Penerapan Prinsip *Confusion* dan *Diffusion* pada Algoritma Kriptografi Klasik: Algoritma *Pulse*

Danny Andrianto 13510011
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
danny.andrianto@itb.ac.id 13510011@std.stei.itb.ac.id

***Abstract*—Pada makalah ini, penulis akan merancang sebuah algoritma enkripsi klasik (berbasis karakter) dengan menerapkan prinsip *Confusion* dan *Diffusion*. Penulis juga akan menlakukan kriptanalisi terhadap algoritma yang berhasil penulis buat untuk menguji kaeamanan dan ketahannya**

***Index Terms*—algoritma kriptografi klasik, *confusion*, *diffusion*, algoritma *Pulse*, plainteks, kunci, cipherteks**

I. PENDAHULUAN

A. Latar Belakang

Pada kriptografi, ada dua macam cipher yaitu cipher klasik dan cipher modern. Cipher klasik adalah cipher yang beroperasi dengan basis karakter sedangkan cipher modern adalah cipher yang beroperasi dengan berbasis bit.

Algoritma kriptografi klasik yang ada kebanyakan sudah sangat tua dan bisa dipecahkan dengan mudah dengan *ciphertext-only attack*. Hal ini disebabkan karena algoritma kriptografi klasik sangat rentan pada *bruteforce attack* dan analisis frekuensi. *Bruteforce attack* sangat mungkin dilakukan karena umumnya kombinasi key yang bisa dipakai pada algoritma kriptografi klasik sedikit sehingga tidak butuh waktu yan sangat lama untuk mencoba semua kuncinya. Analisis frekuensi sangat mugkin dilakukan pada algoritma kriptografi klasik karena algoritma kriptografi klasik tidak dapat menyembunyikan hubungan antara plainteks, cipherteks, dan kunci dan juga fakta dimana perubahan kecil yang dilakukan pada plainteks tidak akan mengubah cipherteks secara signifikan membuat kriptananalis bisa menerka isi plainteks hanya dari cipherteksnya.

Pada algoritma kriptografi modern, terdapat prinsip untuk menjamin hal yang terjadi pada algoritma kriptografi klasik tidak dapat dilakukan yaitu prinsip *confusion* dan *diffusion* yang dikemukakan oleh Claude Shanon pada makalahnya yang berjudul *"Communication Theory of Secrecy Systems"*. Kedua prinsip inilah yang membuat algoritma kriptografi modern jauh lebih aman dibandingkan algoritma kriptografi klasik. Selain itu, karena algoritma kriptografi modern berbasis bit, dengan menambah jumlah bit yang dibutuhkan untuk kunci sudah dapat membuat *bruteforce attack* mustahil dilakukan.

Pada makalah ini, penulis akan mengimplementasikan prinsip-prinsip yang disebutkan pada paragraf sebelumnya untuk membuat algoritma kriptografi klasik yang tahan akan analisis frekuensi maupun *bruteforce attack.*Algoritma yang penulis hasilkan pada makalh ini dinamakan algoritma *Pulse.*

B. Rumusan Masalah

Sehubungan dengan latar belakang yang telah penulsi kemukakan di atas, maka penulis merumuskan masalah:

a. Algoritma *confussion* seperti apa yang cocok untuk kriptografi klasik?
b. Algoritma *diffusion* seperti apa yang cocok untuk kriptografi klasik?
c. Bagaimana ketahanan algoritma yang dibuat terhadap analisis frekuensi dan *bruteforce attack*?

C. Batasan Masalah

Pada pengerjaan makalah ini, penulis akan:

1. Merancang suatu algoritma kriptografi klasik (brebasis karakter) yang mengimplementasi prinsip *confusion* dan *diffusion*.
2. Melakukan kriptanalisis terhadap chiperteks yang dihasilkan dari algoritma yang dibuat.

D. Tujuan Penulisan

Berdasarkan latar belakang dan perumusan masalah di atas, penulisan makalh ini bertujuan untuk:

1. Menghasilkan algoritma kriptografi klasik yang tahan terhadap *bruteforce attack* dan analisis frekuensi.

II. DASAR TEORI

A. Algoritma Kriptografi Klasik

Algoritma kriptografi klasik merupakan jenis kriptografi yang digunaakan pada zaman dahulu dengan hanya menggunakan kertas dan pena dan berbasis karakter. Saat ini, setiap algoritma klasik sudah using karena sangat mudah dipecahkan. Algoritma kriptografi klasik ada dua mcam, yaitu cipher substitusi dan cipher trnasposisi.

Pada makalah ini, definisi algoritma kriptografi klasik penulis persempit menjadi algoritma kriptografi yang melakukan enkripsi/dekripsi dengan berbasis karakter. Hal ini mempertimbangkan akan sulit melakukan operasi enkripsi/dekripsi hanya dengan kertas dan pena karena menerapkan prinsip *confusion* dan *diffusion* yang diterapkan pada algoritma kriptografi modern.

B. Prinsip Penyandian Shannon

Pada tahun 1949, Shannon mengemukakan dua prinsip penyandian data untuk perancangan cipher blok yang kuat yaitu:

1. *Confusion*
   Prinsip ini menyembunyikan hubungan apapun yang ada antara plainteks, cipherteks, dan kunci.
2. *Diffusion*
   Prinsip ini menyebarkan pengaruh satu bit plainteks atau kunci ke sebanyak mungkin cipherteks. Prinsip ini juga menyembunyikan hubungan statistic antara plainteks, cipheteks, dan kunci dan membuat kriptanalisis menjadi sulit

Kedua prinsip di atas, selalu dipakai dalam perancagan cipher blok dan diulang berkali-kali pada satu blok dengan kombinsi yang berbeda-beda untuk mendapatkan keamanan yang bagus.

III. DESAIN ALGORITMA PULSE

Seperti yang telah penulis jelaskan sebelumnya, algoritma yang akan penulsi buat akan menerapkan prinsip *confussion* dan *diffusion*. Pada bagian ini, pertama-tama penulis akan menjelaskan desain algoritma yang penulis namakan *Pulse* berdasarkan prinsip yang ingin diimplementasi.

A. *Confussion*

Prinsip *confussion* akan diimplementasi untuk membuat analisis frekuensi tidak mungkin dilakukan.

Sebelum mulai, mari kita lihat salah satu algoritma *cipher* klasik yang dulu didesain untuk menangkal uji statistic ini yaitu *vigenere cipher*. Pada *vigenere cipher*, substitusi dilakukan berdasrkan dua masukan yaitu karakter palinteks saat itu dan kunci saat itu. Jadi metode subsitutsi yang penulis terapkan mirip dengan metode subsitusi *vigenere cipher.* Sekilas memang rasanya algoritma in akan menangal uji statisik karena setiap karakter plainteks jika memiliki kunci yang berbeda akan menghasilkan karakter cipherteks yang berbeda. Namun, *Kasiski Examination* berhasil mengeksploitasi kelemahan yang ada pada algoritma ini. Pada algoritma *vigenere cipher*, jika plainteks masukan user lebih panjang dibandingkan key-nya key akan diperpanjang dengan menggunakan kata yang sama berulang-ulang. Jadi, ada kemungkinan munculnya cipherteks yang berulang karena kunci yang berulang tersebut. Dari cipherteks yang berulang, dapat dilakukan analisis frekuensi untuk mendapatkan kunci yang dipakai.

Untuk menangani hal ini, penulis bercermin pada *unbreakable cipher* yaitu one-time pad. Dari dua persyaratan yang dimiliki one-time pada, ada satu yang tidak dipenuhi vigenere cipher yaitu kunci harus benar-benar acak. Kan tetapi jika menggunakan kunci yang benar-benar acak, tentunya akan sangat menyulitkn pengguna akrena harus membuat sendiri kunci yang sepanjang dengan plainteks. Karena penulis ingin membuat algoritma kriptografi yang praktikal, penulis memutuskan untuk mengambil jalan lain. Pada algoritma *Pulse*, jika panjang key lebih pendek dibandingkan panjang plainteks, kunci akan diperpanjang dengan karakter-karakter yang di-*generate* dengan cara *pseudo random*. Perbedaan *pseudo random* dengan *absolute random* yang dimaksud pada one-time pad adalah, dengan algoritma yang sama, *pseudo random* dapat menghasilkan hasil yang sama jika input awalnya sama. Dengan begitu, pengguna dapat menggunakan kunci yang mudah diingat untuk melakukan enkripsi maupun dekripsi.

Penggunaan *pseudo random* ini diharapkan selain membuat *Kasiski Examination* mustahil dilakukan, juga menjaga algoritma tetap praktikal.

B. *Diffusion*

Untuk implementasi prinsip *diffusion*, penulis melihat algoritma-algoritma yang ada pada block cipher. Pada block cipher, dari 4 mode yang ada, ada 3 mode yang sudah mengimplementasi prinsip diffusion yaitu CBC, CFB, dan OFB dimana hasil enkripsi satu blok dipropagasi untuk melakukan enkripsi block selanjutnya. Selain itu, ada juga konsep yang disebut *Feistel Network* yang mana mengimplementasi kedua prinsip baik *confussion* ataupun *diffusion.* Namun, penulis rasa penerapan

CBC, CFB, dan OFB bersama dengan *Feistel Network* kurang tepat untuk diimplementasi pada algoritma kriptografi berbasis karakter karena terlalu rumit. Pada bagian ini, penulis memutuskan untuk mengadopsi metode CFB (versi panjang bit yang digunakan sama dengan panjang blok) pada algoritma yang penulis buat.

Sama dengan CFB pada *block cipher*, untuk algoritma *Pulse*, CFB yang tadinya melakukan enkripsi per blok dimodifikasi untuk melakukan enkripsi per-karakter. Penerapan metode ini akan membuat perubahan satu karakter pada plainteks membuat perubahan karakter pada cipherteks merambat hingga ujung. Dengan begitu, prinsip *diffusion* akan terpenuhi.

Akan tetapi, hanya dengan menerapkan metode CFB, menurut penulis masih kurang dalam memenuhi prinsip *diffusion* karena perubahan hanya satu karakter hanya akan mempengaruhi karakter-karakter setelah yang. Oleh karena itu, penulis memutuskan untuk memberikan tambahan pada algoritma yang sudah ada agar perubahan satu karakter mnyebabkan perubahan pada keseluruhan plainteks.

Setelah melakukan berbagai ekseperimen, akhirnya penulis mendapatkan algoritma yang cocok untuk melakukan difusi yang penulis inginkan. Jadi, setelah sukses melakukan enkripsi dengan metode CFB, cipherteks akan ditransposisi terlebih dahulu. Lalu, cipherteks hadil transposisi tersebut akan dienkripsi lagi mengguanakan kunci yang sama. Walaupun sekilas terlihat sederhana algoritma ini efektif dalam menyebarkan pengaruh satu karakter ke karakter-karakter lainnya.

Setelah desain untuk dua prinsip tadi selesai dibuat, berikutnya penulis akan membahas fitur tambahan yang akan diimplementasi pada algoritma *Pulse*. Salah satu, kelemahan algoritma kriptografi klasik adalah hanya bisa melakukan enkripsi pada akrakter alphabet. Hal ini meyebabkan setiap kali sebelum melakukan enkripsi, setiap karakter diluar alphabet pada plainteks akan dihilangkan dan hasil enkripsi pun tidak menyimpan karakter-karakter tersebut. Kosnekuensi dari hal ini, jika mendekripsi pesan, pesan hanya akan mengandung alphabet tanpsa spasi, koma, titik, dan lainnya. Hal ini membutuhkan usaha lebih untuk memberikan tanda baca dan spasi pada teks hasil dekripsi yang tentunya memakan waktu yang tidak sedikit. Oleh karena itu, penulis melakukan modifikasi pada table subsittusi yang digunakan agar men-*supprot* tidak hanya 26 karakter alphabet, tetapi sekitar 98 karakter-karakter yang umum digunakan pada dokumen. Berikut adalah daftar lengkap karakter yang dapat digunakan.

	!	“	#	$	%	&	‘	(	)
*	+	,	-	.	/	0	1	2	3
4	5	6	7	8	9	:	;	<	=
>	?	@	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	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	{	|	}	~	TAB	LF	CR		

Keterangan:

- TAB: horizontal tab (‘\t’)
- LF: line feed, new line (‘\n’)
- CR: carriage return (‘\r’)

IV. Implementasi Algoritma Pulse

Jika pada bagian sebelumnya, penulis hanya membahasa desain dan konsep yang diterapkan pada algoritma *Pulse*, bagian ini akan menjelaskan implementasi algoritma *Pulse* secara nyata. Implementasi akan dibagi dua bagian yaitu *pseudo random key generator* dan enkripsi/dekripsi.

A. *Pseudo Random Key Generator*

Bagian ini akan membahas algoritma yang penulis buat untuk mengimplementasi desain *pseudo random key* yang penulis tulis pada bagian sebelumnya.

Pengimplementasian *pseudo random key generator* ini menggunakan operasi aritmatika, oleh karena itu, pertama-tama penulis mengassign dahulu tiap karakter dengan satu kode bertipe integer untuk emmudahkan penghitungan. Kode yang dipakai iritannya mirip dengan *encoding* ASCII/UTF-8 dengan sedikit modifikasi. Jadi ada 98 karakter dengan kode 0 s.d. 97. Berikut adalah table masing-masing karakter beserta kodenya.

SPACE	0	A	33	b	66
!	1	B	34	c	67
“	2	C	35	d	68
#	3	D	36	e	69
$	4	E	37	f	70
%	5	F	38	g	71
&	6	G	39	h	72
‘	7	H	40	i	73
(	8	I	41	j	74
)	9	J	42	k	75
*	10	K	43	l	76
+	11	L	44	m	77
,	12	M	45	n	78
-	13	N	46	o	79
.	14	O	47	p	80
/	15	P	48	q	81
0	16	Q	49	r	82

1	17	R	50	s	83
2	18	S	51	t	84
3	19	T	52	u	85
4	20	U	53	v	86
5	21	V	54	w	87
6	22	W	55	x	88
7	23	X	56	y	89
8	24	Y	57	z	90
9	25	Z	58	{	91
:	26	[	59 |	92
;	27	\	60	}	93
<	28	]	61	~	94
=	29	^	62	TAB	95
>	30	_	63	LF	96
?	31	‘	64	CR	97
@	32	a	65		

Langkah-langkah men-*generate pseudo random key* adalah sebagai berikut.

1. Hitung panjang key
2. Hitung total kode karakter kunci
3. Hitung rata-rata kode karakter kunci (dibulatkan ke bawah)
4. Inisialisasi variable karakter sebelum dengan rata-ratra
5. Isi karakter berikutnya dengan rumus matematika: (karakter sebelum x panjang key + KPK(karakter sebelum, total kode)) mod 98
6. Ulangi langkah 5 hingga panjang kunci sama dengan panjang plainteks

Misal:
Panjang plain teks = 14 karakter
Kunci masukan = lemon
Kunci akhir = lemonM][OiAw3#

Bisa dilihat pada langkah-langkah di atas bahwa rumus *pseudo random* yang digunakan cukup rumit. Hal ini untuk mengantisipasi agar tidak ada pola yang pada key hasil pseudo generator. Hasilnya pun, seperti bisa dilihat pada contoh di atas, cukup rumit.

B. Enkripsi/Dekripsi

Setelah selesai men-*generate* kunci sepanjang plainteks, proses enkripsi bsia dimulai. Proses enkripsi/dekripsi bisa dibagi dalam 3 tahap yaitu rantai 1, *reversal*, dan rantai 2. Secara umum berikut langkah-langkah dari tiap tahap. Seperti yang telah penulis jelaskan pada bagian desain, table subsitusi yang dipakai mirip dengan table substusi *vigenere cipher*, tetapi memiliki lebih banyak karakter seperti

Enkripsi:

- Rantai 1
  1. Inisialisasi variable karakter sebelum dengan rata-rata kode key
  2. Subsitusi karakter saat ini berdasarkan table substitusi. Gunakan karakter pada plainteks sebagai index baris dan karakter sebelum sebagai index kolom
  3. Isi variable karakter sebelum dengan subsitusi karakter hasil langkah 2 sebagai index baris dan kode karakter pada key sebagai index kolom
  4. Ulangi langkah 2 hingga semua karakter disubsitusi
- *Reversal*
  1. Putar balik string hasil enkripsi rantai 1
- Rantai 2
  1. Ulangi langkah-langkah yang dilakukan pada rantai 1 dengan string hasil tahap *reversal* sebagai plainteksnya

Dekripsi:

- Rantai 1
  1. Inisialisasi variable karakter sebelum dengan rata-rata kode key
  2. Subsitusi karakter saat ini berdasarkan tabel substitusi. Gunakan karakter pada cipherteks sebagai karakter pada tabel dan karakter sebelum sebagai index kolom, lalu cari karakter plainteks yang tepat dari indeks baris yang sesuai.
  3. Isi variable karakter sebelum dengan subsitusi karakter saat ini sebagai index baris dan kode karakter pada key sebagai index kolom
  4. Ulangi langkah 2 hingga semua karakter disubsitusi
- *Reversal*
  1. Putar balik string hasil dekripsi rantai 1
- Rantai 2
  2. Ulangi langkah-langkah yang dilakukan pada rantai 1 dengan string hasil tahap *reversal* sebagai cipherteksnya

Penulis juga melampirkan *source code* program yang penulis buat yang mengimplementasi algoritma *Pulse* ini yang ditulis dalam bahasa C# agar pembaca bisa lebih memahami.

Algoritma enkripsi yang digunakan tidak begitu rumit agar enkripsi bisa dilakukan dengan cepat. Penggunaan metode CFB dan transposisi sudah cukup untuk membuat cipherteks yang acak.

V. PENGUJIAN ALGORITMA PULSE

Dalam pembuatan makalah ini, penulis telah melakukan beberapa pengujian untuk mengetahui kekuatan dari algoritma *Pulse*. Beberapa eksperimen yang penulis lakukan adalah mengubah plainteks atau kunci sedikit untuk melihat perubahan hasil cipherteks, mengubah cipherteks atau kunci sedikit untuk melihat perubahan hasil plainteks, dan analisis frekuensi untuk melihat keterkaitan antara plainteks dan dipherteks.

A. Mengubah plainteks/kunci

Berikut adalah beberapa eksperimen yang penulis lakukan untuk melihat persebaran pengaruh karakter:

Plainteks : attack at dawn
Kunci : lemon
Cipherteks : IXSV0<\pUYGC>{

Plainteks : attack tt dawn
Kunci : lemon
Cipherteks : □*@-L□x|jfa<

Plainteks : attack at daw
Kunci : lemon
Cipherteks : mojBO.2{)_!Hg

Plainteks : attack at dawn
Kunci : lenon
Cipherteks : Ai[W1^qZ+rZV5m

Plainteks : attack at dawn
Kunci : lemo
Ciphertaks : 4MA0rSH'xU)2a9

Dari hasil-hasil enkripsi di atas, bisa dilihat bahwa hanya dengan perubahan sedikit pada plainteks maupun key dapat memberikan hasil cipherteks yang sama sekali berbeda dan seakan-akan random. Ini menunjukkan konsep-konsep yang diimplementasi untuk melakukan difusi sudah melakukan pekerjaannya dengan baik dan sesuai dengan harapan.

B. Mengubah cipherteks/kunci

Berikut adalah beberapa eksperimen yang penulsi lakukan untuk melihat robustness dari algoritma

Cipherteks : IXSV0<\pUYGC>{
Kunci : lemon
Plainteks : attack at dawn

Cipherteks : IXSV0b\pUYGC>{
Kunci : lemon
Plainteks : attack ;^\dawn

Cipherteks : IXSV0<\pUYGC>
Kunci : lemon
Plainteks : =miejaqrv~9K*

Cipherteks : IXSV0<\pUYGC>{
Kunci : lenon
Plainteks : f)Xg|`7CFAkc2s

Cipherteks : IXSV0<\pUYGC>{
Kunci : lemo
Plainteks : d=9&p:_WtQ~TCq

Dari hasil-hasil dekripsi di atas, bisa dilihat bahwa jika ada satu karakter yang berubah pada plainteks tidak mempengaruhi hasil dekripsi secara keseluruhan, hanya ada beberapa karakter yang terkena pengaruhnya. Akan tetapi, jika ada karakter yang hilang, hasil dekripsi sangat jauh dari yang semestinya. Di lain hal, jika kunci berubah sedikit saja, hasil dekripsi akan sangat berbeda dengan teks yang semestinya. Ini menunjukkan kalau algoritma ini cukup *robust* dalam menghadapi *typo* atau salah ketik pada cipherteks, tetapi kurang *robust* dalam menangani pesan yang terpotong atau hilang. Akan tetapi, algoritma ini sangat baik dalam hal menangani kunci yang salah karena plainteks yang dihasilkan sangat berbeda walaupun kunci tidak berbeda jauh.

C. Analisis frekuensi

Analisis frekuensi yang dilakukan penulis mengunakan plainteks yang sangat panjang dengan kunci "oceanography". Plainteks dan cipherteks yang penulis gunakan bisa dilihat di lampiran. Berikut adalah hasil analisis frekuensi dari masing-masing plainteks dan cipherteks:

Plainteks:

SPACE = 1807	! = 0	" = 0
# = 0	$ = 0	% = 0
& = 0	' = 15	(= 7
) = 7	* = 0	+ = 0
, = 118	- = 13	. = 75
/ = 1	0 = 5	1 = 4
2 = 3	3 = 5	4 = 3
5 = 0	6 = 3	7 = 6
8 = 2	9 = 5	: = 5
; = 3	< = 0	= = 0
> = 0	? = 0	@ = 0
A = 19	B = 17	C = 7
D = 8	E = 6	F = 1
G = 2	H = 11	I = 13
J = 0	K = 0	L = 2
M = 3	N = 8	O = 3
P = 25	Q = 0	R = 12
S = 10	T = 29	U = 1
V = 7	W = 1	X = 0
Y = 0	Z = 0	[= 0
\ = 0	] = 0	^ = 0
_ = 0	` = 0	a = 771

b = 109	c = 311	d = 343
e = 1258	f = 225	g = 173
h = 537	i = 651	j = 4
k = 58	l = 328	m = 216
n = 671	o = 684	p = 210
q = 19	r = 589	s = 636
t = 937	u = 326	v = 122
w = 157	x = 20	y = 154
z = 2	{ = 0	| = 0
} = 0	~ = 0	TAB = 0
LF = 204	CR = 0	

Cipherteks:

SPACE = 94	! = 116	" = 122
# = 138	$ = 114	% = 130
& = 121	' = 109	(= 118
) = 120	* = 115	+ = 117
, = 135	- = 119	. = 124
/ = 114	0 = 120	1 = 122
2 = 132	3 = 122	4 = 122
5 = 144	6 = 109	7 = 112
8 = 143	9 = 102	: = 112
; = 119	< = 118	= = 134
> = 132	? = 125	@ = 127
A = 133	B = 126	C = 122
D = 137	E = 129	F = 118
G = 116	H = 128	I = 106
J = 122	K = 128	L = 116
M = 128	N = 116	O = 134
P = 119	Q = 112	R = 113
S = 130	T = 134	U = 110
V = 127	W = 124	X = 120
Y = 143	Z = 117	[= 128
\ = 110	] = 113	^ = 119
_ = 136	` = 120	a = 124
b = 128	c = 124	d = 101
e = 130	f = 127	g = 110
h = 119	i = 126	j = 113
k = 97	l = 109	m = 114
n = 141	o = 135	p = 111
q = 122	r = 104	s = 133
t = 136	u = 140	v = 127
w = 149	x = 151	y = 108
z = 130	{ = 131	| = 137
} = 120	~ = 105	TAB = 116
LF = 118	CR = 114	

Hasil analisis frekuensi di atas menunjukkan tidak ada hubungan yang terlihat antara frekuensi karakter pada plainteks dan cipherteks dimana jika di plainteks sangat terlihat karakter mana saja yang sering muncul dan yang jarang muncul, jumlah kemunculan karakter pada cipherteks tersebar secar merata dan tidak terlihat terlihat jelas karakter mana yang muncul lebihs sedikit atau lebih banyak dari yang lain.

Lalu, karena pada desain awal, algoritma ini juga dirancang agar kebal terhadap pada *Kasiski Examination*, penulis juga melakukan *Kasiski Examination* pada cipherteks untuk mencoba mendapatkan panjang kunci dengan menggunakan kakas yang disediakan pada laman http://pages.central.edu/emp/LintonT/classes/spring01/cryptography/java/kasiski.html. Berikut sebagian hasilnya:

PED: occurs 2 times, at pos 0, 6335
distance(s): 6335
EDE: occurs 3 times, at pos 1, 1823, 6336
distance(s): 1822, 6335, 4513
ZTQ: occurs 2 times, at pos 13, 2510
distance(s): 2497
TQZ: occurs 2 times, at pos 14, 4553
distance(s): 4539
BVE: occurs 2 times, at pos 18, 378
distance(s): 360
HCJ: occurs 2 times, at pos 23, 3769
distance(s): 3746
CHW: occurs 2 times, at pos 33, 4213
distance(s): 4180
WTD: occurs 2 times, at pos 35, 2061
distance(s): 2026
KXU: occurs 2 times, at pos 44, 826
distance(s): 782
LFG: occurs 2 times, at pos 47, 3816
distance(s): 3769
GWW: occurs 2 times, at pos 49, 1050
distance(s): 1001
ESJ: occurs 2 times, at pos 53, 2973
distance(s): 2920
SJA: occurs 3 times, at pos 54, 3003, 4268
distance(s): 2949, 4214, 1265
JAX: occurs 2 times, at pos 55, 2163
distance(s): 2108
XWC: occurs 2 times, at pos 57, 2292
distance(s): 2235
WCS: occurs 2 times, at pos 58, 1357
distance(s): 1299
SNW: occurs 2 times, at pos 60, 938
distance(s): 878
BZY: occurs 2 times, at pos 68, 5627
distance(s): 5559
ZJV: occurs 2 times, at pos 72, 2078
distance(s): 2006
VBM: occurs 2 times, at pos 74, 3120
distance(s): 3046
BME: occurs 2 times, at pos 75, 3136
distance(s): 3061
MEH: occurs 4 times, at pos 76, 2208, 3137, 3433
distance(s): 2132, 3061, 3357, 929, 1225, 296

Hasil di atas menunjukkan bahwa algoritma ini kebal terhadap *Kasiski Examination*. Hal ini bisa dilihat dari hasil yang hampir semua angkanya bukanlah kelipatan dari 12 yang merupakan panjang dari kunci yang diapakai. Berarti, *Kasiski Examination* tidak dapat digunakan untuk menemukan panjang kunci yang digunakan dan algoritma *pseudo random* telah melakukan pekerjaannya sebagaimana mestinya.

VI. KESIMPULAN

Menjawab rumusan masalah yang penulis paparkan pada pendahuluan, berikut kesimpulan-kesimpulan yang dapat penulis ambil:

1. Algoritma *confusion* yang cocok untuk algoritma kritografi klasik, seperti yang penulis implementasikan, adalah menggunakan rumus *pseudo random* untuk men-*generate key* sepanjang plainteks dan menggunakan substitusi dengan table mirip *vigenere cipher.*

2. Algoritma *diffusion* yang cocok untuk algoritma kriptografi klasik, seperti yang penulis implementasikan, adalah menggunakan konsep CFB yang dikombinasikan dengan transposisi untuk menyebarkan pengaruh satu karakter ke teks hasil enkripsi.

3. Ketahanan algoritma *Pulse* terhadap analsisi frekuensi cukup terbukti pada eksperimen dimana hasil analisis frekuensi memberikan hasil yang menunjukkan kemunculan tiap karakter yang rata-rata sama. Sedangkan ketahanan terhadap bruteforce attack, mungkin hanya bisa diukur dengan menghitung total kombinasi yang ada untuk mengenkripsi suatu plainteks. Total kombinasi kunci yang mungkin untuk mengenkripsi suatu plainteks adalah $98^{\text{panjang\_plainteks}}$. Sebagai contoh, kombinasi karakter yang mungkin untuk mengenkripsi plainteks sepanjang 14 karakter adalah 98^{14} atau sekitar 7,5 x 10^{27} kombinasi. Jadi, jika diasumsikan sebuah karakter dapat mencoba 10^{12} kombinasi kunci dalam satu detik, maka butuh waktu sekitar 239 tahun untuk mencoba semua kombinasi. Jadi, bisa dinilai algoritma ini juga cukup tahan terhadap *bruteforce attack*.

Dari kesimpulan-kesimpulan di atas, penulis bisa dengan bangga menyatakan bahwa algoritma *Pulse* buatan penulis telah berhasil memenuhi tujuan penulisan makalah ini yakni menghasilkan algoritma kriptografi klasik yang tahan terhadap analisis frekuensi maupun *bruteforce attack.*

REFERENSI

[1] Munir, Rinaldi. 2006. *Diktat Kuliah IF5054: Kriptografi*. Bandung: Prodi Teknik Informatika, STEI-ITB

[2] Menesez, Alfre J., Paul C. van Oorschot, dan Schott A. Vanstone. 2002. *Handbook of Applied Cryptography*, *5th Printing*.

[3] http://www.asciitable.com/

[4] http://unicode-table.com/en/

[5] http://pages.central.edu/emp/LintonT/classes/spring01/cryptography/java/kasiski.html

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 26 Maret 2013

Danny Andrianto 13510011

Lampiran 1: Source code progam (PulseCipher.cs)

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

namespace Cryptography
{
    class PulseCipher
    {
        private const int NUM_OF_CHAR = 98;

        private static int LCM(int a, int b)
        {
            // Source: http://stackoverflow.com/questions/13569810/least-common-multiple

            int temp1, temp2;

            if (a > b)
            {
                temp1 = a;
                temp2 = b;
            }
            else
            {
                temp1 = b;
                temp2 = a;
            }

            for (int i = 1; i <= temp2; i++)
            {
                if ((temp1 * i) % temp2 == 0)
                {
                    return i + temp1;
                }
            }
            return temp2;
        }



        public static string encrypt_or_decrypt(string input, string key, bool encrypt)
        {
            int[] inputcode = new int[input.Length];
            int[] keycode = new int[input.Length];
            int[] outputcode = new int[input.Length];

            // Convert all character to codes
            for (int i = 0; i < input.Length; i++)
            {
                if (Convert.ToInt32(input[i]) < 32)
                {
                    if (input[i] == '\t')
                    {
                        inputcode[i] = 95;
                    }
                    else if (input[i] == '\n')
                    {
                        inputcode[i] = 96;
                    }
                    else if (input[i] == '\r')
```

```
            {
                inputcode[i] = 97;
            }
        }
        else
        {
            inputcode[i] = Convert.ToInt32(input[i]) - 32;
        }
        if (i < key.Length)
        {
            keycode[i] = Convert.ToInt32(key[i]) - 32;
        }
    }

    // get sum of key code
    int sum = 0;
    for (int i = 0; i < key.Length; i++)
    {
        sum += keycode[i];
    }

    // get average value of key code (floor)
    int average = sum / key.Length;

    // generate pseudo random key if needed
    if (input.Length > key.Length)
    {
        // pseudo random code
        int current = average;
        for (int i = key.Length; i < input.Length; i++)
        {
            keycode[i] = (current * key.Length + LCM(current, sum)) % NUM_OF_CHAR;
            current = keycode[i];
        }
    }

    // Make subsitution table
    int[] substitute = new int[NUM_OF_CHAR * NUM_OF_CHAR];
    int cc = 0;
    for (int row = 0; row < NUM_OF_CHAR; row++)
    {
        for (int col = 0; col < NUM_OF_CHAR; col++)
        {
            substitute[row * NUM_OF_CHAR + col] = cc;
            cc = (cc + 1) % NUM_OF_CHAR;
        }
        cc = (cc + 1) % NUM_OF_CHAR;
    }

    int previous = average;
    // First cycle
    for (int i = 0; i < input.Length; i++)
    {
        if (encrypt)
        {
            outputcode[i] = substitute[inputcode[i] * NUM_OF_CHAR + previous];
            previous = substitute[outputcode[i] * NUM_OF_CHAR + keycode[i]];
        }
        else
        {
            for (int j = 0; j < NUM_OF_CHAR; j++)
            {
                if (substitute[j * NUM_OF_CHAR + previous] == inputcode[i])
```

```
                    {
                        outputcode[i] = j;
                    }
                }
                previous = substitute[inputcode[i] * NUM_OF_CHAR + keycode[i]];
            }
        }
        int[] temp = new int[input.Length];
        Array.Copy(outputcode, temp, input.Length);
        // Reverse result
        for (int i = 0, j = input.Length - 1; i < input.Length; i++, j--)
        {
            temp[i] = outputcode[j];
        }
        // Second cycle
        previous = average;
        for (int i = 0; i < input.Length; i++)
        {
            if (encrypt)
            {
                outputcode[i] = substitute[temp[i] * NUM_OF_CHAR + previous];
                previous = substitute[outputcode[i] * NUM_OF_CHAR + keycode[i]];
            }
            else
            {
                for (int j = 0; j < NUM_OF_CHAR; j++)
                {
                    if (substitute[j * NUM_OF_CHAR + previous] == temp[i])
                    {
                        outputcode[i] = j;
                    }
                }
                previous = substitute[temp[i] * NUM_OF_CHAR + keycode[i]];
            }
        }

        // Convert cipher codes to characters
        char[] output = new char[input.Length];
        for (int i = 0; i < input.Length; i++)
        {
            if (outputcode[i] > 94)
            {
                if (outputcode[i] == 95)
                {
                    output[i] = '\t';
                }
                else if (outputcode[i] == 96)
                {
                    output[i] = '\n';
                }
                else if (outputcode[i] == 97)
                {
                    output[i] = '\r';
                }
            }
            output[i] = Convert.ToChar(outputcode[i] + 32);
        }

        return new string(output);
    }
  }
}
```

Lampiran 2: Berkas plainteks analisis frekuensi

It is certain that when the eruption of Vesuvius started on the morning of 24 August, AD 79, it caught the local population utterly unprepared. Although at the same time, as we now know in retrospect, all the tell-tale signs were there to warn them.
It is mainly thanks to the vivid eye-witness account of the younger Pliny (a Roman administrator and poet, whose many vivid letters have been preserved), that we have some understanding of what happened. And it is through him that we can gain insight into the reactions and feelings of the people caught up in the drama of this natural catastrophe.
Pliny's account leaves no doubt that everyone was caught unprepared. His uncle, known as Pliny the Elder, was stationed in command of the imperial naval base at Misenum, on the north-west extremity of the Bay of Naples. He was not only the senior military officer in the district, but possibly the most well informed living Roman on matters of natural science. His 37-volume Natural History is the longest work on science in Latin that has survived from antiquity.
But for all his science and his seniority, his nephew tells us that the elder Pliny was relaxing, after a bath and lunch, when Vesuvius started to erupt. And the sighting of a column of smoke 'like an umbrella pine' on the far side of the Bay triggered a response more of curiosity than of alarm in him. He and his companions were evidently not anticipating such an event.
The same account reveals, however, that the signs were there. Pliny's casual reference to earth tremors 'which were not particularly alarming because they are frequent in Campania' reveals the Roman's comprehensive ignorance of the link between seismic activity (earth tremors) and volcanic activity.
The volcanologists of today constantly monitor any changes in levels of seismic activity from the observatory on Vesuvius, because they know that the same increase of activity in the deep reservoir of magma (molten or partially molten rock beneath the Earth's surface) causes both earth tremors and volcanic eruptions. Through measuring seismic activity, these scientists expect to predict an approaching eruption months in advance.
They also know that the activity of Vesuvius is recurrent, and that the longer the intervals between one eruption and another, the greater the eventual explosion will be. The frequent but lowlevel activity of Vesuvius in recent centuries has relieved the build-up of pressure in the magma chamber. The catastrophic magnitude of the eruption of AD 79 was connected with the extended period of inactivity that preceded it. A long interval combined with mounting seismic activity is a sure sign of impending disaster.
Of course, the Romans could not know this, and our own knowledge owes much to the care of Pliny's description. The long inactivity of the volcano naturally lulled the people of the region into a false sense of security, though they were aware of the signs of burning at the peak of the mountain.
They were not the first to be so lulled: recent excavations at the site of the new NATO base at Gricignano, on the north of the Bay, have revealed two catastrophic eruptions that preceded that of 79, and wiped out the populations of a densely occupied territory. The most important earlier eruption, known as that of the 'Avellino pumice' occurred around 1800 BC; several sites, especially one near Nola, reveal the destruction of Bronze Age settlements, with their huts and pots and pans and livestock. But of this the Romans knew nothing.
The irony of this is that the Romans were extremely interested in predicting the future, and they had a range of ways to detect what they saw as the approaching wrath of the gods. They were adept, for example, at observing 'portents' in the shape of strange sights and sounds, or unusual births.

Even in these terms, there were warnings of the eruption of Vesuvius. Earthquakes in themselves counted as portentous, and the historian Cassius Dio, writing over a century later, reports repeated sightings of giants roaming the land. This was a bad portent indeed, given that one standard explanation for the volcanoes of south Italy was that, when the gods defeated the rebellious giants and brought peace to the universe, they buried them beneath the mountains, and that it was their stirrings that caused the eruptions.

But while the ancient imagination doubtless conjured up giants in plumes of gas from fumaroles (vents from which volcanic gas escapes into the atmosphere), the earthquakes that Pliny described so casually were more than just portents. Current thinking, however, had not yet caught up with their significance. We know this because, by an extraordinary coincidence, the philosopher Seneca, advisor to the emperor Nero, wrote a discussion of the scientific causes of earthquakes only a few years before the eruption.

Seneca's treatise on the causes of natural phenomena included an entire book on earthquakes, and at the time he was writing, the news was coming in freshly of the catastrophic earthquakes in Campania of AD 63, which caused extensive damage to both Pompeii and Herculaneum.

Seneca writes that he regarded it as likely that earthquakes in different parts of the world were interconnected, and even that they were linked to stormy weather, but he draws no link with volcanic activity. Indeed, he goes so far as to reproach the landowners who were deserting Campania for fear of further earthquakes.

The earthquake of AD 63 caused extensive damage to both Pompeii and Herculaneum, as we can see from repairs made to the buildings. Some areas seem to have been worse affected than others - there are cases where entire houses were demolished and reduced to agricultural land. Upper floors would have been particularly badly affected - and indeed some buildings do have blocked-up doors at the top, indicating that the higher floors had been abandoned.

But more impressive than the signs of damage are the signs of the resilience of the local population. Damaged houses were being extensively repaired and redecorated at the time of the AD 79 eruption, and there was a comprehensive programme of restructuring of public buildings in the Forum of Pompeii.

The evidence points to a continuous process of repairs and rebuilding from AD 63 onwards. It used to be assumed that the earthquake described by Seneca was the only cause of damage, and that signs of incomplete work suggested that it took the cities a long time to recover from the first catastrophe. But we now know from volcanological research that a series of seismic episodes immediately preceded the eruption, causing further damage to structures that had already been repaired.

So, in the house of the Chaste Lovers at Pompeii, archaeologists discovered that the oven of a bakery had suffered major cracking; it had been repaired and plastered over, but had then been damaged again - and building work was already in progress to mend this new damage. In the same block, three cesspits in the street, which linked to latrines in the houses, had been dug out immediately before the eruption, presumably to restore them to full functionality.

Outside in the main street, an open trench was found, cutting the entire length of the walkway as far as a water-tower at the crossroads: seismic activity had interrupted the water supply, but people had been hard at work repairing it. A frequent sight in the excavated houses of Pompeii is that of heaps of plaster, which must have been brought in ready for new decoration. Sometimes even the pots and compasses of the decorators are in position.

The Pompeians in August 79, far from abandoning their city, or fretting about earthquakes as portents of future destruction, were thus tenaciously repairing their city, and trying to carry on with

life as usual. There was every reason to: the economy of the Bay was booming, with the great port of Puteoli as one of the biggest nodes of Mediterranean trade, and the holiday villas of the rich bringing constant investment.
Taken unawares by the eruption, the population of the towns and villas that circled the Bay could only respond with panic. Pliny depicts his uncle as a model of Stoic fortitude: calmly sailing directly into the danger zone (where he subsequently died), and taking a bath, dinner and sleep while the catastrophe unfolded. But all around him is panic - Rectina in her villa, Pomponianus in his.
The young Pliny too stays calm, but his mother weeps and implores, and by the time they set out to flee northwards, a dense black cloud of ash has blotted out the light, and the crowds of screaming people fleeing around them are in terror. The skeletons found in Pompeii and Herculaneum give us an equally eloquent testimony of panic and uncertainty.
The eruption lasted for more than 24 hours from its start on the morning of 24 August. Those who fled at once, unburdened by possessions, had a chance of survival, for the rain of ash and pumice, mixed with lithics, that descended for several hours was not necessarily lethal. It is clear that many, like the elder Pliny, thought their best chance was to take shelter and weather the storm.
It was not until around midnight that the first pyroclastic surges and flows occurred, caused by the progressive collapse of the eruptive column, and these meant certain death for the people of the region. (A pyroclastic flow is a ground-hugging avalanche of hot ash, pumice, rock fragments and volcanic gas, which rushes down the side of a volcano as fast as 100 km/hour or more.)
The hundreds of refugees sheltering in the vaulted arcades at the seaside in Herculaneum, clutching their jewellery and money, met their end swiftly - from the intense heat of the first surge that reached the city.
Subsequent waves reached Pompeii, asphyxiating those who had survived the fall of 3m (10ft) of pumice, and were fleeing across the open in the dark, or hiding beneath roofs. The waves that followed smashed flat the upper floors of houses, and left the corpses encased in successive blankets of gaseous surge and pumice fall.
It is impossible to tell what proportion of the inhabitants died, but the Romans were accustomed to losses mounting to tens of thousands in battle, and even they regarded this catastrophe as exceptional. The corpses found by archaeologists in Pompeii or Herculaneum should be regarded as only a small sample: the destruction encompassed the entire landscape south of Vesuvius to the Sorrentine peninsular. As many died in the countryside or at sea as in the cities. Even as far north as Misenum, the ash lay deep in drifts.
The effect of the eruption was evidently totally traumatic, as is shown by the failure to reoccupy the sites of the cities destroyed. It was normal practice to rebuild the cities of this region after even the most massive earthquakes; but neither Herculaneum nor Pompeii was reoccupied.
Instead, the site of Pompeii was riddled with tunnels by explorers, not by modern explorers as is often imagined, but by the Romans themselves after the eruption. Room after room of the city's buildings had holes hacked through the walls by tunnellers, and though Pompeii has richer finds than any other Roman site, it is a city already extensively sacked by looters.
The cities on the north of the Bay swiftly recovered, and Puteoli continued to be a significant commercial centre. The Bay of Naples continued to attract rich holidaymakers, but never again regained the massive levels of popularity of the two centuries before the disaster, the time when it had been the playground of many rich senators and emperors.
It was not until the 18th century, when Naples flourished under

the Bourbon kings, that the villas of the rich courtiers and ambassadors of that time brought a new flowering to the region. It was at this period that the aristocrats of Europe, as they progressed on their Grand Tours, made the Bay of Naples and its hidden Roman treasures a focus of international fascination.

Lampiran 3: Berkas cipherteks analisis frekuensi

P2EDe2toc.24bq#M4i|Z:15=lzt□3Qz)n-
BvE~n8T/hCJG{zZQfwR(3c=*@H/7/8W]8Td#Bsc>auj~@Kx/[6□u"l6=Fg#~=9ww2D□ES.% • &-j*-
AXw?`c2S2€ n&}WD)
<?i_WR>.IB€ z>Y9gZ{)]<!2~J@5@VbMe^h.WE`l:rMP>S4qilVgd4$$$1%y(nit□!+FmA.VV!/[6swc-
}pt!vKS<ISR":2Ni7rAL-_2rXqc(|pY\Hnor%FooO#l`]$&^cf#MnSx-U$ip`whTQKHqe;CPhBcIk&T□+5t2cw-
k4=|:qVBWDy;Ns[*"InRG02□Z`dvAT?_|:f41m;NP[,RH]G=@7UR?\h!UJU□E(xMQ9dR<}uYxT
rjIOY8|\AD0`_Ti{0tBEh(R3p$vV;79m|^ZYnAO>_a"V~p=Z/vfsY!"tgksOxt=x`S-
G€ L8;0>4t| • {;P8?}lus7I=[$F.k{HuSA2YkPUf#]}p4X<□ZY1>y[@0w4r8q^B`_>@DOepSUGRw-{HJzC|mFgZ."--
HL@YzOJ;cH"z,B0KIu:si8<l;o2M/[qWA)=H1Oa.Z%(q;xP<TU)*-3ITx#
;Y#h8B(S.#w2;Kl6jGhosX]yH07.F[y]9<tL@PW2OT}R~a1I=+8[yi=U~U!wdx\/gXMhbHMUe=REm'cH!E4v{.8AA#4
Ca-
aiLd4kCB3VEwSMnFhwC`,x;?p0hO□!l6sQ7(c€ sUSmnC_|J[p>d]&E}Puq8R8$$Pd=>6No.w=U*Z:%TfVDEYuHQ;O
q&i#u;Fb2Kq□n%<Hs65fF){o44|sbZ~{Qgx8WlV • 6zK! • rGTRS`u[nm4iC=#b^Uj'@*OGn6va{i$?sXAWL!wZ;G,G5|
#4,Zmaft0f&□Ml,iHWP=U\g8ZT%9n_Q\Avnn • SyqF^&FV+FSs=xj5C!B{bCM"<"g[{tGLEq8:ivo%>Y-
YJw=vJwxR}xbPfa7IKe7lf+4b>$(5iw€ 8QZ:G5A\l{ • i"7jYDaIzwq|CE})z{8>lun&BiKr{A^,vc|uA'~hkZ"k'W2s{a0"ol
vb3#e^q□Xk_dr!VoMh • Q<,NK}mmmxuPgq1cLN7hH4#61SOFNS3JM@Rb{vL;g/uhOnwf!=Ny<1ES}_:C*xuqu62jll!
V&~^!t,=i)*X€ f];@□00GbCv-
rG96*XnWn€ >x8@|7e=qq<Y?"\S@`Z5#+(Zxi□5[E%<t:~welZ.uXMffD`]zLuU€ 2pH5JHmbF-.5n€ "X+gL
<*k`fShiH8%@fW+;5EbtLn • Ei=5v3.lh{!BJ874Ck#s€ )Q"<x/(Q&ejXmR&_P8F5r□5Bw=8RV}?^?`n:Y|a • uG\h.VC~
?€ ocpVlU59HQ • |SQLP|=>j2_6wzH_8{_`Ge]6/?DbgRNGT'Q\AiOL^)uA,"&E[6DLWq- • \n8tYI,E<vn} • <z>
g^oro)5e0^Q • 5p8g`NcD'ofa • GB@<J|3*_uI4voGmBw`TBtrSEULlj;:F`2H3KXu@s]/P.]3-
bzlXeoi.A<HR0{/|Eg7uLU=wr\Xs€ `s"Fw@Q-
[RF1:*A$WD:$60)3He+F4N\|7XBzB* • □G`>Xc(Z%&TrY=xzr>84Eh#xJnAyWYBqjTOVaAPOy>ZgjHW\#Xt]'G+oF
I%1rF0yVmO"#1D!TiKpSS:XW69EUpyTkwE&]jR*MazO_Ms5nW:haAKDAWU1GQtE|)□el!-MfD_o-
].z>T(_91!8)w2L?aod}<`?v/sS8)"4.w • +<fxp3ZA%am^0ueg0D>rB%g7=(V3ugpo34*D[Mu~hq(V-SWz+_B-
@+^ZSCZ]%}hs'MO%);f~h"4Ij-
z?L€ AlNkkEw]N5B9"=Gc8eb~(K*gnpEHTc}~QcXDKV1WPi□<%WgTCOl']{$Ng/w60\
W9w~c+xR/84bM/*?IlmOBAK€ E@i$V8|wLoAroyN`E,27@ • DQ|FgO$A}UKNEdN|fRPhzl})Mn#a%5jM;TF|{t□4O
@}8u?tk;P?€ !@ZHc`)R
k@[5oddKYN3?:Hu:&%v#>M&F9Y{0a{m/J"Y%'h,pA$,}1tWShd5PX€ V60e}oMAHL|1□(CmNjZ$=z]"nTK#adS|rI=
<C€ :F!M'
m;ABFWb$i6$"3OaRpo#Ob@huDd4)gl=`7}g)zWYc□a2FyH*&v}mDME&/0q"#9LbBO&~□eEX0<xUK;D.chx}Hh[
m~M%_\4N&Q-50RD1,$ DF5C@My5&W
X4s{]3|F€ |n74%Kn40sFK\r1:%;Sy:R5bo<wT\J,/gI6#H8z#+(Nje<wpve{}lzw$Ywg#"2NaDpo:pG)bsA`A<8MJ)-=P D-
RrAo7+\|K^f +k"5V!5€ Hb+U+)|@%[OUB<l*oTC=.ZdOFJR'?Gk9v\+:#i^PDej\~;\<vu%-EwBG-
UK@G1VT(?vfVnrfz□4p1{.E_#ML7vt #Oqf.F e//v<uO7*s3+v□1B'Sa€ ;j_He[OVZa20r • :ZAw□7€ `Wc(s!_:□
^qTzQ2o}G[5`p{NHp8e;io\~^N36&^r}<fw`v{BZ"uXqG}njM]CqlXH`kJUQPf n?Prw€ S|j4Z5.-
[gRNWiE\G_l7vUnbE93Am \I4Ad!f&}wz • R^=NZk3W2WiI,%?S8]c%Z-
4!PH"q}o92' • 52W`<4.3_dMn~>~Yl]1 • pT"#K"P2*$T_=I^v9SEVe+b16z/t`OI,I?xlkrMsl(N(}YyuR@?F!>.Lh4%_^8i\
LyqIcB#zzp76$'48ryp0Oc<cOw-
ZC)>1xnps8G*IY%c64_z^?waDN3xmlsPZ++7Ht$|Nv?v=0f)_@*@1XYVf€ .Wji%AsdMp`:)wXaEwt`^ •)ZVTo3@t+
2c5|#|81 • □+0oyl'Ad=Q8Q_p+P7K@□Md9YQv)AsD<cmC"ftxL?1:aoDcc([;5j • lYWRGcJupeOakNq-
mHzrR%kN?XdCA.8ZpPw□62,EP6m%R7!:;qs~*CR=`z5CAl! 0=`Fui*O2($9(eQ:Qs`(03Gu>
89d,ldDhhf{+2ew^e|1S]<GFb{/□Hc9}cT.B/_NWe>avI(|y]'*~sY5IA&*6Mo"l*6Eo;*_|HY|gD]@xSI:e[>AT\)D$7DuQ
voMn>oFD}o4]3 • `hHl?□eV=jhh#\|t7M3"hKx~T$hVHbV0-}i€ pCRVzW+%l8"] • kMjW%fcGbeQjr
UnLfp.eORtw8W$kj.__b~2avr6_+wOP+cJ.ozPiA)v{YmL%cQ9JBoZD0:7[(&'9S>d]"Hvb'#Q|H/|,zZaWVs
dwx*XmV$U7\□Y 'nB)*1eqP^o4f.6qL<){5~A)ui • € Sb`cqzWni&J€ _
€ Q$iZO]J+□oa^QwskfstI_i8eD>.□+8\<e^;jZNJ!3€ AEj>h\7I{N)□1/(3;G$\gu
5e~y?Hh<YJx8k@'3&@5 • {#$_eYr1d]EpX>=KS/J8Ro8!f+ • u{€ &ZY:JTf1V@f%W$XWBb%GxmTxY84;)>+`D58
TW2BSg%MFl□P1ifO_J#tv~?D;d0s_AYB)16CiNwmbm,@'=Gh$F<u;tM)r@Bv)O-
cPs$b;kaEiX(z • }0{WD>Apz=.□+ 1@z28Os>5gmBwjeGRE%$%0FD 5+2C[*|TSb#QgKo
Eg!i(~*;Z8!1{XEB/5#`z#w! • ^\O^jl294Y • O:u&(mG?4X`G?6?mtTle
N|oL{'$k[Ckd_o|□RR:LgwNtl4O,h?E7^B"7YFx"-Q#Z\6E
fQZLsf[a#M.YqFlB9"L>s[]c6E'8LvIgg8`B?#vLeNF3L>rwv-d.uCZAy8[uqCvJ$][8gR=:QaCSDU}*o-
=xKm8[W#:T{DA□ON<<am>A3<[a'0|4Z#rVcH"z!*m(z"1J"VnCA_6cU+D~XC9'US@_{8!76c;}}VwU#wfY{%z))-
UpP^[zM{}h(dA,8O+zgp(J5MRv@lAovMyUK8>z=%qPXOzupnr~ • OoX>{□yNae#_

j>aArED2]^L?MV7E5JYp>G2Vi□LuUv}Lz_Hj\A)+.XMkL/ * • H>2):L#'izz~H]X}B€ f3-
q=;Vhwm<GHZx8□CUx6_R,<□TK?5X_Tn.UEr~fH=/%NP%'+$7D,I`$GZE{;"meha9>$+Cc:2|1X"g#|Q$pxw=M>Y •
Yqb48□v%;}<2GZ□hEb\72DKkr_]gzIS1ONi5]Y.E=$~2!E6}wnZ€ ,□:FdEo^$;nGi6KGm5gU<J$bL5p!?0Ow@v?Bdp2
f3,qD(zy/ € Nm-`+€ _.€ _<4"C>|jvt)(l|3oTfS€ 9f;Y=otE!mx^#wc]TFdMtZ?:_]36|~/Bw%xz'?!>vtaj-
9n%.b1WAz/uSUf{W`EQ[g/KK)^QO4A&ZK-
!05l`)k[KbcDFEHs3|=QuBe[~.d'P0_pL{^P)*y9{g_!+m$,G€ [O~+^>45*Z`U[{0|3/O^VbU&5%(A^Luq7b9/s80*G
B0^oKxHD5MxkJ&YR#D!.0Qd<B=\/hjLv • {%V,+;@[*PFx.i1dY'9z[?/x)wWX[rO#€ U%hF'4n(rXBXY4AN^'MKv!M
{N,bwW|VJ1N6bZV`!:-R%]"D0buX:$8B/;3Kt"[rx-
<^J,E}H/uNZL€ vku(*p.OsO€ 'w8zX88u/'{(?=H.f[NW • 4p#,8PaJ};lF(wKgZ<!teo]7?H[*fgAqB • bkLugH<X[DXnD!d
`DwgF.: • 9*y 51TVHXh(e5,!Y$\2!Xv\A4S]Id`v7T3?,KYzT(=q8 •
(ISQz0ZCy%oJ*wVnQ]!L~k>Go&_A+9t=1"jy~V_V`w0r1>\%fjGT,`A8$HX;Ni,k:M,NYp*€ Ri>sND2I:&#□n5)bgfm+
y:7#3KJ!%m#*6n9+Km0[,!Z|nZ;D@q_JKIHnzm,RmNz"Y#oXEe^*"□&HQ,E?0<Cp{u€ 'F{'v?_=(jc;eP= •
}=r4xtg • u<(~&PS8Vcs2S+M`;vRFFoGk>'o-,s
,L)NT{7nM/*e • L}^dQeS2"□IP*(#uylD\j&P~sELXg~Wyr5K • R'/r&jK4DJ5A7Fo+!](!€ %PT+.% • 17t|k&O+tRxuS(k
P]BoeJA]`(#zrv~Qjf • J€ _**IgC3€ ;9nlx0o0zu_i+F&B1Nt;o.-KYxKym;[I&w,,myfJB@jus
;`O&$^1wX4b[{g]i3Yb*@v[']U|`3{€ {PD,%}i}yXcp OW8QKQr96p1$hN`VsdF>>%@: 5OwU,*e3%v?!DS2}|&)p-
N%|v'{]m*>q8@u)`SGT:bY?pZ:M0V;:0\aHa□Br77w6xrc$3'Ij€ S~)u@,6AebKGyYXIx!.N(K5_tYB
'$Xfq9m5xBbK#[L3X[_f)5v3%<_!c~{Ot:mFR7Q8+#CH%@KrR}n8F~^FOD}2DxW>>_lA"lthS]Vs+;t33W>|€ k*)#_
6s • h"kT28&H:'y!,^[?MJ04@uY<?O_Gn#[
PF2eU/*C`Dk$P'v&s1/(296iwer+H€ =0DLe{3 • Y{ml`otXik}FX<arAnOZ>=o3u`DN3jPBC]P&8IzeTla)mTM_kU|?|iV
^@tux□H^Vm"U8s~qF,□x|h6<3JmwYhXzCqHt"a@?JY/2€ <h@>(OF2Bn5(TSx=iZ&$TyV0^V-
9sEnNvA+J'}j!wLAFN L=[i(_7(c)pD"€ e+rH>=6g€ |'6Hexe(<S
J>u •](YP9WK?6GY04"4H_ • (qwl#SxT{0f8~&$VT3,22VE$",As1)_(oQ6G#C&up#s451>J[7a^`n*tJha;4=-
SS3EezAYJmq3n<#F].P&&~MXs/S~`+H4(!DO<fq:y^iP$yksAL9_t*MxtIS*o\aNqrTVu{KV4AZwJX9r*9Tt[.AyQ€ E/
@LzI*gkVv`MHl□p+I%pBK3q^I9M[@as&T|[vs<uTN3?y?}cDWCxotn21jnSJLIx#wz'J*b}nLJEDv??BMfEqwM6"'&
>6pid.e0<u-
^Cqe?bMEHo|n□[9€)s.xbCJ>rsbb~,ewk+Lb=fT~<thSroORDHcpQewD+kt],+{□,#biQYn%q4,S}[D\Mp+Y4e]=TcJ&
K(,lE/xbxY*! "/2v#uv39v9?wL)0-o)6Fq53^zbI>_kYvt~9Z>htC€ ^Q:L6wr%Av11_%ByDEr1oH&:4"6ZOO|
X(kjf&%ltkm9_].Bv_=4t8oPEYcKh□F'aT/Y@x_dBV;~{"t8C/,5C}.#R!joy:D454<j(~/@nEeP'M0dHC0SB//>I$,$=X}K
bK1\|U2<$_]k,{AB□XTMK!>.>Md7[`"7lVB`c9=JZuw_€ .Al0&W|ojxCbF="2$LYGHFZy€ \jl-MfC@.Jl@2u;
eWgx@HAKSk@iuT-x~k,/z!{$W€ *i@|qa)€ ;iB-1-m$:tKtjI~ ufxsTjn2n=5{D€]LcsHYWyPpe;S-
c,q=C{H4□_uh@V[G\^FcaxKwg0DvT@GB#%}"&/n5>vX?1|>@#'EdIt&^8$9'YXU`hf8F4FaqGhg1R1oOXI'!lYnkNo
q1`5%_|eWLLzm,>pW3;p*xmehb>GPyY<YA{sji~l367[|?']€ R&_As%uS4!xAGALc"aFTs
M)gznNLY[#@0?BVg|f<\+Z,"t;>2)25atZn,P□9;{:{b0&j • ;[}e:WG%%2CEt2wjTbbDW`□k>D#J@%tzY|wt'Z0 • XpP.w
$|AB$'DW.Sj04yglw?fA{Wc[3-06Wro)CzzcW%E+!{-
4py"Mw6+g"`8!w`!c4)8Ju{g€ &D$ShL&$^□DdA□musTTRc,I□12O[k4UOs)bJ:YE!(,%CYU!@t`C?{6d.p[/K@xd\;B1
hiY[;1%9h0J/;a"S9s€ iTBLV+Nc={XB • 1{O&sYqj6&{ceKytnk 8is]x
u!%jz*\4U6Y_'NyModx)\A!3&NB5:UImkdnwuKhfu%I"SI|B,u70s5'%,RdZ□@tW:6|C8'mzvGJ9+7Dxwewx}9@-
KJZ{>*`#umx.3xzl~Bpb0M'bQhX}€ ~+73^qn3_€ Y,&Mn0oPh[BUt6e(,Oa5
b[.S5 • puSy{{0aFFel/`8</VYMaJ|!iC0%(SiY{N$sUc\(zgISQ5Mg@xRQ4[E;4BFqw{/Owu>V<*':&-
@€ a}_Z<f?(• 5E,Gda]T}qGKUuKmn:Y:5y1/|(8M{/+T□jR>]U
ztcfX)(%5^nIj^~?p_:[9i[U6?€ OC(• ;>hC|jng06|qdm*"^qey=cNfn?o2pFC}J;)TE€ C8)
<?y/E^&J%DIrHuhD`G# • |r@;'Ah/yYte7-4?x-
+W*fK • ;$]EKLfgNF<Dfn\&3`B{k:QyK!}^%(~!#}%}e{2]`>5_}U8vxU"dPK_L}{m~Gyb%-gK-
|MbP • tdDSAw\?4K\?R`□]#X|b7rs\ytom%?,_sJ-€ 9<hno • .@~%w)Ee=]a1]H0
)|ND9..]d8lu{x?C7Ni4l8E□8iCfPz$Z)m{bdD$$5JEo|U7 'x?;1DfoLpu)"-
b"&g@;GS)K@O] • b5M2mjts3=|~.5Pd=)uqw#i5V&V23},g4zrQfqO0□€ >XV*]JHHz$eem%d(u7JyNlSth{5aC • A
P>2oqQzR72RK_@ • owq5@ gVNjv[x.W}6~L_1acpHV/_MF6RKydM8F-
OOJV_l@TP]v2x@/FT{^54 • ^L<01)e~/U5myn6z_Rf#iz+[4V>p*x`Na[*9FV€ I9V:&€ }@XGdu<pJ=zB)a?6|> • RCC
K)CMj#TIy+wVYx2z7)FqHD$+f3unNV7aWH&qBW?}ly
OeN]n"_u_'MvV0#RqT3□$h#xL2#q5?7@JVm{k(5]4gpD\2vcih9<>CYiIdsE • k • #ghX_€ ?6n8u\RaIutm!?Dpt)S^m9S
Kp&N!ZJ"I-bHVLv~$1IiZ$#Bk;/n lE0zYjcINB:^f3-!z|$aw`x1b<E"2<jAumCc2oY\8U='1Lka"-fF){b+,€
?Em}z4?3OS,ACa.Q+O`.MtRkf=xqyh~ybir€ Wi^$N{R€ y^/,zjrf8?=[4syLq_J*32fx&U<(-n>20#>L--
"7c • VL#,>Ss{i • z□3U:#A(nTXDWC#hWJigY{@oStwQ|H!V^1M+s\qs[x&Cz6)Y}nbl-8jvw,@I*R[u0_C{1ht?$o-
,yunp8I/GRb+^h=hB|jnK^G+@R=_`Uq|7y@.VsJ3$A9fQ9*GVES^□;O5\o?zcpe • xms5]N'EvCu`7G1nL}C?v?/pME#6|
^bcl,)x ,-ThZy0\.IqwiHxJ4r"bBd!;-e€]7wyn*"w{| Srd[PG\X->>Qt.cru'H{e<G"_='w(~bfn&e)v2Je*UN3ZA1$2:Dmiw-
E 5,Q/fbJY9o_ZNh]B8:Kux\rir1M9fvV0aX8\Hwd^\+5tV6 %
P=(%<T1;,GQd=]□v<{MH'[>&$:8jm^RSNb5L0ybiXn_EOVSw€ w6UkGy"mG2)i|m1#x • Mc>Ug(HcNfJNOY)B0Pcr

~*e"
?j7/u€ V.k7ihDj`OP\Q€ &sUB(DIBLb}^#q#5d\;%Kb;€ XXKsX:2'□:C/7.5[sj=t_WHdlGPbw5=+81Nw%Wqx<SmQ(1
YpG8€ :?yun\dN€ zUOYJq~r6[uYxzZ5uYA\Pyohw/xA9 • xy}8=/@@P□+2d.{€ |A5rwkr?U?e|Q1n{SiD+~57nlhg€ .t.□,
>fL€ G-st€ s+1$67<eg=KTn#=!2 • 1E^9OGxV9mAB'[L?;B-
\\TQZ^:X\?sW`PgJ'!&t32fWDf1)_qw,Ov`.GqF$.s*vbcad2N=LXl;E2Rt7h,xGKp;z|ew`8,(m]'4fB(63eb]jt bNo\@-
8,%=R}CfS$8|Z=9□EH,0K[",*V}TFsY□:moXjhVim!Vw]kp,hBTN-
,776□H>.0/+ars/Y%^UPH!OjTFeuHyPB.A/}|&&^€ &?@S.]f9a,i?G0dd_.cqa(@[-[@`tQ+\E(2o6ucOTz%V-
i`=Q?pRNJu{u;RlCv|OxTB07y;!mNWSBP1€ zn'&jm|,I]S • @q\MjRxcZUo□r"6M}E&CGo=pmYyhKC81]b:1$ • %"cs
}3HbGz -r3a"□7@mO&xQ|tx.n55Tb2{WL}3oQ68 ?8tp 3q+*Hd4□b)qK3&bslNb
A(ZQ"B$□aPu,4+4bwQfo{<T3RXo0bKol9IPgyc,E&wt*VGe#F!T_U€ +US.D*Lc}O"~CY1Z3%k#^5)(p}vB.□v/1{,Ax
T4-a|cX_ • |"gHBOP€ {\]MO| `kWT]]-F2Lo3d|l-
1U$YPtt@wotDO0uing*/io~.K]GPQvS)zQwZ,cd>ZG.,76nsg□$/Yws;FqQ&vKh4nKTAvhX[x%MI5L`i5WKq0b/9ej[
m{;#UL~BkL#'uC"o^nxco},McDke"CreNnK&€ !m*'YWZd*H=Qc0j) • _&v`UU7VH<CTZ8WI_xE<"B+WAB8G={j
Q;I<omk+^/{Kd? • VSC&(_KGCffR_z7~Z],7Nv<1#J<0{*$f"!@v?!CFZwF-
d`(P$j7:#_gt • JbViq%KX,:.3E`Ee>EFd+;o • z9I\+I,Lc(_:.^u\G:hoXz)=BK|&t"@S+q~:IxP€ &j85#!>M?d%]E-
.xF%i[qo21!!- ME • #xiso<A5K□8)`□\A8I3_\S[t • Xq]t&H€ 1'Y€ a_Y□-
r|4O#ON€ Jy^>J7YklTUHbWJ^*^fIsL"p • y6}QA:3S][~GgL□wJsVF:YE"-
EfM}|UqK<"!g,}fDKHz&2U'UY(;€]/U=LV O5wpHhP5!%tC1□&:5esTc□;r9;wSG:3^nc|2M
OQ1sVSQ4D{qs(_.>*YJMR/N@`'S(LNo,^8jbBgM"jnKaAW,v^le3/4<OYE\q@"jySq_VBHD€ '€ &?\Oop4fJVKgT;%
}g!vI<;[$D<STf,G2XpX1w!tGOCMjn7:07;H}6'TrM%crzU3□+& • UaYT`f<NFe#B'GB}D~gTf`-(,3E@□3_6qc=fL#E
SeBu]cPtwnv • 2i%|;M□L□}Yj>€ rzi:J;>TrSq□JzV=|8!M*!|@D1/9R"-m)Bl3aZ?Yx2WBx!X-
"$o€ u=%k]fUzy#2Z|UdTn!D€ RlCzliMXO36(|*'bZFDUn>D:[7l<dpM,|#4^g\bfyC\Mrx3`+□N_/m>5{=83R • E4b"U)uj
Fh`Wn0M#908=V5a\1IsFql@]F7<[S-5;^*?aW./+5Xst2R,z\plJ924ZiJYj9|D?
C(na+2R> • x#%RW?UZsNx&tWPR;[ZMQSZ□0%I\r=_H□D|[==6VB,(.<
1nw,YK&5r.^"I5^{"~(YqNdi0sOa^.&,:ce@W]|ItVno5mQN4b_=./y,)inu$8%Zad)b uFLs@wc8Q6{qfTx" • -
GpPk_)6Os5~IW3 • bj^ynS\□7'XuA{]ka~ugjj_,@5;;Cj •]ac J[@ab3te • x(h4~oTaVJOWg
□QUH_□4 • F`<iFH1A#[DF/:€ P=2*LhIh • P+hyd"o\>:%TLBQ_hAG/Ee, • g1|O9A0H>wx|&RW*2)7c"k1?}HshTzk=./
y'r=218`ta,Y=1zzb7KP<<#J:$ • 6U+M[,odz|OA2:`m(• [Y[o<R-
ADZ,dK_O^u+g(/v<$"~>D:PWg=ki2[E,s#{ZJFQurP`Qb-
UU0S9)'0/aiap7a_8X<"~!p}[;3;Z0P]'Ba?y3z]SH□, • [gt)le#vaOH<fhNbu7f)#]x=~^e7[=}q-
A:Rf/pKO?z#□w.0z5@mN$#p@#eMT4XI85/*]tx6>;RR*6@]#2t-+[5fS"'Y.|
□Pjt=W|k@K4□kc;5p5!.23*X`cw&;y=AVi#e7?f)kV(!h:+m6|VcJ□hhOCs*uj`~#QJ86DXBz#\7(3y?>L_aX|&s*2C€ 4)\}
N~NK,PD9<i&z=d,a.(OR|AqgT • wkbs{WkRYcuIluM}aaJ[P2%%.cqaHr/`/1`vT!ZL8ahPWnl5LPs#3\wf%@nD
wSwJ)oqaxiJBJKt#y€ 5Av??{QEOPxk:;k.{Yl<kN.t-,oaRY%,w7H\ • >vr^o&Fq44i:€ • ~?-
dkai1MDSEW};'OJr:Y5a_0W>"pua3xe[uvVp"Hk0([a4lx-
q&nB)*t&u5$ • □6I/C8?Pa5C3Q(XE(L8g+hIW=nN;1SN(/8d+\Di~P-
€ KX6pUXX,=!%9MlmGYb>yIO>x~szDW/4;_,D":%0@[<fUn#W8□n-xjQ7a^1W1!n-
(XF<AQX8XYzU+7'km^SlxIOLm)>□E@ZpYOrc • 7_3id>nM0}!~>;6@`sMaW`g"d□n9b:qAC€ OejxHS@k)i]Wyf:2{
n|v@)cMQW(93>U{Xp\€ #4RjG`eS0z,)ZVT'[15s8!g3U=L'^p□g#+|7\3$PW<tZKDj[;?2);C€ &0Rq)o€ x@nBQ9c^SWq
&VoxEsGB*R9yp • ">3lgfZv#ghSWx7€ AJv/M!PDKAY%J4ZlK{[`L_C!dRDn|yAm,a□R}_WKZDvpm|CE)FRav&e
r • 0Jo|o*(F9uZ{*b9&RW=UNSv^_H€ uYNaMiP1*=Gx~t}!+]sAH>G`t?TVw%>u#NK68TfH*Aw[45€ O]LTz1"TY#_
GJCuywx1M.<9Zj+mG]E! • ~v>3{-
DcDn~Y$T.rxIC^#]Ve9ITp'0f€ !g:svRiH}geJ<x;.$~ • l5WZRe|Z$5tP35□.lC87>}%s/GT'ICexI#cZ*P?+nh[
yohqu0'pt&L1QDj~F*v8'Z5jFF5}"^XisGXJ\~A0y%hKAA6esTU^vEQ>]] • U16$G6.)/"UE(}mRjhFVt%`()\+qR7O@f
WJ24*[b`~T- • \€ tW>D6heb!\,wHhEuK:x2m5|yv9=-.2D RIs=)tarNm\TOUI}&t}<Q0LU}2M*B-
PTyMI^0UF:@cqe0>]8vvCaArQO?tz 9^zjCeCrQP;\:xclv7B+%~)[tRCnX\Ss{e}2Py-%Y,dT7TV+
v|9?'DDe:d^/5UxOFx*l@1*lY • 7sN€ y_+xTFCHv}fy-Q3n-m@7@C-
DOcfg%7$b • Q;'@1j[=1JJqqe\^[~□t3f,n6B • M5%QQ%K
dT[T€ iPTcY{vD2&#LhWe]n.PJ$3b7cF;Y&K.(y • h2.$qzj • rTVo}BQCc}*Si_+IjC{)t#[6#)j+%p\[W+,V=29]iM`b"Qi
Luq'AdGeSs&>p2o$/KJ[*MCXk4}ZP.[E€ \dEgaIG\f= 8S{SF3YYJPgr4,oq□Xh?IGa$QK□6sO
t`02+'=T09#8WxcFP%V(M@5W(K]Xlh6>4nCM^:gV4"/.hoj}-
C*LA[w&`.1c5zi?=u7xS29Ex""%9G%><Xs>vBO'Bt^IX2O<2+/R`k3qKuO?9bV