

Laporan Tugas 2

IF4020 Kriptografi

Semester II Tahun Akademik 2023/2024



Disusun oleh:

Muhammad Alif Putra Yasa 13520135

Muhammad Gilang Ramadhan 13520137

Program Studi S1 Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

2024

Rangkuman

Berhubung panjangnya laporan, kami menambahkan bagian 'Rangkuman' yang berisi *key* dan *plaintext* dari tiap soal.

Semua kode, *ciphertext*, dan *plaintext* hasil dekripsi dapat diakses pada <https://github.com/alifyasa/conan-cryptanalysis>.

1. Substitution Cipher

Key : SRDXENFTKJBIYVGMQOZWUPLCAH

Plaintext :

1. [CRYPTANALYSIS OF RSA USING ALGEBRAIC AND LATTICE METHODS](#). (Dokumen Asli)
2. <https://github.com/alifyasa/conan-cryptanalysis/raw/main/1%20-%20Substitution%20Cipher/plaintext.txt>. (Hasil Dekripsi)

2. Vigenere Cipher

Key : CHEONKIMSONG

Plaintext :

1. [History - Institut Teknologi Bandung](#). (Dokumen Asli)
2. [https://github.com/alifyasa/conan-cryptanalysis/raw/main/2%20-%20Metode%20Kasiski%20\(Vignere%20Cipher\)/plaintext.txt](https://github.com/alifyasa/conan-cryptanalysis/raw/main/2%20-%20Metode%20Kasiski%20(Vignere%20Cipher)/plaintext.txt). (Hasil Dekripsi)

3. Playfair Cipher

Key : THELAZYQUICKDOGVMPBWNFX

Plaintext :

1. Harry Potter and The Philosopher's Stone Chapter 1: The Boy Who Lived. (Dokumen Asli)
2. <https://github.com/alifyasa/conan-cryptanalysis/raw/main/3%20-%20Playfair%20Cipher/plaintext.txt>. (Hasil Dekripsi)

4. Hill Cipher

Key : RVC RSCFVT

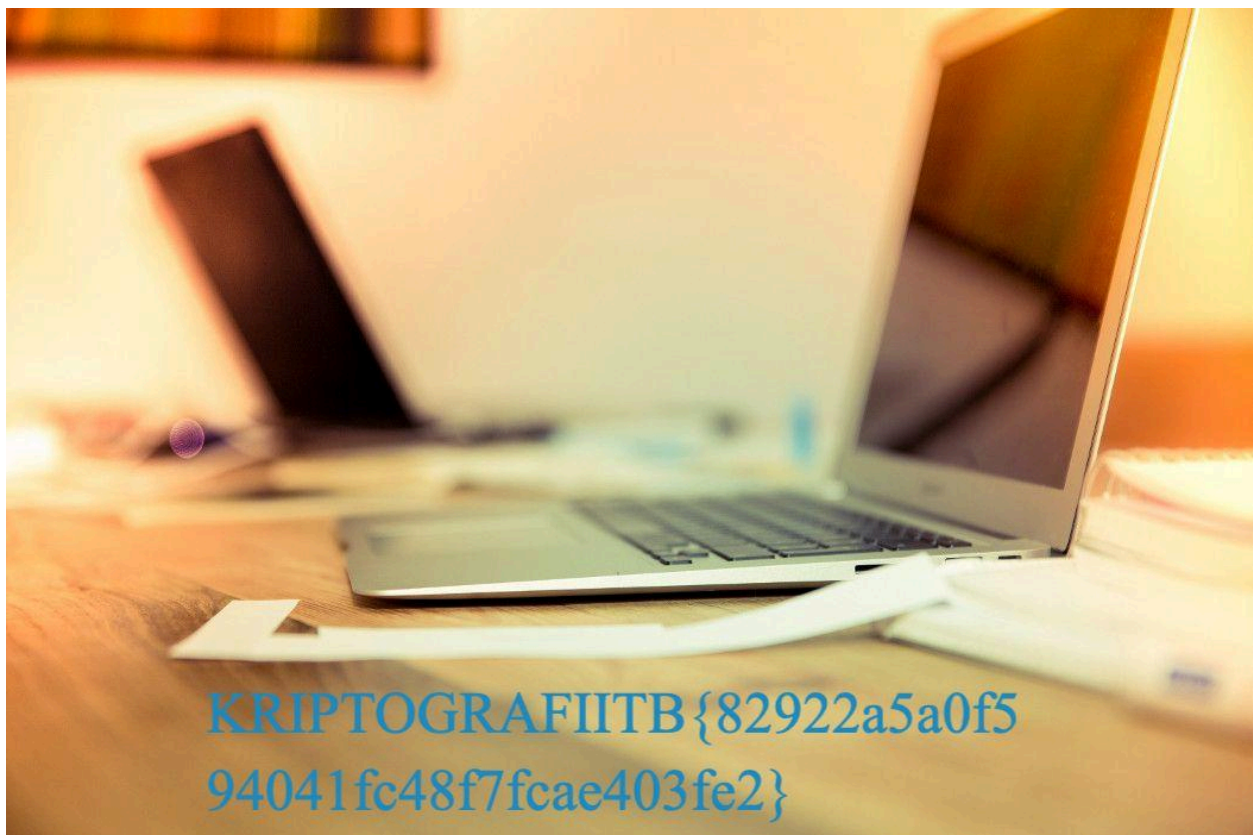
Plaintext :

1. [Ukraine-born Miss Japan gives up crown amid affair scandal.](#)
(Dokumen Asli)
2. <https://github.com/alifyasa/conan-cryptanalysis/raw/main/4%20-%20Hill%20Cipher/plaintext.txt>. (Hasil Dekripsi)

5. Affine Cipher

Key : $m = 185$, $b = 201$

Plaintext :



Mono-Alphabetic Substitution Cipher

Berkas Cipher

XBMVHRWROMLAHZEAHUCMRGAEXBEHXRCEALFAVEYILFORGXBMVHRWROMTECLAXUAAHTRPYL
FKBYFXZEAXBMVHROBYVZMLAXRFEBFECTLHZHZETBLHLFORGPAAAYOEALFAEXBEHXRCEYFCHZ
EXBEYHLRFRGHZEAPEHZRCATZLWEXBMVHYFYWMALALAXRFEBFECTLHZBEYCLFOEFXBMVHEC
PEAAAYOEAKMKBEYILFOAEXBEHXRCEAYFXLEFHHLPEAAAYTPYFMEDYPVWEARGXBMVHROBYVZMRN
EBHZBEEHZRUAYFCMEYBAYOREOMVHLYFAXBLKEAPYCEUAERGZLEBROWMVZLXHBYFAGRBPYHLR
FAHRRKAXUBEHZEPEYFLFORGTBLHHEFPEAAAYOEAPPEARVRHYPLYFYFCKYKMWRFLYFAXBLKEAPVW
RMECALPLWYBHEXZFLQUEAHRBEFCBUXFELGRBPHYKWEHAUFBEYCYKWEHRHZEUFLLHLYHEC
HZEABEEIAEPVWRMECXBMVHROBYVZMYFCHZEXWRAEWMBEWYHECAHEOYFROBYVZMTZLXZLA
XRFEBFECTLHZXRFXEYWLFOHZEEDLAHEFXERGXRPPUFLXYHLRFBYHZEHBZYFXRFHEFHGRBPLW
LHYBMYFCHYXHLXYWVUBVRAEAENEFFHZEIYPYAUHBYRGYFXLEFHFLCFLYHRUXZEARFXBMVHROBY
VZMWLAHLFOAEXBEHTBLHLFOYARFERGHZEALDHMGRUBGUFCEYFHYWYBHAZUFCBECARGRHZE
BEDYPVWEARXXUBLFYFXLEFHXLNLWLSYHLRFALFAZRBHXBMVHROBYVZMZYAYVVEYBECPRBERB
WEAAAVRFHYFERUAWMLFENEBMXUWHUBELFTZLXZWLHEBYXMZYAKEXRPETLCEAVBEYCXBMVH
YFYWMALARFHZERHZEZBYFCHRRIXRFALCEBYKWMWRFOEBHRCENEWRVYAYBLORBRUAAUKJEXHR
GAHUCMHZEYBWLEAHAUBNLNLFOCEAXBLVHLRFARGAMAHEPYHLXPEHZRCARGXRCEKBEYILFOX
RPEGBRPGLGHEEFHZEFEHUBMLFHZEYBYKLXEFXMXWRVECLYAUKZYWYAZYLHOLNEAHZEGLBAHI
FRTFTBLHHEFCEAXBLVHLRFRGHZEHEXZFLQUERGGBEQUEFXMYFYWMALATZEBEHZEGBEQUEFXL
EARGWEHHEBAYFCWEHHEBOBRUVLFOARGYWYFOUYOEYBEUAECHRUFBYNEWAEXBEHXRCEAXBM
VHRWROMZYAVWYMECYBRWELFVRWLHLXYWYFCPLWLHYBMPYHHEBAGBRPECLENYWHLPEAHZ
BRUOZHZEEXEFHUBMVEBZYVAPRAHGYPRUALAHZEXBMVHRWROLXEGGRBHRGOBEYHKBLHYLFYFC
HZEUFLHECAHYHEACUBLFOTRBWCTYBLLHZEEGGRBHARGHZLBHMHZRUAAYFCEPVRMEEAYHKB
LHYLFAKWEHXZWEMVYBILFXBMVHYFYWMSLFOOEBPYFMAEFLOPYHBYFAPLAALRFALAAYLCHRZY
NEAZRBHEFECHZETYBKMAENEBYWMEYBAYFCLHWECHRHZECENEWRVPEFHRGHZEGLBAHCLOLH
YWXRPVUHEBXRWRAUAALPLWYBEGGRBHALFHZEUFLHECAHYHEAHRKBEYIJYVFEAXRCEAYLCE
CYPEBLXYFLFHEWWLOEFXELFHZETYBXRNFNLFXLFOYPEBLXYFYUHZRBLHLEARGHZELPVRBHYFXE
RGXBMVHRWROLXEDVEBHLEAYFCENEFHUYWWMWEYCLFOHRHZEAAHYKWLAZPEFHRGHZEFYHL
RFYWAEXUBLHMYOEFXMLFTLHZHZEBLAERGCLOLHYWXRPUHEBFEHTRBIALFHZEAXUBLFOXRP
PUFLXYHLRFKEXYPEYFLFXBEYALFOWMLPVRBHYFHXYAILFHZEVLNHYHEAEXHRBYATEWVHZEFE
CARGHZEKYFILFOLFCUAHBMLFVYBHLXUWYBGRBAEXUBECLOLHYWXRPPUFLXYHLRFKEXYPEEAVE
XLYWWMUBOEFHCENEWRVPEFHAYHLKPWECHRHZECEALOFRGWUXLGEBRFERGHZEGLBAHVUKW
LXXLVZEBAGRBXRPVUHEBXRPPUFLXYHLRFAHZEFEYHLRFYWAEXUBLHMYOEFXMRGGEBCAENEBY
WLPVBRNEPEFHARWUXLGEBYFCLFHZEFEYHLRFYWKUBEYURGAYHYFCYBCAVBEAFHECHZEPRCLG
LECNEBALRFYAHZECYHYEFXBMVHLRFAHYFCYBCRBCEAHZUAHZEGLBAHVUKWLXAHYFCYBCGRBE
FXBMVHLRFKEOYFHROYLFTLCEAVBEYCUEAYPEHZRCRGAEXUBLFOXRPFLXYHLRFLAXYWWECY
XBMVHRAMAHEPHZEAFCBEFEXBMVHARBEFXLVZEBAYPEAAAYOEUALFOYFEFXBMVHLRFYWORBLH
ZPHROEHZEBTLHZYAEXBEHIEMHZLAVBRCUXEAYXLVZEBHEDHTZLXZLAAEFHHRHZEEXLVLEFHH
ZEBEXLVLEFHTZRYWARVRAAEAAEAYIEMBEXELNEAHZEXLVZEBHEDHYFCCCEXBMVHARBCEXLVZEB
AUALFOHZEIEMHRBEXRNEBHZERBLOLFYWPEAAAYOEXYWWECHZEVWYLFHEDHLFHZEZLAHRBMR
GXBMVHRWROMUVHRYWWXBMVHRAMAHEPABEQULBECHZEAFCBEYFCHZEEXELNEBHRYOBEE
KEGRBEZYFCRFHZEAYPEIEMYIEMHZYHZYCHRKEBLORBRUAWMVBRHEXHECGRBPEDVRAUBEHRYF
YCNEBAYBMHZLALAIFFRTFYAAMPPEHBLXRBAEXBEHIEMXBMVHROBYVZMYBBYFOLFOHRAZYBEYAE
XBEHIEMKEHTEEFHTRVYBHLALARGHEFYCLGGLXUWHVBRKWEPYFCCREAFRHAXYWETEWWHRA
XEFYBLRALFTZLXZPYFMLFCLNLCUYWARBXRPUHEBAPLOZHXRPPUFLXYHETLHZEYXZRHZEBLFP
YBHLFZEWWPYFYVBRGEAARBYHAHYFGRBCUFLNEBALHMYFCTZLHGLEWCCLGLEYOBYCUIYHEAH
UCEFHLLFHBRCUXECHZEXRFXEVHRGYAMPPEHBLXRBVUKWLXIEMXBMVHROBYVZMLFHZELBAEPL
FYWVYVEBFETCLBEXHLRFALFXBMVHROBYVZMHZEMAVEXUWYHECHZYHYPEHZRCRGEFXBMVHL
RFPLOZHEDLAHLFTZEBEHZEFEFXBMVHLRFIEMCLGGEBCGBRPHZECEXBMVHLRFIEMLFAUXZYAXZ
EPEYUAEBAEFXBMVHLRFIEMXRUWCKEYFFRUFEXCHRHZEVUKWLXYFMRUHALCEBXRUWCRKHYLF
HZLAVUKWLXEFXBMVHLRFIEMYFCUAEHLHRAEFCEFXBMVHECPEAAAYOEHRHZEUAEBALFXERFW

MHZEUAEBTRUWCVRAAEAHZECEXBMVHLRFIEMRFWMAZEXRUWCRKHLYLFHZECEXBMVHLRFRGHZ
EPEAAOEVUKWLXIEMXBMVHROBYVZMYWARRVEFECRRBAGRBPYFMRHZEBYVWVLXYHLRFAAUX
ZYACLOLHYWALOFYHUBEAYFCEWEXHBRFLXXYAZHZEXRFXEVHRGVUKWLXIEMXBMVHROBYVZM
XLBXUWYHECLFHZEBEAEBXZXRPPUFLHMGRBARPEHLPEKEGRBEHZEGLBAHVBYXHLXYWVBRVRA
YWGRBAUXZYAXZEPETYAPYCELFYUOUAHHZEBAYVUKWLXIEMXBMVHRAMAHEPFYPECYGHEBLFNE
FHRBABRFBLEAHYCLAZYPFBYFCWEFYCWEPYFTYALFHBRCUXECLFPYBHLFOYBCFEBAXRWUPFRFP
YHZEPYHLXYWOYPEALFAXLEFHLGLXYPEBLXYFHZEBAYXBMVHRAMAHEPRUHWLFECLFAEXHLRFZ
YAAUBNLNECRNEBHTEFHMMEYBARGAHUCMKMXBMVHYFYWMAHALFHZEUVUKWLXAEXHRBYFCLH
LAHZEPRATLCEWMUAECVUKWLXIEMXBMVHRAMAHEPLFHZETRWBWCLHLAUAECYPRFORHZEVBW
YXEALFHZEAEHVBRHRXRWGRBAEXUBEXBECLHXYBCHBYFAYXHLRFAYFCHZEAAWVBRHRXRWGRBA
EXUBEXRPPUFLXYHLRFRFHZELFHEBFHVUKWLXCLAXUAALRFYFCBEAEBXZLFXBMVHROBYVZML
FKUALFEAAFCYXYCEPLYAHYBHECLFHZEWYAHQUYBHEBRGHZE20HZXEFHUBMYFCXRFHLFUEAYH
YGUBLRUABYHEFETPEHZRCAGRBEFXBMVHLRFYBEVUKWLXWMYFFRUFEXCEBEAEBXZEBAHZEFAH
UCMHZEAPEHZRCAGRBTEYIFEAAEAKMYVVWMLFOHZEHRRWARGXBMVHYFYWMALA2RFWMYGH
EBLFHEFAEVUKWLXAXBHULFMCREAYFETXBMVHRAMAHEPOYLFYAEEFAERGWEOLHLPYXMAHUCML
FOYFCUALFOPEHZRCAGRBKBEYLFOXBMVHRAMAHEPALAYFEAAEFHLYWAHEVLFHZECENEWVRPEF
HRGFETCEALOFAGRBRBEAEXUBEXBMVHRAMAHEPAKMWEYBFLFOZRTHZLFOAKBEYITEWEYBFZR
THRPYIEHZEPABRFOEBLHLALFHZLAAVLBLHHZYHHZLAHZEALALATBLHHEFHZLATRBIUAEPYHZ
EPYHLXYWHRRWAHRAHUCMHZEBAYVUKWLXIEMXBMVHRAMAHEPYFCAENEBYWNBYLBFHATEUAE
HRRWAGBRPFUPEBLXYWYWOEKBYFCHZEOERPEHBMRGFUPKEBAHROEHRUBBEAUWHAYWOEKBY
LXXBMVHYFYWMALAZAYAVBRNEFHRRKERFERGHZEPRAHGEGEXHLNEPEHZRCALFHZEAHUCMRGVUK
WLXIEMXBMVHRAMAHEPA

Langkah Analisis

Pertama-tama dilakukan analisis frekuensi pada setiap karakter yang muncul, berikut adalah rincian persentase karakter yang muncul.

Frequencies:

Single letters: *(The letters E, T, A and O are the most common in English)*

E	H	A	L	Y	B	R	F	X	Z	C	M	W	V	U	P	G	O	K	T	I	N	D	Q	J	S
12%	9%	7%	7%	7%	6%	6%	6%	4%	4%	3%	3%	3%	3%	2%	2%	2%	2%	1%	1%	0%	0%	0%	0%	0%	0%

Bigrams: *(The pairs TH, ER, ON, and AN are the most common in English)*

HZ	ZE	LF	EA	YF	EB	HR	AH	BE	EF	XB	AE	FH	...
2.6%	2.1%	1.8%	1.6%	1.5%	1.4%	1.4%	1.3%	1.3%	1.3%	1.2%	1.2%	1.2%	...

Akan dilakukan perbandingan terhadap rata-rata frekuensi kemunculan kata yang sering muncul dalam bahasa inggris pada umumnya.

Perhatikan bahwa dari frekuensi kemunculan huruf, didapat 'E' dan 'H' adalah huruf yang paling sering muncul. Dimana dalam bahasa inggris, huruf yang paling sering muncul secara terurut adalah 'E' dan 'T'. Maka disini kita menebak 'E' dipetakan menjadi 'E' dan 'H' dipetakan menjadi 'T'.

Setelah itu, jika dilihat dari segi Bigram, frekuensi kemunculan terbesar pada bigrams di *ciphertext* secara terurut ialah terdapat pada ‘HZ’ yang mana pada Bigram bahasa inggris yang paling sering muncul ialah ‘TH’. Oleh karena sebelumnya kita mengasumsikan ‘H’ dipetakan menjadi ‘T’, akibatnya ‘Z’ dipetakan menjadi ‘H’ adalah masuk akal.

Sekarang, jika ditinjau dengan analisis frekuensi pada Trigram *ciphertext*, dan frekuensi Trigram yang paling sering muncul dalam bahasa inggris umumnya, maka diperoleh tabel sebagai berikut.

HZE	96	1.92 %	Trigrams Of 1,699,542,842 trigrams scanned: 1. the (59623899, 3.508232%) 2. and (27088636, 1.593878%) 3. ing (19494469, 1.147042%) 4. her (13977786, 0.822444%) 5. hat (11059185, 0.650715%) 6. his (10141992, 0.596748%) 7. tha (10088372, 0.593593%) 8. ere (9527535, 0.560594%) 9. for (9438784, 0.555372%) 10. ent (9020688, 0.530771%) 11. ion (8607405, 0.506454%) 12. ter (7836576, 0.461099%) 13. was (7826182, 0.460487%) 14. you (7430619, 0.437213%) 15. ith (7329285, 0.431250%) 16. ver (7320472, 0.430732%) 17. all (7184955, 0.422758%) 18. wit (6752112, 0.397290%) 19. thi (6709729, 0.394796%) 20. tio (6425262, 0.378058%)
BMV	49	0.98 %	
MVH	49	0.98 %	
XBM	48	0.96 %	
LRF	32	0.64 %	
YFC	32	0.64 %	
VHR	30	0.6 %	
EFH	29	0.58 %	
HLR	29	0.58 %	
LFO	29	0.58 %	
FHZ	25	0.5 %	
LFH	24	0.48 %	
ZEB	23	0.46 %	

Dari tabel kiri di atas, diperoleh ‘HZE’ sebagai Trigram yang paling sering muncul dalam *ciphertext* dan hal tersebut berkorespondensi dengan ‘THE’ sebagai Trigram yang paling sering muncul dalam bahasa inggris. Akibatnya, ‘H’ dipetakan menjadi ‘T’, ‘Z’ dipetakan menjadi ‘H’, serta ‘E’ dipetakan menjadi ‘E’ menjadi semakin cukup valid.

Letters

Of 3,104,375,038 letters scanned:

1. e (390395169, 12.575645%)
2. t (282039486, 9.085226%)
3. a (248362256, 8.000395%)
4. o (235661502, 7.591270%)
5. i (214822972, 6.920007%)
6. n (214319386, 6.903785%)
7. s (196844692, 6.340880%)
8. h (193607737, 6.236609%)
9. r (184990759, 5.959034%)
10. d (134044565, 4.317924%)

Jika meninjau kembali dari frekuensi kemunculan huruf lain, dari tabel di atas dan tabel analisis frekuensi huruf pada *ciphertext* soal yang disajikan sebelumnya, dengan mencoba kemungkinan, huruf lain yang sering muncul pada *ciphertext* yaitu 'Y' dan 'A' sebagai huruf yang sering muncul dalam bahasa inggris, keduanya sama-sama yang menempati *ranking* ke-3 pada masing-masing mereka. Maka kita dapat menduga 'Y' dapat dipetakan menjadi 'A'.

Bigrams

Of 2,383,373,483 bigrams scanned:

1. th (92535489, 3.882543%)
2. he (87741289, 3.681391%)
3. in (54433847, 2.283899%)
4. er (51910883, 2.178042%)
5. an (51015163, 2.140460%)

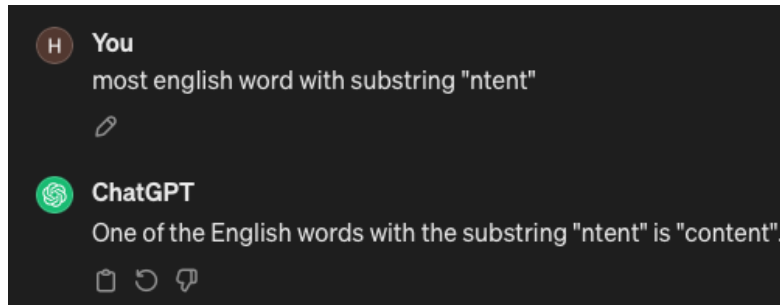
Setelah itu, jika dilihat dari Bigram dan Trigram yang teratas yang paling sering muncul dalam bahasa inggris yang diawali dengan huruf 'A' yaitu berturut-turut 'AN' dan 'AND', serta pada Bigram dan Trigram *ciphertext* pada tabel sebelumnya, perhatikan bahwa 'YF' dan 'YFC' merupakan Bigram dan Trigram teratas yang masing-masing tetap valid jika dipetakan menjadi 'AN' dan 'AND', maka kita dapat peroleh 'Y' dapat dipetakan menjadi 'A', 'F' dapat dipetakan menjadi 'N', dan 'C' dapat dipetakan menjadi 'D'.

Akibatnya, diperoleh potongan *plaintext* yang belum sempurna yaitu sebagai berikut.

UnLXatLRnBatheBthanXRntentGRBPLWL

Sehingga, penulis curiga dengan *substring* “*ntent*” dan kemunculan “*than*” sebelum “*ntent*”. Secara umum, “*than*” merupakan konjungsi, maka dengan kata lain “__*ntent*__...” dapat dicurigai merupakan satu kata.

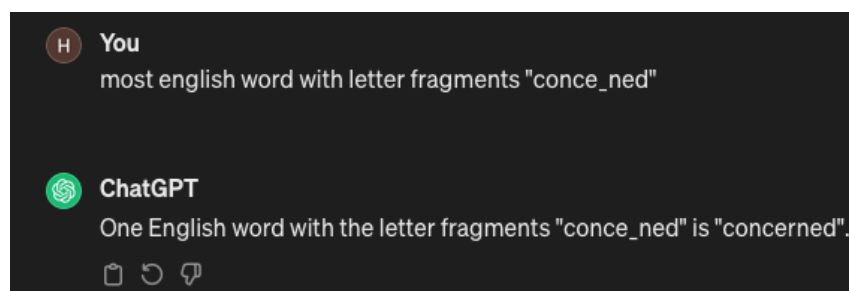
Disini penulis curiga dengan *substring* “*ntent*”, dan mencoba bertanya dengan ChatGPT, dan hasilnya sebagai berikut.



Maka cukup logis, bahwa ‘X’ dipetakan menjadi ‘C’ dan ‘R’ dipetakan menjadi ‘O’ yang mana keduanya tidak berkontradiksi dengan pemetaan huruf yang sudah dilakukan sebelumnya. Sehingga, disini penulis mendapatkan petunjuk baru yaitu sebagai berikut.

`/vtoOBaVhMLAconceBnedTLththeTBLtLnOoC`

Perhatikan bahwa di sini terdapat potongan string “*conce_ned*”. Akibatnya, lagi-lagi penulis curiga dan kembali bertanya kepada ChatGPT, dan hasilnya sebagai berikut.



Oleh karena itu, dapat dilakukan substitusi ‘B’ menjadi ‘R’ yang mengakibatkan diperoleh potongan plaintext lain sebagai berikut.

`AtheAtUdMoGAecretcodeALnAVealLn`

Adapun pada potongan tersebut, terdapat potongan *string* yang mencurigakan, yaitu “_ecretcode_”, yang mana itu cukup masuk akal jika diterjemahkan menjadi “*secret codes*” karena potongan “_ecret” paling mungkin jika diterjemahkan menjadi “*secret*” dan juga diperoleh kata “*codes*” setelah itu. Apabila dilakukan pengecekan juga pada ‘A’ dipetakan menjadi ‘S’ menjadi tetap valid. Akibatnya diperoleh potongan *plaintext* lain sebagai berikut.

```
atLonoGthesePethodsThLWe
```

Dari potongan tersebut, terdapat *string* “*these_ethods*” yang dicurigai dapat merujuk ke kata “*these method*” karena “*these*” sendiri merupakan kata depan dan “_ethods” yang paling mungkin adalah “*methods*”. Sehingga cukup masuk akal dan tetap valid jika ‘P’ dipetakan menjadi ‘M’. Setelah melakukan pemetaan substitusi tersebut, maka diperoleh potongan *plaintext* yang mencurigakan sebagai berikut.

```
standardsVresentedthemodLGLedNersLonasthedataencrMvtLonstandard
```

Dari potongan tersebut diperoleh potongan *string* “*standards_resented*” yang cukup valid jika diterjemahkan sebagai “*standards presented*” karena “*standards*” biasanya merupakan satu kata dan terpisah dari kata lain, maka “_resented” paling mungkin berarti “*presented*”. Akibatnya dapat disubstitusikan ‘V’ menjadi ‘P’ yang mana itu sekaligus berpengaruh terhadap *string* “*dataencr__t*” menjadi “*dataencr_pt*” yang dapat diduga diterjemahkan menjadi “*data encrypt*”. Maka dapat disubstitusikan juga ‘M’ menjadi ‘Y’.

Dari substitusi tersebut, dapat diperoleh potongan *plaintext* lain yang mencurigakan, yaitu sebagai berikut.

```
TomaLnKranchescryptoOraphyLsconcernedTi
```

Dari potongan tersebut, dapat diperoleh potongan *string* “*crypto_raphy*” yang sangat kemungkinan besar dapat diterjemahkan sebagai “*cryptography*”. Sehingga dapat disubstitusikan ‘O’ sebagai ‘G’, yang juga dilakukan pengecekan juga tetap valid.

Akibatnya, diperoleh dua potongan plaintext mencurigakan sebagai berikut.

```
nsecretcodeandthecreatLonoGthesemethods]
```

```
themeanLngoGTrLttenmessages]
```

Dari kedua potongan plaintext tersebut, terdapat potongan *string*

- *“Secretcodeandthecreat_ono_thesemethods”*
- *“themean_ngo__r_ttenmessages”*

yang diduga dapat diterjemahkan sebagai

- *“secret code and the creation of these methods”*
- *“The meaning of written messages”*

Karena *“Secret”, “code”, “and”, “the”, “these”, “methods”* merupakan satu kata. Akibatnya, *“creat_ono_”* dapat dicurigai dapat membentuk satu kata atau lebih yang independen. Hal tersebut sejalan dengan potongan string pada potongan plaintext kedua yang juga dapat dipecah menjadi *“the”, “mean_ngo__r_tten”, “messages”*.


Setelah itu, jika diperhatikan pada potongan *“creat_ono_”* yang jika dicari pada *word finder* tidak menghasilkan kata apapun.

Fill-in-the-Blanks Search

10 Letters

C R E A T O N O

10-Letter Words Including CREAT_ONO_



No results found

Please try another search

Sehingga yang paling mungkin ialah meninjau potongan “*creat_on*” yang jika huruf kosong diisi dengan huruf mati, maka kata tersebut tidak mempunyai arti apapun, maka yang paling mungkin adalah diisi dengan huruf vokal. Dengan sedikit mencoba, maka yang paling mungkin diperoleh adalah “*creation*”, hal tersebut didukung oleh hasil pencarian pada *word finder* berikut.

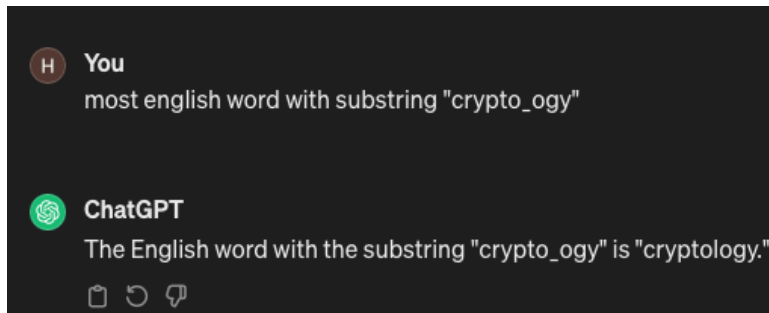
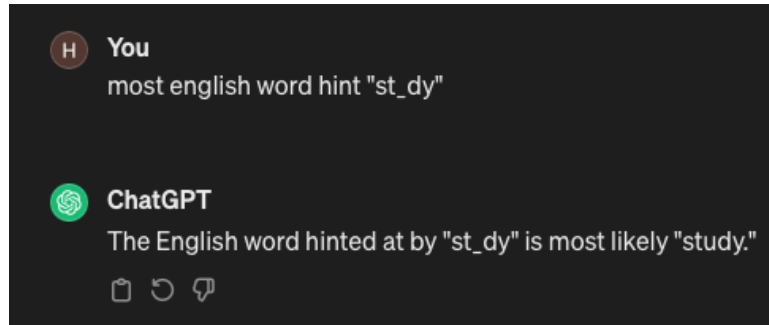
The image shows a web-based search interface titled "Fill-in-the-Blanks Search". It features a search bar with a dropdown menu set to "8 Letters" and a series of input boxes containing the letters C, R, E, A, T, O, N. A yellow search button with a magnifying glass icon is to the right. Below the search bar, a dark blue header reads "8-Letter Words Including CREAT_ON". Underneath, there are two filters: "All words 1" and "Common 1". The search results section displays the word "creation" in a light blue box.

yang mengakibatkan kita bisa substitusikan ‘L’ menjadi ‘I’, sehingga diperoleh “*meaningo__ritten*” yang bisa di-*split* menjadi “*meaning*”, “*o__ritten*”. Yang mana “*o__ritten*” dapat diterjemahkan menjadi “*of written*” sehingga dapat disubstitusikan ‘T’ menjadi ‘W’ dan ‘G’ menjadi ‘F’, yang jika dilakukan pengecekan, diperoleh hasil semuanya tetap valid.

Setelah itu, kita dapat peroleh kembali potongan *plaintext* lain yang mencurigakan, yaitu sebagai berikut.

cryptoWogyisthestUdyofsecretcodesinspealingofcryptoWogy

Dari *plaintext*, tersebut dapat diperoleh 2 potongan *string* yang mencurigakan yang diduga dapat membentuk 2 kata unik untuk kasus kemunculan huruf-huruf pada potongan *string* tersebut, yaitu “*crypto_ogy*” dan “*st_dy*” yang mana paling mungkin jika diterjemahkan menjadi “*cryptology*” dan “*study*”. Hal tersebut sejalan dengan yang dikatakan ChatGPT yaitu sebagai berikut.



Dan jika ‘U’ disubstitusikan menjadi ‘U’, serta ‘W’ disubstitusikan menjadi ‘L’, jika dilakukan pengecekan di tabel solusi pemetaan substitusi sementara keduanya tetap valid.

Setelah itu, kita dapat tinjau potongan *plaintext* berikut.

secretcodesinspealingofcryptology

Pada *plaintext* tersebut terdapat *string* “*inspealingofcryptology*” yang mana cukup masuk akal jika diterjemahkan menjadi kalimat “*in speaking of cryptology*”. Yang dalam hal ini kita dapat dapat peroleh pemetaan substitusi ‘I’ menjadi ‘K’.

Akibatnya kita peroleh table substitusi sementara sebagai berikut.

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

Dikarenakan pemetaan substitusi yang sudah didapatkan hampir lengkap, maka kita bisa memanfaatkan potongan *plaintext* yang sudah lengkap terisi menjadi sebuah kalimat bahasa inggris utuh. Untuk *plaintext* yang sudah terisi tersebut dilakukan penyusunan kalimat dengan spasi yang paling

mungkin berdasarkan konteks bahasa inggris untuk dilakukan pencarian pada search *engine*.

Dalam hal ini kita dapat melakukan pencarian terhadap potongan *string* berikut.

cryptologyisthestudyofsecretcodesinspeakingofcryptologywediscusstwomain

Yang mana jika diterjemahkan dan disusun terhadap kontek dalam bahasa inggris menjadi “*Cryptology is the study of secret codes. In speaking of cryptology, we discuss two main*”.

Adapun setelah dilakukan pencarian pada *search engine*, diperoleh hasil teratas sebagai berikut.



Setelah dilakukan pembacaan dan *scanning* terhadap semua alfabet, kita dapat peroleh hasil akhir untuk pemetaan substitusi yang valid, yaitu sebagai berikut.

Ciphertext	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
------------	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

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

Adapun *Plaintext* merupakan cuplikan dari

<https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=0e8776b71720ab523b686a66ab8a0a9b194a593b>.

Berkas Plaintext

Cryptography is the study of secret codes. In speaking of cryptography, we discuss two main branches: cryptography is concerned with the writing of messages in secret code and the creation of these methods, while cryptanalysis is concerned with reading encrypted messages by breaking secret codes.

Ancient times saw many examples of cryptography. Over three thousand years ago, Egyptian scribes made use of hieroglyphic transformations to obscure the meaning of written messages. Mesopotamian and Babylonian scribes employed similar techniques to render cuneiform tablets unreadable to the uninitiated. The Greeks employed cryptography (and the closely related steganography, which is concerned with concealing the existence of communication rather than content) for military and tactical purposes. Even the Kāma-sūtra of ancient India touches on cryptography, listing secret writing as one of the sixty-four fundamental arts. Hundreds of other examples occur in ancient civilizations; in short, cryptography has appeared more or less spontaneously in every culture in which literacy has become widespread [45].

Cryptanalysis, on the other hand, took considerably longer to develop as a rigorous subject of study. The earliest surviving descriptions of systematic methods of code-breaking come from fifteenth century, in the Arabic encyclopedia S. ubh. al-a 'sha [45]. It gives the first known written description of the technique of frequency analysis, where the frequencies of letters and letter groupings of a language are used to unravel secret codes.

Cryptography has played a role in political and military matters from

medieval times through the 20th century. Perhaps most famous is the cryptologic effort of Great Britain and the United States during World War II. The efforts of thirty thousand employees at Britain's Bletchley Park in cryptanalyzing Germany's Enigma transmissions is said to have shortened the war by several years [45], and it led to the development of the first digital computer, Colossus. Similar efforts in the United States to break Japanese codes aided American intelligence in the war, convincing American authorities of the importance of cryptologic expertise and eventually leading to the establishment of the National Security Agency in 1952.

With the rise of digital computer networks in the 1970s, securing communication became an increasingly important task in the private sector as well. The needs of the banking industry, in particular, for secure digital communication became especially urgent. Developments at IBM led to the design of Lucifer, one of the first public ciphers for computer communications. The National Security Agency offered several improvements to Lucifer, and in 1975 the National Bureau of Standards presented the modified version as the Data Encryption Standard, or D.E.S. Thus the first public standard for encryption began to gain widespread use.

A method of securing communication is called a cryptosystem. The sender encrypts (or enciphers) a message using an encryption algorithm together with a secret key. This produces a ciphertext which is sent to the recipient. The recipient, who also possesses a key, receives the ciphertext and decrypts (or decipheres) using the key to recover the original message, called the plaintext.

In the history of cryptology up to 1975, all cryptosystems required the sender and the receiver to agree beforehand on the same key, a key that had to be rigorously protected from exposure to an adversary. This is known as symmetric or secret key cryptography. Arranging to share a secret key between two parties is often a difficult problem, and does not scale well to scenarios in which many individuals or computers might communicate with each other.

In 1976, Martin Hellman, a professor at Stanford University, and Whitfield Diffie, a graduate student, introduced the concept of asymmetric or public key cryptography in their seminal paper *New Directions in Cryptography*. They speculated that a method of encryption might exist in which the encryption key differed from the decryption key. In such a scheme, a user's encryption key could be announced to the public; any outsider could obtain this public encryption key and use it to send encrypted messages to the user. Since only the user would possess the decryption key, only she could obtain the decryption of the message. Public key cryptography also opened doors for many other applications, such as digital signatures and electronic cash.

The concept of public key cryptography circulated in the research community for some time before the first practical proposal for such a scheme was made.¹ In August 1977 the RSA public key cryptosystem, named after inventors Ron Rivest, Adi Shamir, and Len Adleman, was introduced in Martin Gardner's column on Mathematical Games in *Scientific American* [32]. The RSA cryptosystem, outlined in Section 3.1, has survived over twenty years of study by cryptanalysts in the public sector, and it is the most widely used public key cryptosystem in the world. It is used, among other places, in the SET protocol [84] for secure credit card transactions and the SSL protocol [30] for secure communication on the Internet.

Public discussion and research in cryptography in business and academia started in the last quarter of the 20th century, and continues at a furious rate. New methods for encryption are publicly announced; researchers then study these methods for weaknesses by applying the tools of cryptanalysis. Only after intense public scrutiny does a new cryptosystem gain a sense of legitimacy. Studying and using methods for breaking cryptosystems is an essential step in the development of new designs for more secure cryptosystems. By learning how things break, we learn how to make them stronger.

It is in this spirit that this thesis is written. This work uses mathematical tools to study the RSA public key cryptosystem and several variants. We use tools from numerical algebra and the geometry of numbers to get our

results. Algebraic crypt- analysis has proven to be one of the most effective methods in the study of public key cryptosystems.

Tool yang digunakan:

1. https://wilsoa.github.io/gallery/frequency_analysis.html
2. <https://www3.nd.edu/~busiforc/handouts/cryptography/Letter%20Frequencies.html>
3. <https://www.merriam-webster.com/wordfinder>
4. <https://github.com/alifyasa/conan-cryptanalysis/blob/main/1%20-%20Substitution%20Cipher/main.py>
5. <https://chat.openai.com/>

Metode Kasiski

Akan dilakukan pencarian panjang kuncinya terlebih dahulu. Pencarian panjang kunci yang dilakukan dapat meninjau jarak perulangan kriptogram yang muncul, kemudian akan diambil faktor persekutuan terbesarnya sebagai kemungkinan panjang key.

Untuk mempermudah pencarian, dapat dilakukan pencarian terhadap Trigram yang sering berulang, karena dalam bahasa inggris terdapat beberapa Trigram yang sering berulang, contohnya seperti “THE”, “AND”, “ING”, dsb. Dalam hal ini dapat kita tinjau tabel kemunculan Trigram pada *ciphertext* tersebut sebagai berikut.

DPQ	12	0.19 %
GPK	12	0.19 %
GRM	12	0.19 %
ALS	11	0.17 %
LVR	11	0.17 %
VOI	11	0.17 %

Dilakukan list terhadap setiap jarak kemunculan terhadap masing-masing huruf yang sama.

Trigram	Distance to the Next
---------	----------------------

DPQ	108, 288, 60, 156, 672, 1248, 696, 24, 180, 1716, 1083
GPk	372, 228, 144, 348, 516, 600, 324, 84, 84, 1368, 744
GRM	108, 384, 72, 312, 1248, 708, 1224, 408, 48, 96, 732
ALS	252, 60, 996, 48, 384, 264, 432, 1332, 228, 591
LVR	156, 444, 888, 456, 408, 204, 84, 108, 2160, 384
VOI	924, 1164, 408, 264, 48, 1620, 48, 948, 156, 72

Dari distance tersebut diperoleh faktor persekutuan dari masing-masing Trigram dan Key dari Trigram jika hasil enkripsinya “THE” (karena dalam inggris paling sering muncul), yaitu:

Trigram	Greatest Common Divisor	Key
DPQ	3	KIM
GPk	12	NIG
GRM	12	NKI
ALS	3	HEO
LVR	12	SON
VOI	12	CHE

Maka dapat diperoleh kemungkinan panjang kunci ialah 3 atau 12 atau faktor dari 12 (2, 3, 4, 6, 12).

Perhatikan bahwa jika ternyata panjang kuncinya adalah > 3 , maka bisa dibuktikan dengan matematis, Semua Trigram *ciphertext* yang memiliki $GCD < 3$ pada tabel di atas bisa jadi tidak dipetakan satu-satu terhadap *plaintext*-nya, karena key yang melalui Trigram tersebut bisa jadi berbeda (tidak sama untuk setiap satu putaran *key*).

Akibatnya, diputuskan untuk melakukan percobaan pada panjang key 2, 3, 4, 6, 12 melalui metode "Frequency Analysis". *Ciphertext* dibagi menjadi n kolom, dimana n adalah tebakan panjang kunci. Setelah dicoba panjang kunci 2 sampai 16, didapatkan bahwa ketika dibagi 12, distribusi frekuensinya mirip dengan frekuensi letter bahasa inggris.

```

Guessed Key Length: 12
Key Guess: C [('G', '12.04'), ('P', '11.13'), ('V', '08.94')]
Key Guess: H [('L', '12.23'), ('P', '08.94'), ('A', '08.94')]
Key Guess: E [('I', '12.41'), ('R', '09.31'), ('X', '08.39')]
Key Guess: O [('S', '10.95'), ('H', '10.04'), ('B', '08.76')]
Key Guess: N [('R', '13.14'), ('G', '11.68'), ('A', '08.21')]
Key Guess: K [('O', '11.13'), ('X', '10.04'), ('D', '08.39')]
Key Guess: I [('M', '12.04'), ('B', '09.12'), ('V', '08.76')]
Key Guess: M [('Q', '13.69'), ('A', '09.31'), ('U', '08.94')]
Key Guess: H [('L', '10.40'), ('W', '08.58'), ('F', '08.39')]
Key Guess: O [('S', '11.13'), ('B', '09.67'), ('H', '08.94')]
Key Guess: N [('R', '12.77'), ('G', '10.95'), ('N', '08.58')]
Key Guess: V [('Z', '11.86'), ('K', '11.68'), ('U', '08.76')]
Overlapp Key Guess: CHEONKIMSONG

```

Kemudian dilakukan tebakan menggunakan panjang key 12. Untuk setiap kolom, asumsikan karakter dengan frekuensi paling tinggi adalah karakter 'E' (Karena dalam bahasa inggris hal tersebut biasanya terjadi). Setelah itu dilakukan penyusunan key pada ada pada Trigram yang mempunyai faktor persekutuan = 12 pada tabel sebelumnya, yaitu "CHE", "SON", "NKI", "NIG", "HEO", "KIM" untuk membentuk suatu key tebakan dengan panjang 12. Setelah disesuaikan, didapatkan key CHEONKIMSONG. Dapat dicoba bahwa key tersebut benar dan menghasilkan plaintext yang utuh.

Key: CHEONKIMSONG

Untuk program yang digunakan pada solusi ini dapat dilihat melalui pranala berikut:

[https://github.com/alifyasa/conan-cryptanalysis/blob/main/2%20-%20Meto%20Kasiski%20\(Vignere%20Cipher\)/main.py](https://github.com/alifyasa/conan-cryptanalysis/blob/main/2%20-%20Meto%20Kasiski%20(Vignere%20Cipher)/main.py)

Hasil Deskripsi: <https://www.itb.ac.id/history>

Tool yang digunakan:

1. <https://gchq.github.io/CyberChef/>
2. <https://planetcalc.com/8550/>
3. [https://github.com/alifyasa/conan-cryptanalysis/blob/main/2%20-%20Metode%20Kasiski%20\(Vignere%20Cipher\)/main.py](https://github.com/alifyasa/conan-cryptanalysis/blob/main/2%20-%20Metode%20Kasiski%20(Vignere%20Cipher)/main.py)
4. [https://github.com/alifyasa/conan-cryptanalysis/blob/main/2%20-%20Metode%20Kasiski%20\(Vignere%20Cipher\)/trigram.py](https://github.com/alifyasa/conan-cryptanalysis/blob/main/2%20-%20Metode%20Kasiski%20(Vignere%20Cipher)/trigram.py)

Playfair Cipher

Ciphertext

Berkas *ciphertext* dapat diakses melalui *link* berikut: <https://github.com/alifyasa/conan-cryptanalysis/raw/main/3%20-%20Playfair%20Cipher/ciphertext.txt>.

Analisis

Analisis dimulai dengan mengamati frekuensi bigram *ciphertext*. Dari hasil tersebut, akan dicoba untuk merekonstruksi *key* yang digunakan. Berkas penuh percobaan rekonstruksi *key* dapat diakses pada <https://github.com/alifyasa/conan-cryptanalysis/tree/main/3%20-%20Playfair%20Cipher/keys>.

Berikut alur rekonstruksi *key*:

1. Diamati frekuensi 'HE' dan 'EL' di *ciphertext*. Karena dua bigram tersebut memiliki frekuensi tertinggi, masuk akal apabila diasumsikan bahwa mereka adalah 'TH' dan 'HE'.
2. Dengan alasan yang sama dengan poin sebelumnya, didapatkan bigram ciphertext 'HM' dan 'MH' sama dengan bigram plaintext 'ER' dan 'RE'.
3. Sejauh ini, telah didapatkan 4 karakter dalam baris '_ T H E L'. Bigram '_T' berkorespondensi dengan bigram 'TH', yaitu bigram dengan frekuensi yang tinggi. Bigram yang memenuhi ini adalah 'IT' dan 'AT'. Dengan meninjau frekuensi 'IT', 'TI', 'AT', dan 'TA', didapatkan bahwa 'A' memiliki frekuensi yang cocok.
4. Selanjutnya, rekonstruksi *key* dilakukan dengan mengambil cuplikan ciphertext yang setengah terdekripsi dan mencari padanan kata bahasa Inggris yang cocok. Pertama, ditemukan 'THE KH ERETHELA

VA' yang diduga merupakan 'THEY WERE THE LAST'. Dari itu, didapatkan bahwa 'KH' adalah 'YW' dan 'VA' adalah 'ST'.

5. Kemudian cuplikan 'LPMVTM ERALYEAKE XGFXTG', yang diduga merupakan 'SEVERAL YEAR'. Dilanjutkan dengan 'BTHEYHADNTMETLPRSEVERALYEAR' yang diduga merupakan 'THEY HADNT MET FOR SEVERAL YEAR'.
6. Kemajuan berikutnya dari 'THEYWEREPERFTDTLYNORMAL' yang diduga merupakan 'THEY WERE PERFECTLY NORMAL' dan 'PEOPLEHEMADESEVERAAUMPORTABETELEPHONECALL' yang diduga merupakan 'PEOPLE HE MADE SEVERAL IMPORTANT TELEPHONE CALL'.
7. Setelah langkah diatas, sebagian besar *plaintext* telah didekripsi. Langkah terakhir adalah untuk mencarinya di Internet.

Di akhir, didapatkan bahwa key adalah
THELAZYQUICKDOGVRMPSBWNFX.

Plaintext

Hasil *plaintext* merupakan cuplikan dari *Harry Potter and The Philosopher's Stone Chapter 1: The Boy Who Lived*. Hasil penuh dekripsi dapat diakses melalui <https://github.com/alifyasa/conan-cryptanalysis/raw/main/3%20-%20Playfair%20Cipher/plaintext.txt>.

Hill Cipher

Ciphertext

Ciphertext dapat diakses melalui link <https://github.com/alifyasa/conan-cryptanalysis/raw/main/4%20-%20Hill%20Cipher/ciphertext.txt>.

Analisis

Diketahui *key* memiliki bentuk 3×3 . Dalam Hill Cipher, *ciphertext* merupakan perkalian *key* dan *plaintext*. Jika *ciphertext* dan *plaintext* juga berbentuk 3×3 , *key* dapat diperoleh dengan mengalikan *ciphertext* dengan *inverse plaintext*.

Adapun untuk potongan plaintext yang diketahui di soal adalah “HELLOAIHA” yang merupakan kalimat awal surat dan potongan ciphertext pada awal surat adalah “IIXJPYVIE”.

Adapun matrix untuk potongan *plaintext* dan *ciphertext* tersebut berturut-turut adalah sebagai berikut.

$$\begin{bmatrix} 7 & 11 & 8 \\ 4 & 14 & 7 \\ 11 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 8 & 9 & 21 \\ 8 & 15 & 8 \\ 23 & 24 & 4 \end{bmatrix}$$

Sehingga, melalui perkalian operasi matriks *ciphertext* dengan *inverse plaintext*, kita dapat peroleh *key*-nya adalah “RVCRSCFVT”.

Setelah itu, dengan *key* yang sudah diperoleh kita bisa mendapatkan *plaintext* secara penuh melalui operasi dekripsi Hill Cipher.

Plaintext

Plaintext merupakan artikel <https://www.france24.com/en/live-news/20240207-ukraine-born-miss-japan-gives-up-crown-amid-affair-scandal>.

Hasil dekripsi terdapat di <https://github.com/alifyasa/conan-cryptanalysis/raw/main/4%20-%20Hill%20Cipher/plaintext.txt>.

Affine Cipher (Bonus)

Untuk memecahkan problem tersebut tentunya kita harus mengetahui skema enkripsi yang digunakan. Dapat dilakukan reverse engineering terhadap kode yang telah dibuat pada soal.

Perhatikan bahwa pada soal terdapat bilangan n , b , m sebagai input terhadap fungsi `affine_cipher`. Sebelum masuk ke fungsi tersebut, perhatikan potongan kode di bawah ini.

```
n = 256
b = random.randint(1,n)
m = random.randint(1,n)
while math.gcd(m, n) != 1:
    m = random.randint(1, n)
```



```
hex_values = read_image_to_hex(image_path)
if hex_values is not None:
    Cipher_hex = affine_cipher(hex_values, m, b, n)
    Bytearray_cipher = array_of_hex_to_bytearray(cipher_text)
    create_file_from_bytes("./chall.jpg", bytearray_cipher)
if __name__=="__main__":
    main()
```

Dari potongan kode tersebut nilai b dan m dibangkitkan secara random, dimana $\text{gcd}(m, n) = 1$ dan $n = 256$.

Dari sini, kita mengetahui bahwa m harus relatif prima dengan 256.

Perhatikan juga bahwa $256 = 2^8$ merupakan bilangan perpangkatan 2 yang tidak mempunyai faktor lain selain 2. Akibatnya, dapat diperoleh semua bilangan yang relatif prima dengan 256 adalah semua bilangan ganjil yang < 256 yang dapat dibuktikan secara *trivial*.

Jika ditinjau fungsi-fungsi lain seperti `read_image_to_hex`, `array_of_hex_to_bytearray`, `create_file_from_bytes` semuanya merupakan fungsi pembantu dalam konversi tipe data pada masing-masing kondisi terhadap pembacaan file gambar dalam bytes, konversi dari byte ke hexa, hexa ke byte, dan pembuatan file dari bytes. Dengan demikian, fungsi-fungsi tersebut tidak terlalu berarti terhadap logic utama enkripsi yang dilakukan.

Untuk logic utama enkripsi yang dilakukan pada soal ini adalah terletak pada fungsi `affine_cipher`, yang mana potongan kode fungsi tersebut dapat disajikan sebagai berikut.

```
def affine_cipher(hex_values, m, b, n):
    cipher_hex = []
    for i in range(len(hex_values)):
        C = hex((m * int(hex_values[i], 16) + b) % n)
        cipher_hex.append(C)
    return cipher_hex
```

Dari potongan kode tersebut, dapat diketahui bahwa untuk setiap nilai hexa pada variabel C yang dimasukkan ke dalam array `cipher_hex`, sebelum dilakukan konversi ke hexadecimal, dilakukan perkalian m terhadap

masing-masing elemen untuk setiap array `hex_values` yang mana pada main function `hexa_values` diisi dengan nilai hexa yang diperoleh pada pembacaan bytes pada gambar. Kemudian untuk setiap perkalian tersebut ditambah dengan nilai `b` dan hasilnya di-modulo `n`.

Akibatnya kita dapat deduksikan persamaan matematika berikut.

$$C_i = (m \times P_i) + b \bmod n$$

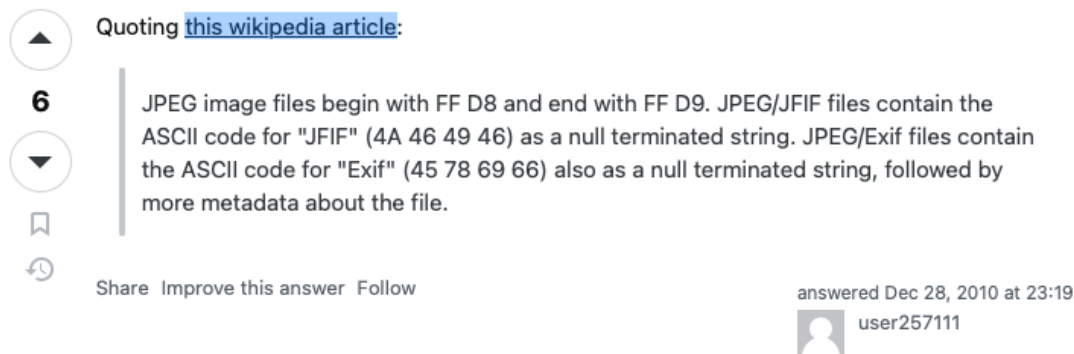
Karena $n = 256$, maka persamaan tersebut menjadi:

$$\begin{aligned} C_i &= (m \times P_i) + b \bmod 256 \\ \Leftrightarrow C_i &= 256 * k_i + m * P_i + b \dots(1) \end{aligned}$$

untuk suatu bilangan bulat k_i .

Sekarang, perhatikan bahwa pada file dengan format JPG yang valid itu haruslah selalu diawali dengan 2 bytes FF dan D8, serta diakhiri dengan 2 bytes FF dan D9. Berikut kami mendapatkan informasi tersebut melalui laman *stackoverflow* berikut:

<https://stackoverflow.com/questions/4550296/how-to-identify-contents-of-a-byte-is-a-jpeg>.



Quoting [this wikipedia article](#):

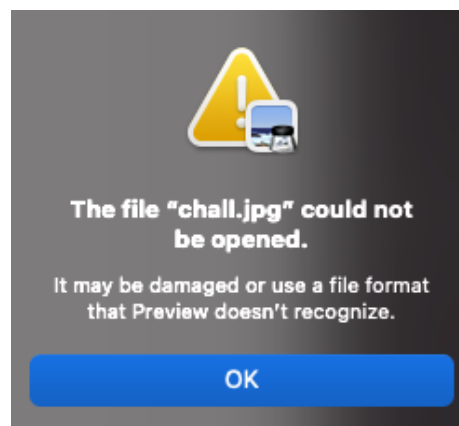
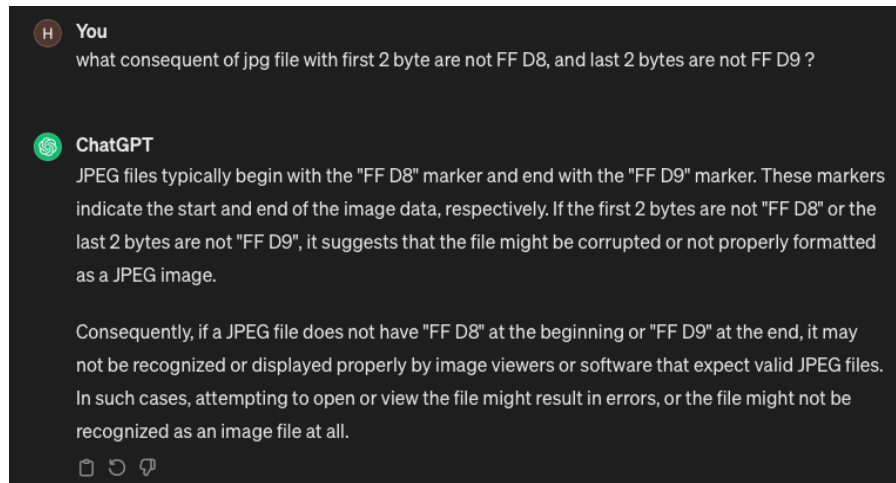
6

JPEG image files begin with FF D8 and end with FF D9. JPEG/JFIF files contain the ASCII code for "JFIF" (4A 46 49 46) as a null terminated string. JPEG/Exif files contain the ASCII code for "Exif" (45 78 69 66) also as a null terminated string, followed by more metadata about the file.

Share Improve this answer Follow

answered Dec 28, 2010 at 23:19
user257111

Untuk mengenai *file* soal yang tidak bisa dibuka (*corrupt*) yang merupakan hasil enkripsi pada setiap *byte*-nya dengan *Affine Cipher*, itu diakibatkan 2 *byte* awalnya tidak diawali dan diakhiri dengan masing-masing *byte* yang telah disebutkan sebelumnya untuk membentuk *file* JPG yang valid.



Oleh karena itu, karena *plaintext* haruslah menghasilkan informasi konten yang informatif. Maka haruslah, *file* JPG yang dihasilkan pun valid. Akibatnya masalah sub-soal dapat direduksi menjadi mencari nilai m dan b pada persamaan (1) yang diketahui 3 buah nilai P_i yaitu FF, D8, D9 dalam basis *hexadecimal* yang dikonversi ke desimal berturut-turut menjadi 255, 216, 217. Jika ditinjau 2 bytes pertama dan 2 bytes terakhir dari *ciphertext*, maka akan didapat nilai unik dari masing-masing *bytes* tersebut secara terurut adalah 16, 225, 154 atau dapat dilihat melalui tangkapan gambar console terminal penulis di bawah ini.

```

nodrop@Gilangs-Air 5 - Affine Cipher % python3 main.py
16
position 0
225
position 1
16
position 65386
154
position 65387
total length 65388

```

Akibatnya, dapat kita deduksikan persamaan berikut.

$$16 = 256k_1 + 255m + b \dots(2)$$

$$225 = 256k_2 + 216m + b \dots(3)$$

$$154 = 256k_3 + 217m + b \dots(4)$$

Jika persamaan (2) dan (3) dikurangkan maka akan diperoleh persamaan:

$$209 = 256(k_1 - k_2) + 39m \dots(5)$$

$$71 = 256(k_2 - k_3) - m \dots(6)$$

Dari persamaan (6) dapat diperoleh:

$$m = 256(k_2 - k_3) - 71$$

$$m \equiv -71 \pmod{256}$$

$$m \equiv 185 \pmod{256}$$

Karena pada bagian sebelumnya telah diperoleh m adalah bilangan ganjil < 256 , maka haruslah $m = 185$.

Sekarang, tinjau kembali persamaan (2), kita akan manipulasi persamaan tersebut menjadi sebagai berikut.

$$16 = 256k_1 + 255m + b$$

$$\Leftrightarrow b = 16 - 256k_1 - 255m$$

$$\Leftrightarrow b = 16 - 256(k_1 + m) + m$$

$$\Leftrightarrow b = 256u + 16 + m \dots(7)$$

Untuk suatu u bilangan bulat.

Apabila persamaan (7) disubstitusikan dengan $m = 185$, maka akan diperoleh:

$$b = 256u + 16 + 185$$

$$\Leftrightarrow b = 256u + 201$$

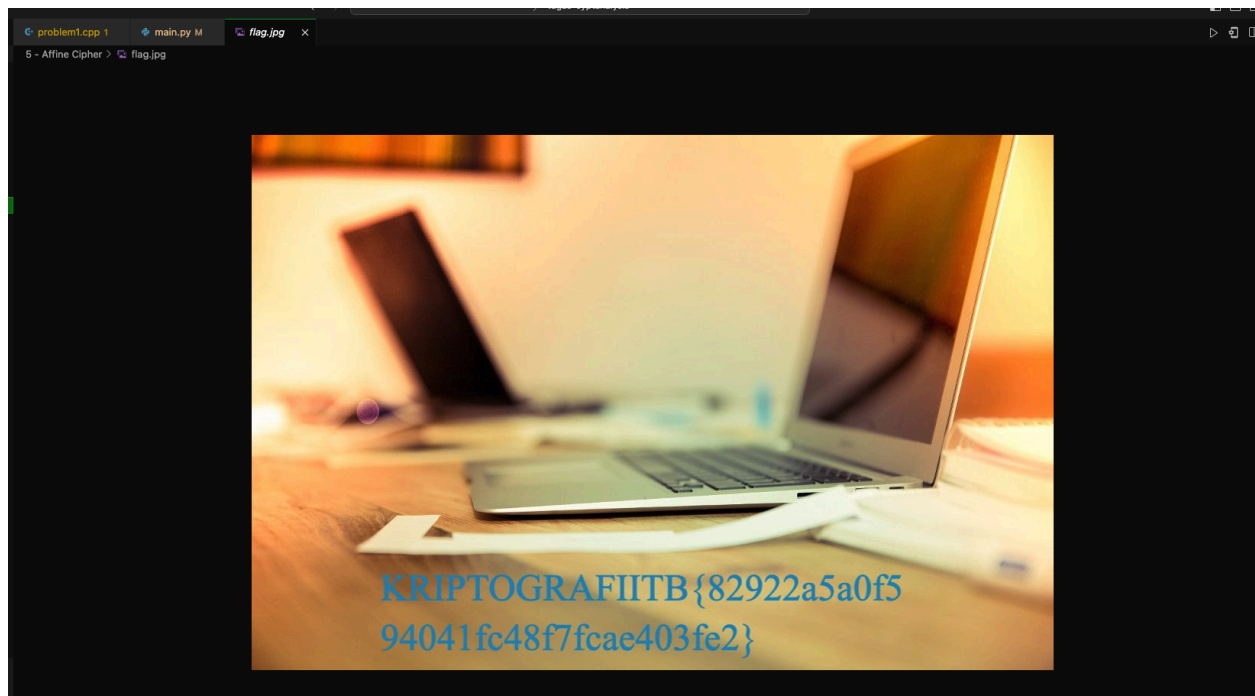
$$\Leftrightarrow b \equiv 201 \pmod{256}$$

Karena pada soal, $b \geq 1$ dan $b \leq 256$, maka haruslah $b = 201$. Dapat dicek juga bahwa $m = 185$ dan $b = 201$ ke persamaan modulo soal adalah benar.

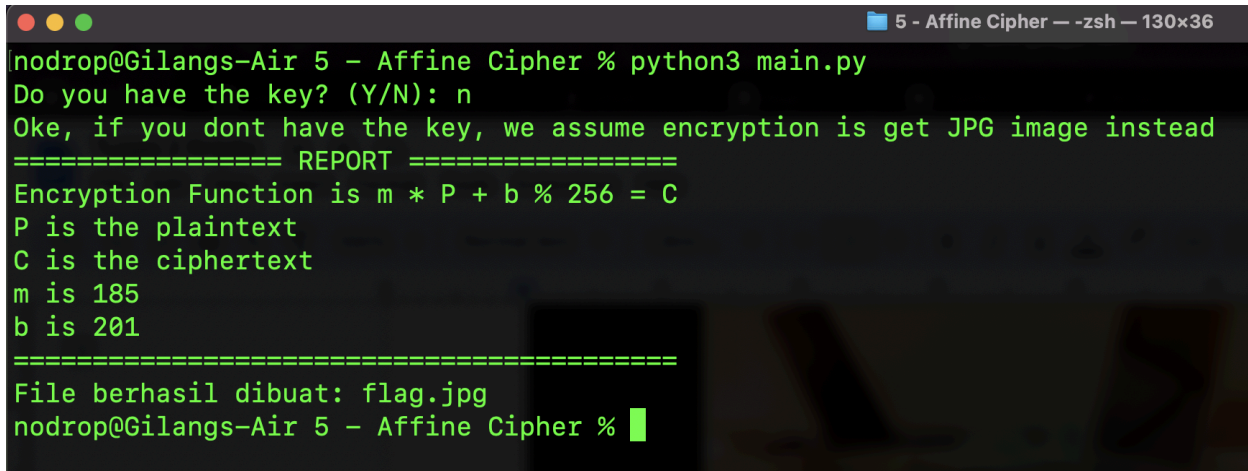
Dari kunci tersebut, file hasil enkripsi dapat didekripsi dengan mudah melalui program sederhana yang telah dibuat melalui pranala berikut:

<https://github.com/alifyasa/conan-cryptanalysis/blob/main/5%20-%20Affine%20Cipher/main.py>

Sehingga diperoleh hasil dekripsi-nya adalah gambar JPG sebagai berikut.



Untuk solusi kedua tanpa pendekatan matematika secara *analytics*, yaitu dapat memanfaatkan komputasi *programming*, dapat dilihat melalui pranala: <https://github.com/alifyasa/conan-cryptanalysis/blob/main/5%20-%20Affine%20Cipher/main.py>



```
nodrop@Gilangs-Air 5 - Affine Cipher % python3 main.py
Do you have the key? (Y/N): n
Oke, if you dont have the key, we assume encryption is get JPG image instead
===== REPORT =====
Encryption Function is m * P + b % 256 = C
P is the plaintext
C is the ciphertext
m is 185
b is 201
=====
File berhasil dibuat: flag.jpg
nodrop@Gilangs-Air 5 - Affine Cipher %
```

Untuk penjelasan program yang dibuat sebenarnya pendekatannya sama, cuma pada pendekatan komputasi *programming*, kita dapat memanfaatkan invers modulo m terhadap n , yang pada kasus soal ini m relatif prima terhadap n dan $1 \leq m \leq n$, maka dapat hitung melalui *library* perpangkatan *invers* pada *python*.

Untuk nilai m dan b dapat dihitung sebagai berikut.

$$\begin{aligned}diffc_{12} &= bc_1 - bc_2 \\diffp_{12} &= bp_1 - bp_2 \\m &= diffc_{12} * (diffp_{12}^{-1} \bmod n) \bmod n \\b &= bc_1 - bp_1 \bmod n\end{aligned}$$

Dengan :

bc_1 = byte pertama gambar ciphertext

bc_2 = byte kedua gambar ciphertext

bp_1 = byte pertama gambar plaintext

bp_2 = byte kedua gambar plaintext

