ciphertext* tidak memiliki makna) menggunakan *tools online*.

3.2 Maping Substitute Character

Dari proses enkripsi dapat dipetakan untuk masing-masing karakter yang sering digunakan dalam penulisan dokumen. Pemetaan belum termasuk karakter – karakter unik lainnya.

Table 2. Contoh *Maping* Karakter ke bentuk ASCII

Karakter	ASCII	Karakter	ASCII	Karakter	ASCII	Karakter	ASCII	Karakter	ASCII
!	SOH EOT	4	SOH ETB	G	STX FF	Z	ETX SOH	m	ETX DC4
“	SOH ENO	5	SOH CAN	H	STX (CR)	[	ETX STX	n	ETX NAK
#	SOH ACK	6	SOH EM	I	STX SO	\	ETX ETX	o	ETX SYN
$	SOH BEL	7	SOH SUB	J	STX SI	]	ETX EOT	p	ETX ETB
%	SOH BS	8	SOH ESC	K	STX DLE	^	ETX ENO	q	ETX CAN
&	SOH (TAB)	9	SOH FS	L	STX DC1	_	ETX ACK	r	ETX EM
‘	SOH (LF)	:	SOH GS	M	STX DC2	`	ETX BEL	s	ETX SUB
(	SOH VT	;	SOH RS	N	STX DC3	a	ETX BS	t	ETX ESC
)	SOH FF	<	STX SOH	O	STX DC4	b	ETX (TAB)	u	ETX FS
*	SOH (CR)	=	STX STX	P	STX NAK	c	ETX (LF)	v	ETX GS
+	SOH SO	>	STX ETX	Q	STX SYN	d	ETX VT	w	ETX RS
,	SOH SI	?	STX EOT	R	STX ETB	e	ETX FF	x	EOT SOH
-	SOH DLE	@	STX ENO	S	STX CAN	f	ETX (CR)	y	EOT STX
.	SOH DC1	A	STX ACK	T	STX EM	g	ETX SO	z	EOT ETX
/	SOH DC2	B	STX BEL	U	STX SUB	h	ETX SI	{	EOT EOT
0	SOH DC3	C	STX BS	V	STX ESC	i	ETX DLE	|	EOT ENO
1	SOH DC4	D	STX (TAB)	W	STX FS	j	ETX DC1	}	EOT ACK
2	SOH NAK	E	STX (LF)	X	STX GS	k	ETX DC2	~	EOT BEL
3	SOH SYN	F	STX VT	Y	STX RS	l	ETX DC3	(spasi)	SOH SYN

4. Conclusion

Berdasarkan paparan hasil penelitian dan pembahasan yang telah diuraikan, dapat diambil kesimpulan, dengan penjelasan sebagai berikut:

1. Persentase keberhasilan proses enkripsi dan dekripsi mencapai 100% dengan jumlah karakter (*printable* dan *non printable*) yang berbeda – beda. Tingkat kesulitan *ciphertext* dalam hal pengujian serangan dapat dikatakan cukup sulit, hal ini terjadi karena sulitnya untuk menafsirkan hasil *ciphertext* khususnya terhadap karakter ASCII yang *non-printable* yang dibuktikan dari hasil percobaan sebanyak 20 kali dan seluruh *ciphertext*-nya sulit untuk dibaca
2. Perpaduan antara kode khusus ASCII yang bersifat modern menjadi penambahan keamanan yang signifikan dan menghasilkan dua *ciphertext* untuk tiap karakter *plaintext.* Hal ini dapat menambah keautentikan data teks.
3. Masing-masing karakter *ciphertext* saling berkaitan dan membentuk keutuhan data yang tidak bisa dideskripsikan bila kehilangan satu bagian. Berdasarkan hal tersebut, maka integrasi karakter ASCII *non printable* ini terbukti mampu diterapkan untuk meningkatkan kelemahan Algoritma *Caesar Cipher* berdasarkan jumlah kunci (sejumlah 26) dalam mengamankan data teks.

References

[1] D. C. Dermawan and T. A. Cahyanto, “Aplikasi Kirim Pesan Berbasis Jaringan Lokal Dengan Menerapkan Algoritma RSA Sebagai Teknik Dalam Menjaga Kerahasiaan Pesan,” pp. 1–8, 2018.

[2] I. R. Munir, “Kriptografi Klasik Bagian 1,” 2020. [Online]. Available: http://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2020-2021/kripto20-21.htm. [Accessed: 27-Nov-2020].

[3] D. Seftyanto, M. Apriani, and T. Haryanto, “PERAN ALGORITMA CAESAR CIPHER DALAM MEMBANGUN KARAKTER AKAN KESADARAN KEAMANAN INFORMASI,” in *Seminar Nasional Matematika dan Pendidikan Matematika FMIPA UNY*, 2012.

[4] R. Manggala Budiasa, “Makalah IF3058 Analisis Kriptografi dalam penentuan Cipherteks kode ASCII melalui metode Aljabar Boolean,” 2010.

[5] K. Hernawati, “Implementasi Cipher Viginere pada kode ASCII dengan Memanfaatkan Digit Desimal Bilangan Euler.”

[6] G. Yadav, “ASCII Code Table - Printable, Non-Printable & Extended PDF,” 2019. [Online]. Available: https://tutorialsbookmarks.com/ascii-code-table/. [Accessed: 27-Nov-2020].

[7] “The ASCII Character Set.”

[8] A. Jain, R. Dedhia, and A. Patil, “Enhancing the Security of Caesar Cipher Substitution Method using a Randomized Approach for more Secure Communication,” 2015.

[9] M. Fadlan, S. Sinawati, A. Indriani, and E. D. Bintari, “Pengamanan Data Teks Melalui Perpaduan Algoritma Beaufort Dan Caesar Cipher,” *J. Tek. Inform.*, vol. 12, no. 2, pp. 149–158, 2019.