Studi dan Perbandingan Kriptanalisis antara *Vigenère Cipher* dengan *"Eager Evil Musician" Cipher*

Kaisar Siregar - 13506072

Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung
Jl. Ganeca No. 10, Bandung, Jawa Barat
e-mail: kaisar.siregar@gmail.com

ABSTRACT

***Vigenère Cipher* adalah salah satu metode enkripsi berjenis kunci simetrik dengan menggunakan metode subtitusi polialfabetik. Metode yang digunakan dalam *Vigenère Cipher* adalah dengan mengenkripsi sebuah teks alfabetis menggunakan *Caesar Cipher* yang berbeda sesuai dengan huruf pada kata kunci yang berpadanan dengan plainteks itu sendiri.**

Untuk membuat unbreakable chiper (chiper yangvtidak dapat dipecahkan), hal yang harus dilakukan adalah menggunakan kunci yang benar-benar acak dan panjang kunci sama dengan panjang plainteks. Salah satu algoritma kriptografi yang tidak dapat dipecahkan adalah one-time pad, yang menggunakan deretan krakter kunci yang dibangkitkan secara acak. Sayangnya, algoritma ini tidak efisien karena bermasalah saat menyimpan dan mendistribusikan kunci yang sangat panjang.

Dengan memanfaatkan sifat algoritma untuk unbreakable chiper, seperti pada one-time pad, Vigènere chiper bisa dikembangkan menjadi lebih susah untuk dipecahkan. Caranya adalah dengan membangkitkan deretan karakter kunci *pseudorandom* sepanjang plainteks dengan memanfaatkan sifat bilangan Ulam terhadap kunci masukan dari pengguna. Selain itu untuk mempersulit proses kriptanalisis, maka basis dari tabel enkripsi yang digunakan adalah tabel extended ASCII yang berukuran 256x256, bukan tabel alfabetis (*Tabula Recta*) biasa yang berukuran 26x26. Algoritma enkripsi ini penulis sebut sebagai *"Eager Evil Musician" Cipher* (yang merupakan anagram dari *ASCII Ulam Vigenère Cipher*).

Kata Kunci: *Vigenere Cipher*, Deret Ulam, ASCII

1. PENDAHULUAN

Kriptografi, yang berasal dari bahas Yunani, κρύπτω (*krýpto)* yang berarti tersembunyi dan γράφω (*gráfo)* menulis, adalah seni dan ilmu dalam penyembunyian informasi. Kriptografi dewasa ini dianggap sebagai bidang matematika dan juga sains komputer.

Dalam pandangan global, kriptografi dapat dinyatakan dalam dalam dua buah proses, yaitu enkripsi dan dekripsi. Emkripsi adalah proses mengubah plainteks biasa menjadi sebuah dokumen yang tidak memiliki makna (atau dikenal dengan *ciphertext*). Di lain pihak, dekripsi adalah sebuah proses yang simetri dengan enkripsi di mana dekripsi akan mengubah *ciphertext* menjadi sebuah plainteks asalanya. Pasangan proses enkripsi dan dekripsi dituangkan ke dalam sebuah algoritma yang disebut sebagai *cipher*.

Dalam notasi matematika, sebuah proses enkripsi dapat dinotasikan sebagai:

$$E(P) = C \qquad (1.1)$$

Di mana fungsi enkripsi E memetakan plainteks P ke *ciphertext* C. Sementara proses dekripsi dapat dinotasikan sebagai berikut:

$$D(C) = P \qquad (1.2)$$

Di mana fungsi dekripsi D memetakan *ciphertext* P ke plainteks awalnya Adapun fungsi *cipher* yang merupakan komposisi antara enkripsi dan dekripsi dapat ditulis sebagai berikut:

$$D(E(P)) = P \qquad (1.3)$$

Terlihat bahwa proses dekripsi yang dilakukan pada sebuah *ciphertext* hasil enkripsi akan menghasilkan plainteks awal.

Selain itu bidang kriptografi terbagi dalam beberapa jenis dan juga area pembelajaran. Salah satunya adalah bidang kriptografi polialfabetis. Seperti namanya, kriptografi polialfabetis menggunakan *cipher*polialfabetis dalam prosesnya. Sebuah *cipher* poialfabetis didasari oleh proses subtitusi menggunakan beberapa alfabet subtitusi yang berbeda. *Cipher* polialfabetis kemudian berkembang menjadi sebuah *cipher* baru yang dikenal dengan *cipher*

polialfabetis kunci progresif yang diciptakan oleh Johannes Trithemius. *Cipher* ini mensubtitusi alphabet dalam interval yang acak untuk tiap hurufnya. Kunci progresif paling klasik dikenal dengan nama *tabula recta*.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Gambar 1.1: *Tabula Recta* 26 Alfabet

Tabua recta ini digunakan di banyak *cipher*.Salah satunya adalah *Vigenère cipher*.

2. *VIGENÈRE CIPHER*

Vigenère Cipher adalah salah satu metode enkripsi berjenis kunci simetrik dengan menggunakan metode subtitusi polialfabetik. Metode yang digunakan dalam *Vigenère Cipher* adalah dengan mengenkripsi sebuah teks alfabetis menggunakan *Caesar Cipher* yang berbeda sesuai dengan huruf pada kata kunci yang berpadanan dengan plainteks itu sendiri. Di mana jauh pergeseran huruf plainteks ditentukan oleh nilai desimal dari huruf kunci tersebut (A = 0, B = 1, C = 3 ... Z = 25)

Secara aljabar, *Vigenère Cipher* dapat ditulis sebagai berikut:

$$Ci = (Pi + Ki) \bmod 26 \qquad (2.1)$$

di mana
P_i : karakter plainteks
K_i : karakter kunci
C_i : karakter chiperteks

Sedangkan untuk proses dekripsinya dapat ditulis sebagai berikut:

$$Pi = (Ci - Ki) \bmod 26 \qquad (2.2)$$

Pada Vigènere *chiper*, jika panjang kunci lebih pendek daripada panjang plainteks, maka kunci tersebut akan diulang penggunaannya.

Contoh penggunaan Vigènere *chiper*:
P: AKUANAKSEHATTUBUHKUKUAT
K: SAYANGMAMASAYANGMAMASAY
C: SKSAAGWSQHSTRUOATKGKMAR

Terlihat bahwa huruf huruf A dienkripsikan menjadi beberapa huruf yang berbeda, hal ini merupakan karakteristik utama dari *polialfabetic cipher*.

3. KRIPTANALISIS PADA *VIGENÈRE CIPHER*

Pada awalnya *Vigenère cipher* dianggap tidak dapat dipecahkan, namun seiring berkembangnya teknik dan metode kriptanalisis yang ada, kekuatan *Vigenère cipher* lambat laun berkurang. Bahkan metode seperti *Kasiski Examination* serta *Frequency Analysis* dapat memecahkan sebuah *Vigenère cipher* dengan relatif cepat.

3.1. *KASISKI EXAMINATION*

Kasiski Examination mengeksploitasi fakta bahwa beberapa kata umum pada suatu bahasa lokal, sperti *"the"* pada bahasa Inggris kemungkina akan dienkripsi oleh key yang sama. Contoh kasusnya adalah seperti ini:

P: **CRYPTO**ISSHORTFOR**CRYPTO**GRAPHY
K: ABCDABCDABCDABCDABCDABCDABCD
C: **CSASTP**KVSIQUTGQU**CSASTP**IUAQJB

Terlihat bahwa kata CRYPTO terenkripsi menjadi sekuens yang sama yaitu CSASTP. Maka secara intuitif kita dapat menyimpulkan bahwa kata CRYPTO secara kebetulan dienkripsi dengan sekuens kunci yang sama. Kasiski menyimpulkan bahwa jarak antara sekuens alfabet pada suatu *ciphertext* kemungkinan besar merupaan kelipatan dari panjang kunci yang digunakan. Pada kasus ini jarak antara CSASTP adalah 16. Maka kita bisa mengambil hipotesis bahwa panjag kuncinya antara 1, 2, 4, 8, dan 16 (terbukti bahwa panjang kunci adalah 4).

Jika panjang kunci telah diketahui, maka kunci dapat ditentukan dengan beberapa cara, antara lain:

1. *Exhaustive key search*, yakni dengan membangkitkan semua kemungkinan kunci. Jika panjang kunci adalah *p*, maka cara ini membutuhkan 26^p kali percobaan.

2. Mengelompokkan huruf-huruf pada chiperteks ke sejumlah *p* kelompok (*p* = panjang kunci). Lalu, melakukan teknik *Frequency Analysis* pada tiap-tiap kelompok tersebut.

3.1. *FREQUENCY ANALYSIS*

Ketika panjang kunci yang digunakan telah diketahui, *ciphertext* kemudian dapat dikelompokkan menjadi *n* kolom. Di mana *n* adalah panjang kunci yang dimaksud sehingga tiap kolom merepresentasikan alfabet yang dienkripsi dengan sebuah huruf pada kunci. Dengan memanfaatkan fakta-fakta yang berhubungan dengan frekuensi kemunculan huruf (seperti ‘e’ adalah huruf yang paling sering muncul pada bahasa Inggris), maka kita akan mendapat basis yang baik untuk menganalisis tiap huruf kunci yang ada.

4. KONSEP *ONE-TIME PAD*

Agar konsep *Vigenère Cipher* dapat menjadi sebuah *unbreakable cipher* (cipher yang tidak dapat dipecahkan kecuali dengan metode *brute-force*), maka dibutuhkan dua syarat berikut:

1. Kunci yang digunakan harus benar-benar acak
2. Panjang kunci harus sama dengan panjang plainteks

Kedua syarat inilah yang diimplementasikan dalam *one-time pad*. *One-time pad* berisi deretan karakter-karakter kunci yang dibangkitkan secara acak. Satu *pad* hanya digunakan sekali saja untuk mengenkripsi pesan, setelah itu *pad* yang telah digunakan dihancurkan supaya tidak dipakai kembali untuk mengenkripsi pesan yang lain.

Algoritma yang digunakan *One-time Pad* sama dengan Vigènere *chiper*, contohnya:

P: AKUANAKSEHATTUBUHKUKUAT
K: AWYDKANEUKLAOPQTBNSMDUI
C: AGSDXAXWYRLTHJRNIXMWXUB

Degan metode ini, akan sangat kecil kemungkinan yang ada (bahkan hampir tidak mungkin) akan terjadinya kasus di mana suatu sekuens pada plainteks dienkripsi dengan sekuens kunci yang sama, sehingga tertutup kemungkinan untuk *Kasiski Examintaion.* Hal inilah yang membuat *One-time Pad* tidak dapat dipecahkan.

Namun walaupun begitu, *One-time Pad* memiliki kelemahan yang amat besar, yaitu pada penyimpanan kunci. Untuk pesan yang amat panjang, maka kunci yang dipakai akan panjang pula. Pada aplikasi untuk data statis hal ini akan menimbulkan masalah pada penyimpanan kunci yang ada.

Sedangkan pada aplikasi kriptografi untuk komunikasi pesan, timbul masalah dalam pendistribusian kunci, yakni bagaimana kunci dapat dikirim secara aman, apakah melalui saluran komunikasi yang sama dengan saluran untuk pesan (hal ini tentu mengganggu *traffic* pesan yang padat dan memerlukan kunci yang berlapis untuk melindungi kunci yang dikirim), atau melalui saluran komunikasi kedua yang umumnya lambat dan mahal.

5. *EAGER EVIL MUSICIAN CIPHER*

Eager Evil Musician Cipher (yang merupakan anagram dari ASCII Ulam *Vigènere Cipher*) adalah suatu *cipher* yang dirancang penulis dengan memodifikasi *Vigenere Cipher* dengan menggunakan bilangan Ulam untuk membangkitkan kunci *pseudorandom* yang memiliki panjang kunci sama dengan plainteksnya, sehingga mengatasi permasalahan penyimpanan dan distribusi pada *one-time pad*. Selain itu pada *Eager Evil Musician Cipher* tabel yang digunakan bukanlah tabel alfabetis yang berukuran 26x26, namun tabel *extended* ASCII yang berukuran 256x256. Hal ini dimaksudkan agar mengurangi huruf yang berulang pada *ciphertext* sehingga akan menyulitkan *frequency analysis* dan juga pengecekan secara manual.

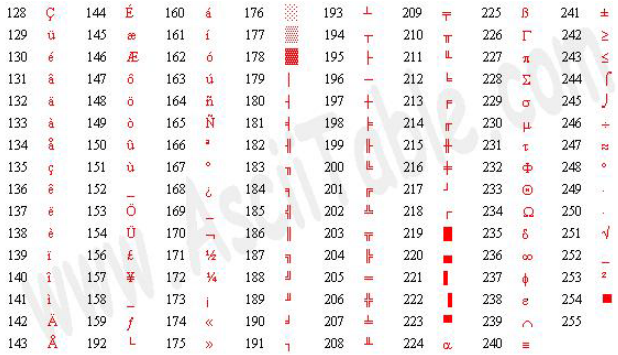
5.1. *ASCII*

American Standard Code for Information Interchange (ASCII) adalah *encoding* karakter yang didasari oleh alfabet bahasa Inggris. Kode ASCII merepresentasikan sebuah teks pada komputer, media komunikasi elektroik, dan alat-alat lain yang memiliki hubungan dengan teks.

ASCII terdiri dari definisi 128 karakter, 33 di antaranya merupakan karakter non-cetak yang biasanya merupakan karakter yang mengatur bagaimana suatu teks diproses. Selain 128 karakter ASCII, terdapat pula tambahan 128 karakter tambahan yang terdefinisi pada tabel *extended* ASCII. Kedua tabel inilah yang menjadi basis pada *Eager Evil Musician Cipher* sebagai pengganti *tabula recta* biasa.

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Gambar 5.1: *128 Karakter ASCII*



Gambar 5.1: *128 Karakter extended ASCII*

5.2. BILANGAN ULAM

Sebuah bilangan ulam adalalah sekumpulan bilangan bulat positif yang merupakan anggota dari deret Ulam yang dirancang oleh Stanislaw Ulam, seorang ahli matematika dari Polandia yang juga aktif dalam pembuatan bom atom pada *Manhattan Project*, yang dipublikasikan pada SIAM (*Society for Industrial and Applied Mathematics*) *Review* pada tahun 1964.

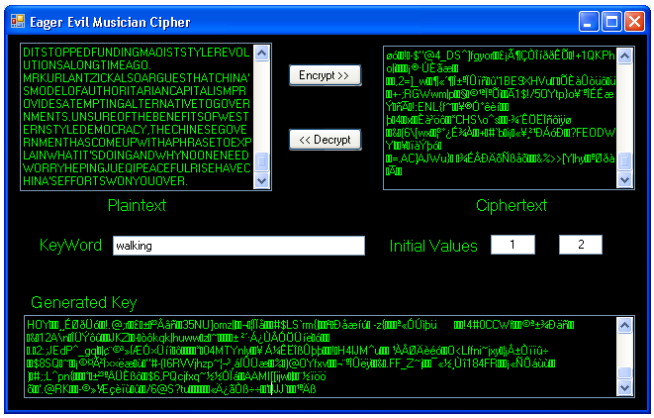
Deret Ulam (u, v) dimulai dengan U_1 = u dan U_2 =v, yang merupakan dua bilangan Ulam pertama di mana 0 < u < v. Kemudian untuk n > 2, U*n* dedifinisikan sebagai bilangan bulat positif terkecil yang merupakan penjumlahan dari dua buah bilangan ulam sebelumnya yang berbeda dan hanya dapat terdefinisi satu kali.

Menurut definisi 3= 1 + 2, merupakan bilangan Ulam, dan begitu pula 4 = 1 + 3 (4 = 2 + 2 tidak dianggap karena kedua bilangan penjumlahnya harus merupakan bilangan Ulam yang berbeda). Sementara, 5 bukan merupakan deret Ulam karena dapat didefinisikan lebih dari satu kali, yaitu 5 = 2 + 3 = 1 + 4.

Beberapa bilangan Ulam standar (Deret Ulam (1, 2)) yang pertama adalah 1, 2, 3, 4, 6, 8, 11, 13, 16, 18, 26, 28, 36, 38, 47, 48, 53, 57, 62, 69, 72, 77, 82, 87, 97, 99, … Sementara beberapa bilangan Ulam standar yang juga merupakan bilangan prima adalah 2, 3, 11, 13, 47, 53, 97, 131, 197, 241, 409, 431, 607, 673, 739, 751, 983, 991, 1103, 1433, 1489, ….

Schmerl dan Spiegel pada tahun 1994 membuktikan bahwa Deret Ulam (2, v) untuk *v* bilangan ganjil≥ 5 mempunyai tepat 2 buah bilangan Ulam genap. Sedangkan Cassaigne dan Finch pada tahun 1995 membuktikan bahwa Deret Ulam (4, v) untuk $5 \leq v \equiv 1$ (mod) 4 mempunyai tepat 3 bilangan genap.

Adapun algoritma untuk membangkitkan n bilangan Ulam pertama dari Deret Ulam (u, v) dapat dilihat pada fungsi generateUList rancangan penulis yang menggunakan bahasa C# pada bagian lampiran.



Gambar 5.2: *Tampilan program Eager Evil Musician Encryptor*

5.3. ALGORITMA *EAGER EVIL MUSICIAN CIPHER*

Secara garis besar, algoritma *Eager Evil Musician Cipher* terbagi menjadi dua, yaitu tahap pembangkitan kunci acak dan tahap enkripsi dan dekripsi itu sendiri.

5.3.1. PEMBANGKITAN KUNCI ACAK

Tahap awal pada *Eager Evil Musician Cipher* adalah tahap membangkitkan kunci acak berdasarkan kunci yang dimasukkan oleh pengguna dan dua nilai bilangan Ulam pertama. Anggap panjang plainteks adalah n, dan panjang kunci masukan pengguna adalah m, dan m < n. Maka kunci bangkitan yang dihasilkan dapat dinotasikan sebagai berikut:

$$K_i = \begin{cases} k_i, & ,i < m \\ (k_{(i \bmod m)} + l_{(i-m)}) \bmod 256, & i \geq m \end{cases} \qquad (5.1)$$

Di mana:

K = Kunci bangkitan
k = Kunci masukan pengguna
l = Deret Ulam (u, v) yang dibangkitkan dengan nilai u dan v sesuai dengan masukan pengguna

Dari notasi tersebut terlihat untuk karakter kunci bangkitan ke i, di mana i > m, didapatkan dari pergeseran kunci masukan penggna yang direpetisi sampai jumlahnya sama dengan panjang plainteks dengan bilangan Ulam yang bersesuaian. Mod 256 digunakan karena *Eager Evil Musician Cipher* menggunakan karakter *extended* ASCII,

bukan alfabet biasa. Hasil akhir dari pembangkitan kunci ini adalah sebuah kunci acak yang memiliki panjang sama dengan plainteks. Sehingga konsep *one-time pad* telah terpenuhi.

Contohnya adalah sebagai berikut untuk Deret Ulam (1, 2)

P: aku adalah seorang kapiten mempunyai pedang panjang kalau berjalan prok prok prok

k: Saya Anak Sehat

K: Tc|e&Iyn{2m□□□£□□²□e□»³Â□¶ËÒÓòÖë

õ»ð!Ý39<Q?OjTCrr□U□¤¬´É·ÇäÓ□Çþó

Terlihat bahwa kunci yang dibangkitkan mempunyai panjang yang sama dengan panjang plainteks (mungkin beberapa karakter tidak tercetak pada dokumen ini karena merupakan karakter non-cetak). Perlu diketahui karena pada *Eager Evil Musician Cipher* menggunakan tabel ASCII maka penggunaan huruf kapital, karakter spasi, dan simbol-simbol lain juga akan berpengaruh.

5.3.2. PROSES ENKRIPSI DAN DEKRIPSI

Proses enkripsi dan dekripsi pada *Eager Evil Musician Cipher* pada dasarnya sama dengan proses enkripsi dan dekripsi pada *Vigenère Cipher* biasa. Perbedaannya hanyalah pada *Eager Evil Musician Cipher* bilangan modulo yang digunakan adalah 256, bukan 26. Sehingga notasi enkripsi dan dekripsi-nya menjadi seperti berikut:

$$Ci = (Pi + Ki) \bmod 256 \qquad (5.2)$$

di mana
P_i : karakter plainteks
K_i : karakter kunci bangkitan
C_i : karakter chiperteks

Sedangkan untuk proses dekripsinya dapat ditulis sebagai berikut:

$$Pi = (Ci - Ki) \bmod 256 \qquad (5.3)$$

Contoh proses enkripsi pada *Eager Evil Musician Cipher* adalah sebagai berikut:

P: **CRYPTO**ISSHORTFOR**CRYPTO**GRAPHY
k: ABCD
C: □□ □□□□¥¥□-³º¯ÂÇºÎÛÚÞßÝîäöò

Terlihat bahwa tidak seperti pada *Vigenère Cipher* biasa yang akan menyebabkan sekuens CRYPTO dienkripsi menjadi sekuens yang sama yaitu CSAS (lihat bab 3.1 untuk lebih jelasnya), *Eager Evil Musician Cipher* akan mengenkripsi sekuens tersebut menjadi sekuens yang berbeda.

6. PERBANDINGAN KRIPTANALISIS *VIGENÈRE CIPHER* DENGAN *"EAGER EVIL MUSICIAN" CIPHER*

Pada bab ini akan dilakukan studi perbandingan kriptanalisis menggunakan *Vigenère Cipher* dan *Eager Evil Musician Cipher*. Plainteks yang digunakan adalah sama, yaitu plainteks hasil jawaban soal nomor 2, tugas kriptanalisis IF3058 tahun 2009, Teknik Informatika Institut Teknologi Bandung (dapat dilihat di lampiran).

Plainteks tersebut dienkripsi dengan metode *Vigenère Cipher* menggunakan kata "walking" dan menghasilkan *ciphertext* seperti berikut:

XEQYZ ROYAX OBBID IYKQG XEEOD WYKWR YKJVZ WBZEB
PNENP CMRZE QFOBG KPHPY VRZDI YQQEK IEXLM ERAAC
XQAME SERQF CDEYS BPUIE DDWPR ENVSV TCENP QTNYO
EDKBN LKRXK TEKYE ADQBT ETDZW YOPEE YUNQA SFBMG
NWTER MEOIO QIWHX CLLCA VYHOH OZGNW NERIG UBTSO
XRXOO YIWHX ACWSV XOJGH SBUZD EZDPR XZAJS ENYWT
LMPVT ASPXM JEAAC BMPKL TTYVU UOTPN JLZDE DDIGK
XRZKL PGOTP BKPZR AEDWN YPIYQ BVSAT SOKPZ REIOK
HZEVP CNNTJ EOKKE UOSER MEUKM HSBUZ DETBE VTAGW
KAFKO HPVLB APAEK JBAPW LSAGR AVPVU NQENR SBVSL
ODCQO RAFZB IAEKF FCBBM ATZEZ TRWSD OAYUS ECDPN
TPHPS ZFZDE JGWAZ DEAYT VZANP CAOG
P TWOJH ZZOYD KBTBU DOMGO MUPDB RCETS KTNIG
OQKUO OPIZX BUOOY PKZPI PVTCX YGJNT XOGUK PPXZH
YOILX IAJWR LLQPR WNREI TKYHL XVRRO RPZWE ZOAWC
WFGUT SKBGN ASEKB RXQNY OEFGC EYMGK OJHFK PNYLL
LXAGU OTLBB HVETD YEAOJ TPBVN ZEOYK TAKSS NRIAT
ALNRQ AGESO OBRXI IYOLG ULRZT MPZET DYEAO IAROI
AJLEC CXRIP IGOIF LWRLC QGIWN DYBUO OCZEV GXUST
WITKI AVOZF SWYMO QAZAR PCBRJ ENERM EKOUW DABLW
NZZQA OKNAY TYIKN OEKGK ZFZBB UKXBN GWERZ SPBDV
IATSO XBRHS FQORY PSERI GMHOM KTNZP IEELR YPOHK
ZQYYH TXINT ZRFCA VGWRP GWEYA TSKVG NAYHO ZRGUE
LBITU WLERW HMDCW OIERU TSOZR GNELR CTKJU XLMEU
BCLFM NZOTZ DPEUS AEKAV TCLPZ WYRSH TMPGG GEDLZ
BGZCZ XKYAO IZXAS XKMLV QZOPE OXCZH ARZPQ AZARG
SMJYW STWAH XAMLX GBLUO FGQYR XEVOM AZKTP
VTZKX UERWJ ZDEHY ZYJOE PCKUO JALXL UUSCS SVNYA
EDDPR CKRWN UNZPE CACVZ AAMSB NTWMP BQPGJ WCSBR
XFODR CNQQR WKVGF ECVQW RYPHC YCTNO OXOWS IDIYK
AVSWG PWIXO JGPPN BXPST XPVYX OZUKU GNMZP NRTOI
GOPBC YHTXI FYKFE ZWJKN IDDZN TOFZB UVTCT SOEBX
HDERM JXETP BTVYP SERMG NENRC BUGPC SSVNY ZOYOQ
AUNDP BBBCE NQBQR TZSLB WHTZT SOEBX HDTDA RTZEO
LWEJA ROSAC APEDG QGNET DXMVM DBZEZ FOPPW KGFGJ
ANDQI KLACD QAMHO MKTVT OTTDC GOKND CCPNW
SERMH TWNOD PRCKR WNBEG ZEZBO NTEZL DQBTE TOYMF
TPIYD MEBAN PEVVR WTPBI YRUIY YBUKN CZEVG XEEDS
VGKNN LVENX OAYNQ GYPOA ZMQLQ NOSVT SWOTC BFZUL
PBMIU HUESW AYWLZ XOGOI ELQWZ XGUCV IAZVI NUIYY
KACQC RYPHL DKUOJ ADWWQ KHOQK CGNKR TDIEO
WNNKX VZWLT CUCXK VTNMF GPEXZ BVTCA WDMET
WTTFM GUCOG OZASA NECCA YQRPY NGNAB PXMSO PSZPE
RYPEC XAGEH EOOUB INANI BUKYH TXMFK COGOZ ASANE

```
RIFIK MPEXJ OPHLZ PEGOE EYMKV HATXE UGPIE CLBOJ
GLXLJ NUNZY VRTAE OGWEX UHPZQ AMFUP AQCKW CPPCY
XESPR IIKYH TXIFK BFZBB FCKNJ YCBBA R
```

Gambar 6.1: *Ciphertext hasil Vigenère Cipher*

Sedangkan apabila plainteks yang sama di enkripsi dengan *Eager Evil Musician Cipher* yang menggunakan Deret Ulam (1, 2), hasilnya menjadi

```
»©¶¿Â¼¼È³ÌËÚâÑïÛðæñù!!+2>7=LWhUpw{w~□□□«□¥¯¾µÆÂØø
éåú#>(<JCQV[\akew□§□□´ÂÄÖááø
        3%@OdXwai□□□□§□ÈÆÑÝã-ü6&TTg□□¢£°°À¿³ÆÇÙ
Üë
:K7P$_fo□□ªÁÚÝÖðçî
"9&,TUPc□iz□□-¡¹¯´º¹Óõ──────────────────────────
ù
        $,"*H<Ygws}±□±ÀÆÊâéñÃý$66b!Y_]|□¢□¨À¹ðÞûêÞ¾Ê/2
□z□□□ØÆÕíù66[APegs|□□±©ÍÔÂÑÑôéçøþö-0CmJml□¦□¶°¿-ÍÂûV
[□£□ □¦¿¼³µÕãÞëQ2IDQ\\zM□□□ÚÖùõó
8J+TEVPP□w□□□¹ÃÜÍÚæÛâê.#5;YFPdnz□¦□ÔÚÙ %-
+<JQ□|v□°ÇÐíýèø
        70eU|d□□¢-Å¾ÈÎç×ñòö%0SYeoy□ª³¶½ÍÒÝëÏiü=6=@B}}
ÂÌ×ÕëÚù$4GNK`exI}ur□□¯®èÞÛü÷3<ISeO\p□□□□ªÊ,/@+QMi□□
-êÈç÷8H¾¿¹Å·ÉÃÉÐµÓÉÑàæîØóìø
þüÜÚ-"?8PB?\Zljyr□□□□□§£©©¹ÉµÎËíýü
-(:)O?ETIqRml□x□¢□«·ÄÝÎúí
-<(4T7og}Z□□□□□»·ÁÝÞä
"31HjX□□□¬½£»¼Â½ÑÅÛãû-
>.RDR]d{w®□ÒØÒäòõ.F.BWLtown□□¦¬□»£½ÈÀõØÿú-
%(7NRnjx□□°¥±ÔÏéæ
──────────────────────────────────────────
ìô,7QcKrjt□□¯□´ÂØiÜ
øEx□s¯ÇÊÖëø2!=EiQrhi□|¤-³ÌÊÝ§èàòçõúþ
──────────────────────────────────────────
$/#V\wm{□¢µ¸²²ÎÏÿ%*]Z□□©□□¹¹¼»ËÄßáîê6N>NNZbh□□□ªÕîÚúé
ÿ2+:5D\Thy□□□□ÇÁÀÕª □ìçî$(/O<[Eorq¡□ªÖÖæÿ
──────────────────────────4;FJ]o□o□ÅÏßÝĀßûôù)BE
_v□¨¡À¢¹ºÌÆåïè÷ê!k]hg{¨□□®½»ÓÕÍðÝð-,2^>z□¿È¾èÎ×ûõ1'D-
YMSlZrf□v□□¥ÃËÜêõõí41SZVrde□▯□□Ò«,$)@@XOj□□¶ÎâÚìE3»
¯Î¨ÅÃ¶À»ÙÊÌÜåñáüéþ÷
──────────────────────────────────────────
5+?:SMGck{^s▯□□□«□ ´°□w□´ÚêÍûÿ -
/?M8XT[d^s`ht}□□·¬ÆÌÏÝêýð
        -'B<QRnnl{v□□□¥□Í°ÙÍßý&*/AUn□□□·¿¯¾¿ÄÉÃÏìò
──────────────────────────?EIPGTTd□ ¢ÀÊØÐøë÷î
$'%8;HNTsp{y~□£ □±Ì¾ÄÀïøû
-,",:OjWyw□□¡¼¥ÉËâÜÌ
í&0IMS[f{□□□°·Çêêçö──────────────────────────ð;
□|□x□ÕÑÚàÃ-
3@TK`bxe□□□¯□ÌÍÎÙýéþõ&,EVZ□m□□□¼«»¹ËÏú3[n□□□□□²±»
ÐÍàÍ×û──────────────────────────────==MFTXR□j□□¤
Õ¼ïêó
        3<:M;X[`□x□□□¼¸ÒÄåÉèéïþ#"5DNPZ=ht□□§ÜÉßú5.7L
Wq□□□ÅÙÌ÷æúñ
+8bH□□ ¨¦Ã½Æ½ÕèÑúÚé#;RoVux□¡§¾¼½½×ÜÜò9;AC[n□ÍÐÅÔà
Þþó/:?JMh]ax□u|□□©§ãÑèûñò>2UKXaT[□v□¡ÁÎõ0&BOHgf□□□ã×
öå3H°Ã¬È´ÎÀÅÎÎÚÃãÕèôëòûù



-
%$9BCETHgde□r□□□□¨□³-«ººÐÉã÷ò
,#!?3^CR]\efxnq□□¢ ÂµÆÑÍîïïÉÔ*0PJOgn{h□▯□□ -¶ÙÇåå
+1<\^v□□µª³µÍº¹ÎÚîõú&0AD8WGUt□§□ÔÇßÚñûóû*,:QFVvl{o□□¦
□¨¸©Î¹ÔëÃÿ0#
```

```
.4NXso|n¤§¹ÍÈÈåÜÃðÿ9DMKze}□□¤-Ò¯éçì
──────────────────────────

──────────────────────────────────────────
        ý3~□w□□Çß²õó)BGR[n]r□□□¬½³ÉÕÊÍðóëþ÷
%'5[Zvw□□¥²¯¾È½¿
*"_a□h □ª¤·±¸ÏÀòÏ
>EDNCVXk□□¤□îßÜýò        Ã<-@B\@IVt|□□□½ÈÄÏÃôÐÞäôÃ
%$==ZX{gx□□¹×ïßÿ□íù
```

Gambar 6.2: *Ciphertext hasil Eager Evil Musician Cipher (karakter _____ menunjukkan karakter-karakter non cetak)*

Untuk proses kriptanalisis kedua *ciphertext* di atas penulis akan menggunakan bantuan program *CryptoHelper*.

6.1. KRIPTANALISIS PADA *VIGENÈRE CIPHER*

Seperti yang telah dijabarkan pada bab 3, *Vigenère Cipher* rentan terhadap *Kasiski Examination* hal ini dapat terlihat dengan analisis seperti berikut:

- *Tri-graph* ERM adalah *tri-graph* yang paling sering muncul, yaitu sebanyak 6 kali. Dengan dua kemuculan dengan selang terdekat adalah pada karakter ke 1262 dan 1276. Sehingga kemungkinan panjang kuncinya adalah 1, 2, 7, dan 14.
- Sekuens COGOZASANE muncul pada karakter ke 1652 dan 1715. Karena sekuens tersebut cukup panjang (10 karakter) aman diasumsikan bahwa sekuens tersebut terjadi karena dienkripsi dengansekuens yang sama. Sehingga kemungkinan panjang kuncinya adalah 1, 3, 7, 9 dan 21.
- Dari kedua kemungkinanpanjang kunci tersebut terlihat bahwa 7 merupakan panjang kunci sebenarnya karena merupakan faktor dari 14 dan 63 (1 kita abaikan karena terlalu pendek).

Setelah mendapatkan panjang kunci, langkah selanjutnya adalah dengan megelompokkan chipertext berdasarkan kelipatan faktor pembagi. Setelah tu, kita melakukan serangan *Frequency Analysis* pada *ciphertext* dengan menggunakan fakta bahwa kata ‘THE’ adalah kata yang paling umum muncul pada bahasa Inggris. Sehingga dengan mengasumsikan bahwa *tri-graph* ERM adalah ‘the’ dan ERM dienkripsi oleh huruf ke 3, 4, dan 5 dari kunci maka kita mendapat kunci sementara adalah --LKI—Kemudian kita mengenkripsi sementara *ciphertext* dengan kunci asumsi kita tersebut. Dengan menggunakan frasa-frasa yang dapat dengan mudah dilengkapi, kita akan mendapatkan bahwa kata kunci yang dimaksud adalah “walking”.

Contoh kasus di atas membuktikan bahwa *Vigenère Cipher* amat lemah, khususnya pada saat di mana kata kunci yang digunakan relatif jauh lebih pendek dari plainteksnya.

6.2. KRIPTANALISIS PADA *EAGER EVIL MUSICIAN CIPHER*

Seperti yang telah dipaparkan pada Bab 5, karena mengadopsi konsep *one-time pad*, *Eager Evil Musician Cipher* mempunyai resistansi terhadap *Kasiski Examination*. Hal ini terlihat pada analisis kemunculan *tri-graph* berdasarkan program CryptoHelper, di mana hampir semua kemunculan tri-graph yang ada berjumlah satu (hanya terdapat dua buah *tri-graph* dengan kemunculan dua kali dari 1372 *tri-graph* yang ada. Jarak kemunculan masing-masing *tri-graph* tersebut juga tidak menunjukkan kelipatan dari 7), sehingga tidak memungkinkan dilakukan serangan *Kasiski Examination* pada *ciphertext* tersebut.

Selain itu, distribusi kemunculan karakter pada *ciphertext* hasil *Eager Evil Musician Cipher* cenderung merata dan bahkan mendekati distribusi normal. Hal ini dibantu dengan fakta bahwa plainteks yang ada dienkripsi dengan kunci yang memiliki panjang yang sama dan digunakannya tabel *extended* ASCII sebagai basis subtitusi karakter.

7. KESIMPULAN

Dari berbagai analisis serta studi kasus yang telah dilakukan, penulis mengambil kesimpulan sebagai berikut:

- *Eager Evil Musician Cipher* dapat mengemulasi karakteristik *one-time pad* tanpa perlu menyimpan kata kunci yang memiliki panjang sama dengan plainteks yang ada. Yang diperlukan hanyalah sebuah kata kunci biasa dan dua buah bilangan bulat positif, u dan v, yang berguna untuk membentuk Deret Ulam (u, v) yang akan membangkitkan kunci acak dengan panjang sama dengan plainteks.
- Tidak seperti *Vigenère Cipher* yang rentan terhadap serangan *Kasiski Examination* dan *Frequency Analysis*, *Eager Evil Musician Cipher* memiliki ketahanan terhadap serangan tersebut. Hal ini berkat karakteristik dari *one-time pad* yang berhasil diemulasi dan penggunaan tabel *extended* ASCII sebagai tabel subtitusi karakter.
- Kelemahan yang dimiliki oleh *Eager Evil Musician Cipher* adalah *cost* yang cukup mahal untuk membangkitkan Deret Ulam (u, v) yang besar. Sehingga apabila teks yang akan dienkripsi amat besar, proses enkripsi akan memakan waktu cukup lama.

DAFTAR PUSTAKA

[1] Munir, Rinaldi, Diktat Kuliah IF5054 Kriptografi, Penerbit ITB 2006

[2] Breaking The Vigenere Cipher, http://nob.cs.ucdavis.edu/classes/ecs253-1999 01/ vigenere.html

[3] Ulam Sequence, http://mathworld.wolfram.com/UlamSequence.html

[4] Vigenere Cipher, http://en.wikipedia.org/wiki/Vigenere_cipher.html

[5] ASCII, http://en.wikipedia.org/wiki/ASCII.html

LAMPIRAN

Plainteks Studi Kasus

BEFORE I CAME TO CHINA I TRIED TO LEARN A BIT ABOUT CHINESE ETIQUETTE. THE ONE THING I REMEMBER LEARNING IS THIS WHEN IT COMES TO CLINKING WINE GLASSES AT A FORMAL RECEPTION IT'S POLITE TO MAKE SURE THAT THE RIM OF YOUR GLASS IS LOWER THAN THAT OF THE PERSON YOU'RE CLINKING WITH.
THE OTHER DAY I WAS AT A CHINESE NEW YEAR RECEPTION HOSTED BY THE STATE BROADCASTER CCTV. AT TOASTING TIME, THE CCTV EXECUTIVES FANNED ACROSS THE ROOM WITH THEIR WINE GLASSES HELD OUT AT ABOUT WAIST LEVEL, MAKING IT IMPOSSIBLE FOR ANY OF US TO GET OUR GLASSES LOWER THAN THEIRS. THEY WON THE POLITENESS BATTLE, BUT DON'T CONFUSE ETIQUETTE WITH A LACK OF AMBITION.
THIS YEAR CCTV IS PLANNING TO OPEN RUSSIAN AND ARABIC LANGUAGE CHANNELS REPORTSAL, SO SAY THAT THE STATE RUN NEWS AGENCY, XINHUA, HAS PLANS TO START UP ITS OWN INTERNATIONAL NEWS CHANNEL. CHINA IS DETERMINED TO PROJECT ITS OWN IMAGE AND PERSPECTIVE AS FAR AS IT CAN SO THIS COUNTRY'S IMAGE MAKERS, MAYBE INTERESTED IN THE RESULTS OF AN OPINION POLL CONDUCTED FOR THE BBC WORLD SERVICE.
THE POLL SUGGESTS THAT GLOBAL ATTITUDES TOWARDS CHINA AND RUSSIA ARE WORSE THAN THEY WE'RE A YEAR AGO, ALTHOUGH CLEARLY THERE ARE A HUGE NUMBER OF CAVE AT STO THROW AT A SINGLE POLL WHICH TAKES BROAD CONCLUSIONS FROM A LIMITED NUMBER OF INTERVIEWS.
AS I'M SURE MANY OF YOU WILL BE KEEN TO TELL ME BUT HOW THE WORLD SEES CHINA AND HOW CHINA SEES THE WORLD MATTER QUITE A BIT AN AMERICAN WRITER JOSHUA KURLANTZICK GOES THROUGH SOME OF CHINA'S IMAGE MAKING EFFORTS IN HIS BOOK CHARM OFFENSIVE HOW CHINA'S SOFT POWER IS TRANSFORMING THE WORLD. THE WRITER LISTS THE THINGS THAT CHINA'S DONE IN ORDER TO WIN FRIENDS AROUND THE WORLD, ITS ENDED BORDER DISPUTES WITH ITS NEIGHBOUR SIT PLAYS AN ACTIVE PART IN GLOBAL INSTITUTIONS SUCH AS THE UN AND THE WORLD TRADE ORGANIZATION. IT DOESN'T INTERVENE UNILATERALLY IN OTHER COUNTRIES INTERNAL WARS AND IT STOPPED FUNDING MAOIST STYLE REVOLUTIONS A LONG TIME AGO.
MR KURLANTZICKAL SO ARGUES THAT CHINA'S MODEL OF AUTHORITARIAN CAPITALISM PROVIDES ATEMPTING ALTERNATIVE TO GOVERNMENTS. UNSURE OF THE BENEFITS OF WESTERN STYLE DEMOCRACY, THE CHINESE GOVERNMENT HAS COME UP WITH A PHRASE TO EXPLAIN WHAT IT'S DOING AND WHY NO ONE NEED WORRY HE PING JUE QI PEACEFUL RISE HAVE CHINA'S EFFORTS WON YOU OVER.

Potongan Source Code Program *Eager Evil Musician Cipher (Open Source)*

```
/*
 * File: Ulam List.cs
 * Berisi sebuah ulam list, dan proses pembangkitannya
 */
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Eager_Evil_Musician
{
    public class Ulam_List
    {
        //Variabel Global
        public List<long> uList;


      //Konstruktor
```

```
        public Ulam_List(int u, int v, int n)
        {
            uList = new List<long>();
            generateUlist(u, v, n);
        }

        //Fungsi untuk membangkitakan n bilangan pertama Deret Ulam (u, v)
        public void generateUlist (int u, int v, int n)
        {
            List<long> temp = new List<long>();
            List<long> buff = new List<long>();

            temp.Add(u); temp.Add(v);
            uList.Add(u); uList.Add(v);
            long cand = u+v;
            long count=0;
           int i=0;
           long x;

            while (uList.Count < n)
            {
                while (i < uList.Count)
                {
                    x = uList[i] ;
                    foreach (long y in temp)
                    {
                        if((x!=y) && (y<=x))
                        {
                            buff.Add(x+y);
                        }
                    }
                    i++;
                }

                for (int j = 0; j < buff.Count; j++)
                {
                    if (cand == buff[j])
                        count++;
                    if (count > 1)
                        break;

                }
                if (count == 1)
                {
                    uList.Add(cand);
                    temp.Add(cand);
                }
                count = 0;
                cand++;
            }
        }
    }
}
```

```
/* File: Evil Vigenere.cs
 * Berisi metode enkripsi dan dekripsi menggunakan Eager Evil Vigenere Cipher
 */

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Eager_Evil_Musician
{
    public class EvilVigenere
    {
        //Variabelm Global
        public String plaintext;
        public String key;
        public String ciphertext;
        public Ulam_List ulist;
        public bool isencrypt;
        public string extKey;

        //Konstruktor
        public EvilVigenere(string pctext, string _key, bool mode)
        {
            isencrypt = mode;
            if (isencrypt)
            {
                plaintext = pctext;
                ciphertext = "";
            }
            else
            {
                ciphertext = pctext;
                plaintext = "";
            }
            key = _key;
            ulist = new Ulam_List(1,2,pctext.Length);
            extKey = "";
            generateExtKey();

        }

        //Membangkitkan Kunci Acak dengan bantuan deret ulam
        public void generateExtKey()
        {
            int ctr = 0;
            long buff=0;
            long rem=0;
            foreach (long x in ulist.uList)
            {
                buff = ((int)key[ctr]) + x;
                Math.DivRem(buff,256,out rem);
                extKey += ((char)rem).ToString();


                if (ctr == key.Length-1)
```

```
                    ctr = 0;
                else
                    ctr++;
            }
        }

        //Metode Enkripsi
        public void encrypt()
        {

            long buff = 0;
            long rem = 0;
            for (int i = 0; i < extKey.Length; i++)
            {
                 buff = ((int)plaintext[i]) + ((int)extKey[i]);
                 Math.DivRem(buff,256,out rem);
                 ciphertext += ((char)rem).ToString();

            }
        }

        //Metode Dekripsi
        public void decrypt()
        {

            long buff = 0;
            long rem = 0;

            for (int i = 0; i < extKey.Length; i++)
            {
                buff = ((int)ciphertext[i]) - ((int)extKey[i]);
                Math.DivRem(buff, 256, out rem);
                rem--;
                if (rem < 0)
                    rem += 256;
                plaintext = extKey;
            }
        }
    }
}
```