Incognita Block Cipher

Elvina[1], **Saskia Imani**[2].

[1,2] Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika (STEI), Institut Teknologi Bandung (ITB), Jalan Ganesha 10, Bandung 40132
E-mail: [1] 13517079@std.stei.itb.ac.id, [2] 13517142@std.stei.itb.ac.id

Abstract. Incognita merupakan rancangan *block cipher* baru yang diturunkan dari beberapa block cipher yang sudah ada dengan menganalisa apa yang menurut penulis menjadi kekurangan dari *block-block cipher* tersebut. Berdasarkan hasil pengamatan, *block-block cipher* seperti DES, GOST, dan AES menggunakan tabel yang telah didefinisikan di awal oleh pembuat algoritma dan menetapkan panjang kunci yang digunakan. Hal tersebut dapat memudahkan kriptanalisis dalam melakukan serangan, khususnya *brute force attack*. Dengan Incognita, diharapkan keamanan kunci menjadi meningkat, dan kerahasiaan pesan menjadi lebih aman. **Keywords**: *ciphertext*, *plaintext*, kunci, kriptografi, enkripsi, dekripsi, *block cipher*

1. Pendahuluan

Pada era digital saat ini, pertukaran data tidak lagi dilakukan dengan jalur manual atau analog. Semua data disimpan dan dipertukarkan secara digital. Data berupa pesan biasanya digunakan untuk komunikasi antara satu orang dengan orang lainnya. Setiap pesan memiliki makna, ada yang memiliki makna biasa, ada juga yang memiliki makna penting dan rahasia. Penyadapan pesan sangat mungkin terjadi karena pesan dikirim melalui teknologi digital. Untuk menjaga kerahasiaan pesan, dibuat algoritma-algoritma kriptografi. Ada berbagai macam algoritma kriptografi, baik pada kriptografi klasik maupun kriptografi modern. Salah satu dari banyaknya tipe kriptografi modern adalah *block cipher*. Algoritma-algoritma *block cipher* melakukan enkripsi pada pesan dengan membagi-bagi pesan menjadi *block-block* dengan ukuran tertentu terlebih dahulu. Dengan demikian, diharapkan kriptanalis menjadi lebih sulit dalam melakukan serangan (*attack*).

Beberapa contoh algoritma *block cipher* yang terkenal adalah DES, GOST, dan AES. Ketiga algoritma tersebut memiliki prinsip yang mirip dalam melakukan enkripsi, karena GOST dan AES dibuat dari kekurangan-kekurangan yang dimiliki oleh DES. Untuk melakukan pengembangan dari algoritma DES, perancang algoritma perlu menganalisa terlebih dahulu apa yang terjadi di dalam algoritma DES. Pada algoritma DES, pembangkitan tabel-tabel yang digunakan untuk melakukan pengacakan *plaintext* dan kunci dalam implementasi *confusion* dan *diffusion* dilakukan secara manual, sehingga setiap orang yang melakukan enkripsi dengan menggunakan DES akan membangkitkan tabel yang sama.

Incognita merupakan algoritma yang dibuat dengan latar belakang masalah tersebut, dengan mengambil inspirasi dari beberapa algoritma yang sudah ada, serta menerapkan imajinasi dari perancang algoritma, sehingga menjadi sebuah algoritma baru yang diharapkan dapat menjadi solusi yang lebih baik daripada beberapa algoritma yang sudah ada sebelumnya.

2. Studi Pustaka

2.1. Block Cipher

Block Cipher merupakan salah satu tipe dari kriptografi modern yang memanfaatkan kunci simetri. Selain *block cipher*, tipe kriptografi modern lain yang memanfaatkan kunci simetri adalah *stream cipher*.

Pada kriptografi klasik, enkripsi dilakukan dengan mengoperasikan setiap karakter yang dibaca pada sebuah text atau file random tertentu. Berbeda dengan kriptografi klasik, kriptografi modern melakukan operasi-operasi bit dalam melakukan enkripsi dan dekripsi. *Stream cipher* melakukan operasi pada setiap bit yang ada pada suatu *plaintext* satu per satu pada tiap bit yang ada. Sementara itu, *block cipher* membagi terlebih dahulu *plaintext* ke dalam beberapa blok yang berukuran sama. Misalnya pada algoritma kriptografi DES (atau lebih tepatnya algoritma kriptografi DEA), *plaintext* dibagi ke dalam blok-blok berukuran 64 bit. Setiap blok harus memiliki ukuran bit yang sama. Jika ukuran *plaintext* tidak tepat berjumlah kelipatan 64, maka pada saat dilakukan pembagian blok harus ditambahkan *padding* sehingga tiap blok berukuran tepat 64 bit.

Terdapat beberapa kelebihan dan kekurangan dari *block cipher* bila dibandingkan dengan *stream cipher*. *Block cipher* memiliki algoritma yang relatif lebih rumit, sehingga waktu yang dibutuhkan untuk enkripsi dan dekripsi menjadi lebih lama. Sejumlah besar *block cipher* juga memiliki ketergantungan antar-*block*, sehingga *block cipher* seringkali tidak dapat dieksekusi secara paralel terhadap teks. Namun lebih sulit melakukan serangan terhadap *block cipher*. Karena ketergantungan bit pada *block* bit itu sendiri dan bahkan pada *block* lainnya, block cipher lebih aman terhadap serangan *plaintext* maupun *ciphertext*. Ketergantungan ini pula menyebabkan serangan terhadap *bit* akan mudah diketahui, karena pesan yang termodifikasi bila didekripsi akan menjadi sangat berbeda dari pesan aslinya.

2.2. Confusion dan Diffusion

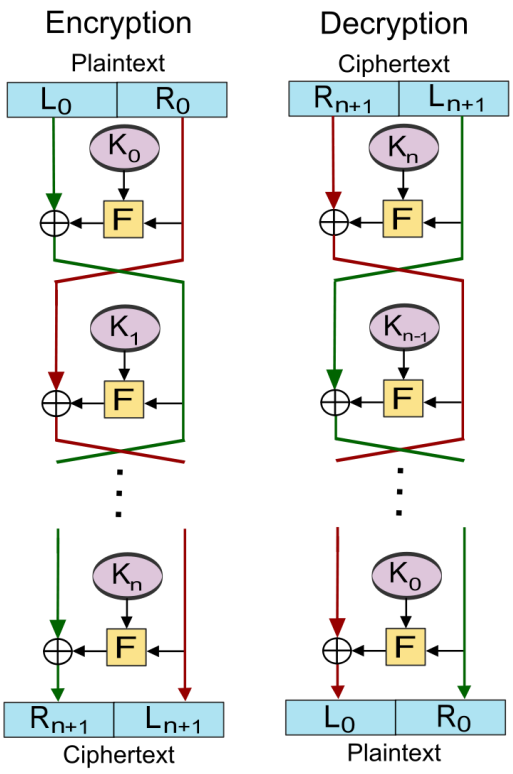
Confusion dan *diffusion* adalah dua properti dari sebuah cipher yang aman, yang dikemukakan oleh Claude Shannon. *Confusion* berarti setiap bit dari ciphertext bergantung pada beberapa bagian dari kunci, sehingga hubungan antara keduanya tersamarkan. Sedangkan *diffusion* berarti jika satu bit pada teks (*plaintext* maupun *ciphertext*) diubah, maka sebanyak mungkin bit pada teks hasil enkripsi atau dekripsi akan berubah, sehingga hubungan antara *plaintext*, kunci, dan *ciphertext* tersamarkan. Dalam teori awal Shannon, *diffusion* merujuk kepada penyamaran struktur statistik *plaintext* dalam *ciphertext*. *Confusion* dan *diffusion* pada suatu algoritma *cipher* umumnya ditingkatkan dengan menerapkan permutasi dan substitusi.

2.3. Iterated Cipher (cipher berulang)

Dalam *cipher* berulang, sebuah blok *plaintext* dienkripsi secara berulang kali menjadi sebuah *ciphertext* yang berukuran sama, dan begitu pula sebaliknya dalam dekripsi. Satu kali enkripsi atau dekripsi disebut dengan satu *round*, dan *round* ke-i menggunakan suatu kunci putaran (K_i) yang dibangkitkan dari kunci awal (K).

2.4. Jaringan Feistel

Jaringan Feistel adalah sebuah model *cipher* yang bersifat reversibel, sehingga dapat digunakan satu model yang sama untuk proses enkripsi dan dekripsi. Perbedaan dari kedua proses tersebut hanyalah penggunaan kunci (K_i) dengan urutan yang terbalik antara proses enkripsi dan dekripsi. Jaringan Feistel banyak digunakan dalam algoritma *block cipher*, contohnya DES, LOKI, GOST, dan Blowfish. Ilustrasi dari struktur sebuah jaringan Feistel adalah sebagai berikut:



Gambar 1. Struktur model jaringan Feistel

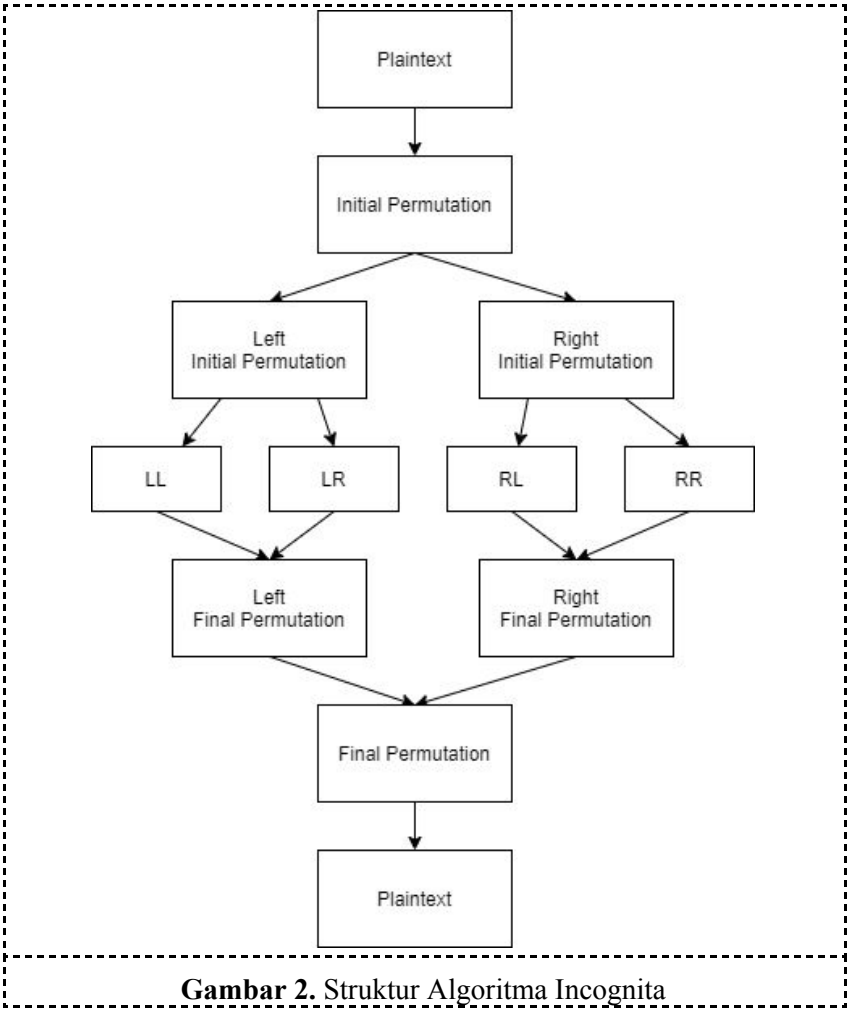
2.5. Kotak-S

Substitution box (*S-box*) atau kotak-S adalah sebuah komponen dalam kriptografi yang melakukan substitusi dengan cara memetakan satu atau lebih bit menjadi satu atau lebih bit lainnya. Substitusi yang dilakukan pada sebuah blok bit berukuran m menghasilkan sebuah blok bit berukuran n, dengan m dan n dapat memiliki nilai yang sama ataupun berbeda. Sebuah kotak-S berukuran mxn diterapkan dengan menggunakan sebuah *lookup table* yang berisi 2^m blok bit yang berukuran n.

3. Proposed Block Cipher

Incognita merupakan algoritma *block cipher* yang memanfaatkan prinsip-prinsip *confusion* dan *diffusion* dari Shannon, cipher berulang, jaringan feistel, dan kotak-S. Algoritma ini dapat menerima *plaintext* tanpa batasan panjang dan akan membagi-bagi *plaintext* tersebut ke dalam blok-blok yang berukuran 128 bit. Panjang kunci yang diterima oleh algoritma ini memiliki batasan panjang minimal sebesar 64 bit. Selama panjang kunci lebih besar atau sama dengan 64 bit, tidak ada batasan lain untuk membuat kunci dapat digunakan sebagai kunci enkripsi.

Sebelum dimulai penjelasan lebih rinci mengenai algoritma yang diusulkan, berikut ini merupakan gambaran struktur dari algoritma Incognita :



Gambar 2. Struktur Algoritma Incognita

3.1. Initial Processing

Pada implementasi algoritma yang telah dibuat, program akan menerima *plaintext* yang diinput oleh pengguna. Sebelum diolah oleh algoritma Incognita, seperti yang telah dijelaskan sebelumnya, *plaintext* akan dibagi menjadi beberapa blok berukuran 128 bit. Proses pembagian dilakukan dengan melakukan iterasi terhadap bit-bit *plaintext* dan menyimpannya ke dalam *list of blocks*. Jika panjang *plaintext* tidak tepat kelipatan 128 bit, maka akan dilakukan padding dengan menambahkan bit-bit *padding* sehingga semua blok yang dibuat memiliki panjang yang sama.

Selain *plaintext*, pengguna juga akan memasukan key yang akan digunakan dalam algoritma enkripsi. Key yang digunakan harus memiliki panjang minimal 64 bit, karena key tersebut akan digunakan dalam proses perulangan cipher, sehingga perlu batasan panjang minimal sehingga key memiliki panjang minimal sebesar bagian dari *plaintext* yang akan diolah.

3.2. Initial Permutation dan Final Permutation

Initial Permutation dan *Final Permutation* pada Incognita masing-masing dibangkitkan sebanyak 3 kali. Satu kali pada 128 bit blok *plaintext*, satu kali pada potongan sebelah kiri *plaintext*, dan satu kali pada potongan sebelah kanan *plaintext*. *Initial* dan *Final Permutation* Incognita memanfaatkan tabel permutasi yang dibangkitkan dari key, sehingga tabel yang dibuat akan unik tergantung dari key yang dimasukkan oleh pengguna.

Tabel permutasi yang digunakan pada *initial permutation* dibangun dengan mengalokasi terlebih dahulu angka-angka dari 1 sampai panjang tertentu (128 bit untuk blok 128 bit, dan 64 bit untuk bagian kiri dan kanan blok yang berukuran 64 bit) ke dalam sebuah *list*. Kemudian, dilakukan

permutasi *random* dengan menggunakan key sebagai *seed*. Dengan demikian, akan dihasilkan tabel permutasi yang unik tergantung dari key. Setelah itu, tabel *final permutation* dibuat dengan mencari invers dari tabel permutasi yang telah dibuat sebelumnya.

Tabel-tabel permutasi ini akan dimanfaatkan sebagai acuan permutasi *plaintext* dalam blok 128 bit, serta bagian kiri dan kanan blok yang berukuran 64 bit. Algoritma permutasi yang digunakan diambil dari cara algoritma DES dan GOST melakukan permutasi, yaitu dengan mengacak urutan bit *plaintext* berdasarkan urutan angka yang ada di dalam tabel.

3.3. Jaringan Feistel

Jaringan feistel yang digunakan pada algoritma Incognita memanfaatkan operasi-operasi sederhana seperti permutasi, operasi XOR, dan substitusi dengan menggunakan kotak-S. Struktur umum yang digunakan dalam jaringan feistel untuk algoritma Incognita mirip dengan jaringan feistel pada algoritma DES. Sebelumnya, telah dijelaskan bahwa blok *plaintext* 128 bit akan dipecah menjadi 2 bagian kiri dan kanan sepanjang 64 bit. Setiap bagian ini kemudian masing-masing akan diolah dalam jaringan feistel secara sendiri-sendiri. Setiap proses pengolahan akan diulang sebanyak 16 kali, dengan *round key* yang berbeda-beda setiap putarannya. *Round key* dibuat dari key yang dimasukkan oleh pengguna. Karena panjang kunci minimal adalah 64 bit, maka panjang *round key* untuk bagian kiri dan kanan masing-masing akan berukuran minimal 32 bit. Berikut ini merupakan tahapan operasi yang dilakukan pada jaringan feistel :

- Permutasi *plaintext* dengan D-Box atau P-Box milik algoritma DES
- Transformasi *plaintext* dengan bit-bit *round key*
- Substitusi *plaintext* dengan memanfaatkan kotak-S
- Pergeseran bit-bit *plaintext* berdasarkan kotak-S yang dibuat
- Pergeseran bit-bit *round key* dengan *shift table* untuk menghasilkan *round key* yang baru untuk iterasi selanjutnya

Shift table yang digunakan pada algoritma ini merupakan pengembangan dari *shift table* yang dimiliki oleh DES. Pada DES, *shift table* telah didefinisikan di awal, sehingga pergeseran *round key* akan tetap untuk semua proses enkripsi. Namun, algoritma Incognita memanfaatkan key yang dimasukkan oleh pengguna sebagai *seed* untuk melakukan pengacakan pada *shift table*. Pergeseran yang dilakukan pada tiap bit key akan sebesar 1 atau 2 ke kiri, sehingga *shift table* hanya diisi dengan angka 1 atau 2. Namun, urutan dan banyaknya angka tersebut diacak berdasarkan key.

Misalnya, pada percobaan hasil implementasi dari algoritma yang telah dibuat penulis, dimasukan 2 key berbeda : “asdfghjkl1234567890” dan “lorem ipsum dolor sit amet”. Kedua key tersebut akan menghasilkan 2 *shift table* yang berbeda, yaitu :

- Key : “asdfghjkl1234567890”
  Shift table : [2, 2, 2, 1, 1, 2, 1, 2, 1, 1, 2, 1, 2, 2, 2, 2]
- Key : “lorem ipsum dolor sit amet”
  Shift table : [2, 2, 2, 1, 2, 1, 1, 1, 1, 2, 2, 1, 2, 1, 1, 2]

3.4. Kotak-S

Di dalam jaringan feistel yang digunakan untuk mengolah *plaintext* pada algoritma Incognita, digunakan kotak-S yang berfungsi sebagai acuan dalam melakukan substitusi dan pergeseran (*shift*). Kotak-S pada Incognita tidak didefinisikan di awal, melainkan dibangun berdasarkan key yang dimasukkan oleh pengguna. Sama seperti proses pembuatan tabel permutasi dan *shift table*, pertama-tama dialokasikan terlebih dahulu angka-angka dengan rentang tertentu ke dalam sebuah tabel. Tabel kotak-S terdiri dari 4 buah *list* berisi angka 1 sampai 16 yang diacak berdasarkan key yang dibuat.

Untuk kotak-S iterasi pertama, angka-angka di dalam tabel akan dibuat berdasarkan *seed* berupa key asli. Untuk kotak-S pada iterasi 2, bit-bit key akan diputar (*rotation*) ke kiri sebanyak 1 buah untuk menghasilkan nilai yang berbeda, sehingga pembangkitan kotak-S dilakukan dengan menggunakan *seed* yang jauh berbeda. Hal ini dilakukan untuk iterasi-iterasi berikutnya, sehingga dihasilkan kotak-S yang unik untuk setiap iterasi. Untuk memudahkan pemahaman, berikut merupakan penggambaran proses pembangkitan kotak-S :

- Kotak 1, key “asdfghjkl1234567890” :
  Bit key : 1000011011001...011000001110
  Seed : 3006258055010926047890037408676437398595761678
  Kotak-S yang dihasilkan : [8, 15, 12, 4, 2, 14, 7, 1, 13, 5, 0, 3, 6, 9, 11, 10]
- Kotak 2 :
  Bit key : 0000110110011...110000011101
  Seed : 30352533919801257154693093955489425166053686l
  Kotak-S yang dihasilkan : [14, 0, 4, 1, 5, 2, 6, 8, 10, 13, 7, 15, 12, 3, 11, 9]
- Kotak 3 :
  Bit key : 0001101100111...100000111010
  Seed : 607050678396025143093861879109788503321073722
  Kotak-S yang dihasilkan : [11, 15, 5, 3, 0, 1, 6, 13, 4, 7, 2, 8, 12, 9, 10, 14]
- Dan seterusnya, untuk kotak-kotak berikutnya

4. Eksperimen dan Analisis Hasil

4.1. Implementasi

Implementasi dari Incognita menggunakan bahasa Python. Program menerima masukan berupa teks, kunci, mode, dan jenis proses dari pengguna. Mode yang dapat digunakan antara lain adalah ECB, CBC, dan Counter Mode. Sedangkan jenis proses adalah enkripsi atau dekripsi. Program kemudian akan menghasilkan keluaran berupa *plaintext* atau *ciphertext*, tergantung dari jenis proses yang dilakukan.

Dalam melakukan konversi antara bit, byte, hex, dan string; dalam melakukan operasi xor; dan dalam melakukan pengacakan untuk menentukan permutasi dan sebagainya, program menggunakan modul-modul bawaan Python seperti math, random, dan itertools, serta modul numpy. Program sendiri dibagi menjadi tiga modul, yaitu:

- Modul utama, **main.py**, yang mengandung kode untuk proses pembagian teks menjadi blok-blok dan melakukan proses enkripsi atau dekripsi kepada semua blok secara iteratif
- Modul algoritma, **algo.py**, yang mengandung algoritma Incognita dan melakukan proses enkripsi atau dekripsi kepada sebuah blok
- Modul konversi, **conversions.py**, yang mengandung operasi-operasi untuk melakukan konversi dari bit ke byte, byte ke string, string ke byte, byte ke bit, byte ke hex, dan hex ke byte.

4.2. Pengujian

Proses pengujian dilakukan untuk memastikan bahwa program telah berjalan sebagaimana mestinya dan menghasilkan keluaran yang diinginkan. Pengujian menggunakan kunci ‘kuncites’ dan *plaintext* berupa potongan teks dari sebuah novel sebagai berikut:

> "You're lovely, but you're empty," he went on. One couldn't die for you. Of course, an ordinary passerby would think my rose looked just like you. But my rose, all on her own, is more important than all of you together, since she's the one I've watered. Since she's the one I put under glass. Since she's the one I sheltered behind a screen. Since she's the one for whom I killed the caterpillars (except the two or three for butterflies). Since she's the one I listened to when she complained, or when she boasted, or even sometimes when she said nothing at all. Since she's my rose."

Hasil keluaran *ciphertext* program (dalam bentuk HEX) adalah sebagai berikut:

Tabel 1. Hasil Enkripsi oleh Program

ECB	dd684469aaa528bd1b1a3c40ae448f1f965a9fe97f4891c303edfcb45ebdb67014d73c 0d2061374252e14e60708ede8a4f4cf7fdc2dc36f5e18d14786dd11f1e43a8ed4055a6 df5c6c31ce7844b268e8ae2ece1bd4b3a237ef6c72cb025b39f78641587b903c14c27d 98e32984a25f42547ea9975e43c164c57e0409fa8f5c66a200361d33f586faa01e8eb9 425bb960736228de7de2a926334bd8ed6ec8d8cc3642812b7139cbd5644206b5d4f849 00bf946711780e45d42a61ccc9c3b9860865a97f02b392828e0867cfe590838090c01e 7d118b5b2c64db61ff86b392b77728934c0c47d279f450dd2951a00fd8eb20d511a3a9 08443cf70bcc7cc4f2dbe98c5c1f6ed3a19bdf14dcb960e8f18763b32977337c8e8f00 1e7638a8302b420c0458c6ec2660ec2173d824ef8456cd729832aebae49dcb58e297c7 8ff68c4ae577b1625a580f3076e699e30e2cbbd9a02f87935efa3891d75f3a18df0e90 11a3628fc4f624ca986f64a9d7367e0177dc733de0e056c6974f49b3e26f8ff7ad8ee3 abc38133f3ab4b1e746c0a3da1ab13a80d47694ee6988039916b39e6be182950f02e83 515d799c15e6312f84f162da91f1104f38d0cd0c8dada147008bf4f728934c0c47d279 f450dd2951a00fd8ebb5518789b155172517c1844e79b7a45d66613166167e47f15f1c 0f11647038436292b74323240d74957516de32e0fbb3f1f4840342cb7e1d272fcb53c5 1f60d9e8e7ca381649f2be0e25ee23b5176416bcd5180759fd23955dad5037148811ed a3287ed81ff3bacdd03d86bdb66990704a6334343be74e0ab36603072370d3af6cc5bc bd0856444399a110f0f4afc326
CBC	dd684469aaa528bd1b1a3c40ae448f1f0c2631cbd1338109e9d67229ea6df14dbda595 d69db788ddc1af0c2b6ff42e3d1ba9fe73845c909933dfce00ef96dfee7769902c7521 464c4082840debedd64f26e52f87557a9b8f5a195a08ab1fd2244eadc2e09cf4a96384 1dceb9c24262e5242781b5882eedb25e067b31d63776fdd0aa69d861610316ca26d1aa bc61ded48365264507fb3664ffff3296ed5a68caf31d4838c789909672fe5426d1c411 ea338591983b526a1ff5099c34a4811a0cf8b606624f2633b9b8f06d8a40af32cc65d7 95ef8556bfe11a0b7ae406017d5fd32c6beeb3dd0b3bca509f0200c3a2a653ea3837c8 aec41ad3abe8df9da2ea49a1ea4ad331d6ed18f226607ed69c1d517cdb064dea3c960f 8e3e29eddb7fe386e361fd5dbd34b03ad11066c9d2fda721cea8044d26496e00b8f530 3228b6a9184c3656b62ab236952dbcb834e7e4b10692f5507ac1be12b194af19fa457c c2236f8b191259a51a733a05143adb281450a8001e01722e8efadd6ede210a83585cdb 2267e03d0fb3cab6810e824180638fb7f9ad7087a194e6f8497127033893fc796460b9 6c32760c8d6ad3bc6c6eed83ea37cf8cf321ddfec6d9ddd296e8471573fbc45f11a71b 81c5e9068fb1051b476bf9217ba5d47146bea786025e53460572a117facaf5c33b15d8 ba31885265296862eb2c8110241424e4b76656354b9fd20813e94e2526eeb2371335f5 3c72f9f00aeefc8dc4d4d425a322f9c71deea12ce0d270ba078c9a3cd4184ecde43287 d94843f3dd9b457e932df6ad6fb98f906517a9b4a032fee3be2123203e0cbd7a98f65f 5ef12335b85b517aebe0fec944
Counter	70e915d7a13d2bd4d17a5af4e956ce8572ac09a2ef32b3db4d704875e35dc3584680da d6c0673cf08f7107f62b41c0c567a598bed63ab6f1076b446c6e09dc047a5095c0e436 279387270fe6a36f81826d2d8ef7f87da1800b68194ca723d61b3e860a83d6340fb4d3 5e0fec2a1aace025bb15abdb3ec2a34a0719753d08b56c3fd64ee6726cb405ce693838 dc4406956bbe160dae2c5c8969fd1128b44f15d20d8785fe5928fb7e8f2d7b39074f0e 8244efc356de2476f233a85730390d17932c5ecbe77438b003c72f7a00996f0c9b3a2e 8c46ec7877896cbb170ca6715a9a2cc146a615379227965c3b2a1d0a61fb69a2432a8a 725fee668e0b3e26183cae528429c588a82d84c67dc1a4728e968455ed12e293f9898c 476094227fc2d30d0d941ddac0b039aac035f4d2abaca5cc4eb518b1c2f1b6e407358c 54abeddc4814ce9fc6d2a61dc08435c7dee9278e8005aeddf9d5a382c80c3bc11ad925 db031a9009dae3e01ec5985e4abb6ec185845bb60fb2e1bd8a804e7f4b7163c1d94514 8d51e513e8bb41c178e23658c20f9d40b643058bf07bc965a9d43778d95a9c0ad205f8 43e0ae689278f65e93a033dc5da01743ccf27ce029b88452fafd78bb0cc1c6ee16b397 44c76eb10ad92b419b49a2c3628eec56de34b89b1ef47c0c8f5dce4df678e48e09891e 78315cdf0dd556bf0657a2a4529f6c89173e7e9d0bc8b5d04d4c6f3de29d7bf17cbee6 1f8ac7b7de6f70e111e38c56b429268c4ada5286c2dc574866e0f82aa977b73b0b8598 84c5ef7ed954e2f50fe7677d3f14de14bd56dc5b2969e3ca66b67f9bbb2c8ad7f816ef

	33ec5ca0ca0beb6f4dcc7d925f

Dapat dilihat bahwa ketiga mode menghasilkan *ciphertext* yang berbeda-beda. Kemudian dilakukan proses dekripsi dari ketiga *ciphertext* di atas menjadi bentuk *plaintext* semula. Ketiga algoritma berhasil mengembalikan *plaintext* yang sama dengan semula.

4.3. Analisis Perubahan Kecil pada Ciphertext

Dilakukan analisis terhadap pengaruh perubahan kecil pada *ciphertext*, dengan cara mengganti sebuah byte pada *ciphertext*. Untuk pengujian ini, digunakan sebaris *plaintext* sebagai berikut.

It is only with the heart that one can see rightly; what is essential is invisible to the eye.

Dari HEX *ciphertext* yang dihasilkan ketiga algoritma, diubah satu byte pada indeks ke-100, dengan menambahkan satu bit kepada byte tersebut.

Tabel 2. Perubahan Kecil pada *Ciphertext*

ECB	c07314a59220b43e1183e686579c749fde77bcad211098c2f767e65484e3affe42e0903 7365b61149adf15d802f0eaa9988ae160de1011c6e548ae0d41de6f615916587a658747 1c40c3cc3d4126ba5cdd7284479bdfd02ab285bd4868c0a1d3
CBC	c07314a59220b43e1183e686579c749ff877fe8f92ef21dffd97c6f1add25d7ec0c3b87 8a98a84cec088e75fc597ce7b657249429a9c32624572454690ff2cb6e9993db7e6d358 14926aa1c01d7486b821231eab87bb23ffcf41208c0e61b055
Counter	1bc45acbf56f219ad16c0ce6ec5b8a8564b118a2fe38a78e4b351c78ef59974e04c7dad dc4296be6846007eb2d08889144b2c6bec23da2e9436c10382b1ec604344293d3a86f21 95896e41dfac3c8b8f743adde6bb7db4864e270e51ab639749

Kemudian dilakukan dekripsi dari ketiga *ciphertext* tersebut. *Plaintext* yang dihasilkan dari dekripsi adalah sebagai berikut:

Tabel 3. Hasil Dekripsi dengan Perubahan Kecil pada *Ciphertext*

ECB	It is only with the heart that o¬nãa¨E)dözw,dly; what is essential is invisible to the eye.
CBC	It is only with the heart that oìlTbálø÷gé$sye±rly; what is dssential is invisible to the eye.
Counter	It is only with the heart that one can see rhghtly; what is essential is invisible to the eye.

Berdasarkan *plaintext* yang dihasilkan, dapat dilihat bahwa perubahan sebuah byte pada satu blok *ciphertext* akan menyebabkan perubahan pada *plaintext* di blok yang sama pada mode ECB. Sedangkan untuk mode CBC, blok *plaintext* yang mengalami perubahan adalah blok yang sama dan blok berikutnya. Untuk mode Counter, yang mengalami perubahan hanya byte yang bersangkutan.

4.4. Analisis Perubahan Kecil pada Plaintext

Dilakukan analisis terhadap pengaruh perubahan kecil pada *plaintext*, dengan cara mengganti sebuah byte pada *plaintext*. Untuk pengujian ini, digunakan sebaris *plaintext* pada bagian sebelumnya (subbab 4.3) dengan perubahan sebagai berikut.

```
4974206973206f6e6c79207769746820746865206865617274207468617420f6e652063616e2073
756520726768746c793b207768617420697320657373656e7469616c20697320696e7669736962
6c6520746f20746865206579652e
```

Hasil enkripsi pada ketiga mode dari *plaintext* di atas adalah sebagai berikut.

Tabel 4. Hasil Enkripsi dengan Perubahan Kecil pada *Plaintext*

	Plaintext Asli	**Plaintext dengan Perubahan**
ECB	c07314a59220b43e1183e686579c749fde 77bcad211098c2f767e65484e3affe42e0 9037365b61149adf15d801f0eaa9988ae1 60de1011c6e548ae0d41de6f615916587a 6587471c40c3cc3d4126ba5cdd7284479b dfd02ab285bd4868c0a1d3	c07314a59220b43e1183e686579c749fde 77bcad211098c2f767e65484e3affe4a64 107155df617696ec1fc805d4e8e1988ae1 60de1011c6e548ae0d41de6f615916587a 6587471c40c3cc3d4126ba5cdd7284479b dfd02ab285bd4868c0a1d3
CBC	c07314a59220b43e1183e686579c749ff8 77fe8f92ef21dffd97c6f1add25d7ec0c3 b878a98a84cec088e75fc497ce7b657249 429a9c32624572454690ff2cb6e9993db7 e6d35814926aa1c01d7486b821231eab87 bb23ffcf41208c0e61b055	c07314a59220b43e1183e686579c749ff8 77fe8f92ef21dffd97c6f1add25d7eecc5 b82cee8ea28cd09b6667401fcc3259f4c9 1c9d9433225471c456d4ff2ebfdc9dbdf9 c5d75f6682692bc05d1884d004379ef182 37019cd270a9bc8e2db23d
Counter	1bc45acbf56f219ad16c0ce6ec5b8a8564 b118a2fe38a78e4b351c78ef59974e04c7 daddc4296be6846007eb2c08889144b2c6 bec23da2e9436c10382b1ec604344293d3 a86f2195896e41dfac3c8b8f743adde6bb 7db4864e270e51ab639749	1bc45acbf56f219ad16c0ce6ec5b8a8564 b118a2fe38a78e4b351c78ef59974e04c7 daddc4296be6946007eb2c08889144b2c6 bec23da2e9436c10382b1ec604344293d3 a86f2195896e41dfac3c8b8f743adde6bb 7db4864e270e51ab639749

Berdasarkan *ciphertext* yang dihasilkan, dapat dilihat bahwa perubahan sebuah byte pada satu blok *plaintext* akan menyebabkan perubahan pada *ciphertext* di blok yang sama pada mode ECB. Sedangkan untuk mode CBC, blok *ciphertext* yang mengalami perubahan adalah blok yang sama dan semua blok yang ada berikutnya hingga akhir teks. Untuk mode Counter, yang mengalami perubahan hanya byte yang bersangkutan.

4.5. Analisis Perubahan Kecil pada Kunci

Dilakukan analisis terhadap pengaruh perubahan kecil pada kunci, dengan cara mengganti sebuah byte pada kunci. Untuk pengujian ini, digunakan sebaris *plaintext* pada bagian sebelumnya (subbab 4.3). Sedangkan kunci yang digunakan untuk enkripsi adalah ‘kuncites’. Kunci ini kemudian diberi perubahan sebagai berikut sebelum melakukan dekripsi.

```
6b756e6379746573
```

Perbandingan *plaintext* asli dengan *plaintext* hasil dekripsi adalah sebagai berikut.

Tabel 3. Hasil Dekripsi dengan Perubahan Kecil pada *Ciphertext*

Asli	It is only with the heart that one can see rightly; what is essential is invisible to the eye.
ECB	ÍÚPú¨Ð{q èÿ1=Ì°¬Y¦ÉÿñgÊäÒ:] s[GMÂ@N6L[ÛôëTÐ

	í¤z£â5'|W·Â´Ïë¾²FAcMÑÞm`Ãña×7RÜ9r
CBC	ÞØºþÕE/¬ÌM·sê ·ô'£½¨ÈZ6D<Aî/8^ÎÃ¿B 9¾«fä8w0, ¥HÚ³K6O ÐTeâ é²UU/*û#¹ s[GMÂ@N6L[ÛôëTÐ
Counter	_Na7KÞvP6# iØºþÕE/¬ÌM·sê ¹bÝÅÐñ/kùki?Pûz2 9¾«fä8w0, ÐTeâ

Berdasarkan hasil enkripsi dan dekripsi dengan perubahan byte pada kunci, dapat dilihat bahwa perubahan kecil pada kunci akan menyebabkan kegagalan secara menyeluruh dalam proses dekripsi.

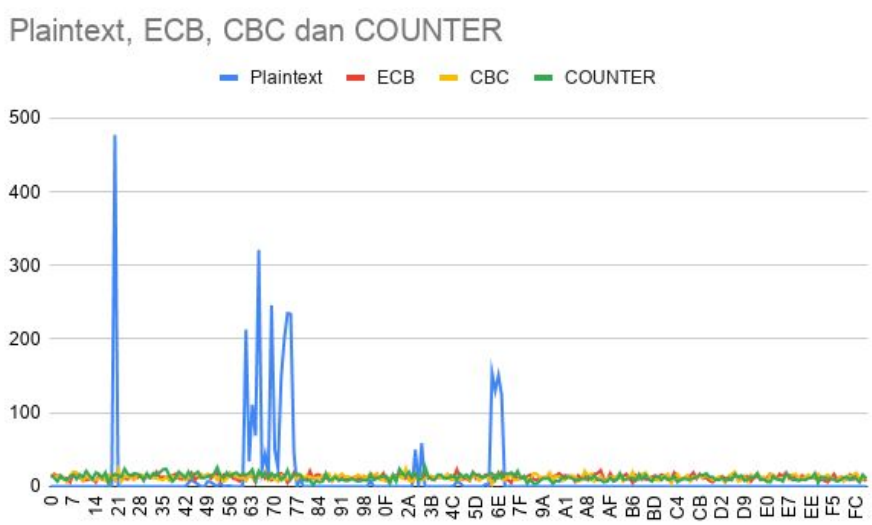
4.6. Analisis Keamanan

Berdasarkan pengujian yang dilakukan terhadap perubahan pada *ciphertext*, *block cipher* Incognita tergolong aman, karena terdapat perubahan yang sangat jelas terlihat pada *plaintext* yang dihasilkan sehingga mudah untuk mengetahui jika suatu pesan telah terkena serangan. Enkripsi juga akan menghasilkan *ciphertext* yang sangat berbeda untuk setiap byte yang diubah pada *plaintext*, dengan pengaruh terhadap blok alih-alih hanya terhadap byte yang bersangkutan.

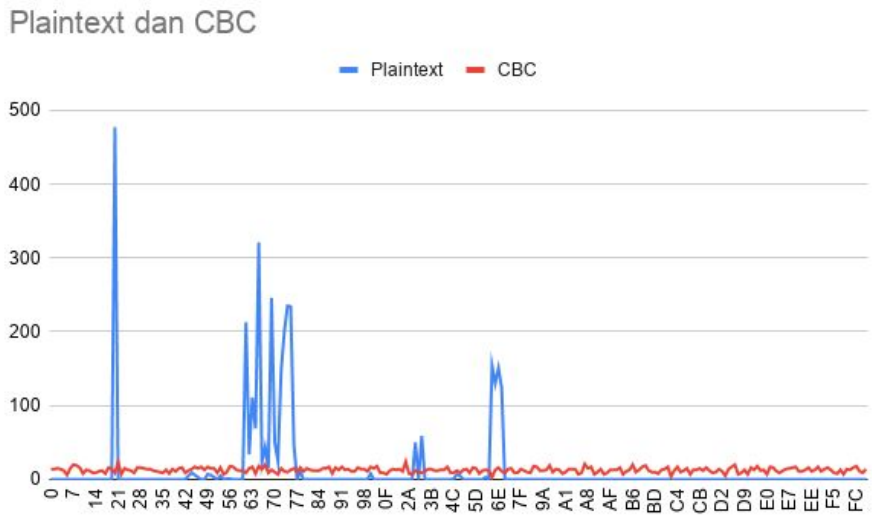
Bagi kriptanalis, untuk melakukan serangan dengan melakukan *brute force*, kemungkinan kunci minimal adalah sebanyak 2^{64} , dengan jumlah maksimal yang tidak terbatas tergantung pada panjang kunci yang dimasukkan oleh pengguna, sehingga kemungkinan kunci pada Incognita paling tidak lebih banyak daripada algoritma DES, GOST, dan AES. Dari pernyataan ini, dapat disimpulkan bahwa algoritma Incognita relatif aman terhadap *brute force attack*.

Sebagai indikator yang jelas untuk mengukur *confusion* pada Incognita, perubahan kecil pada tingkat byte dari kunci akan membawa perubahan yang sangat besar dalam proses enkripsi dan dekripsi, dan menunjukkan bahwa terdapat penyamaran yang baik antara hubungan *plaintext* dengan kunci.

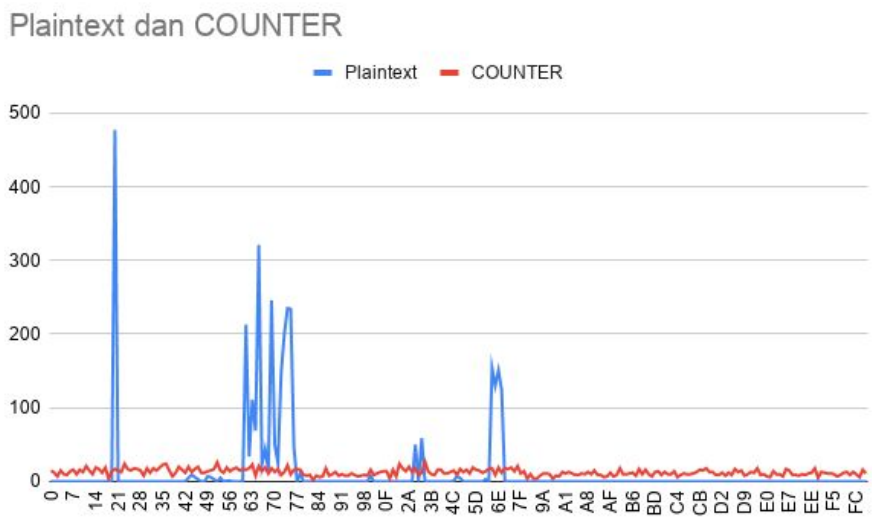
Sedangkan untuk mengukur *diffusion* pada Incognita, dapat dilakukan analisis distribusi karakter dalam bentuk bigram HEX pada *plaintext* dan *ciphertext*. Distribusi dari karakter dapat dilihat dengan grafik analisis frekuensi bi-gram pada gambar-gambar di bawah ini.



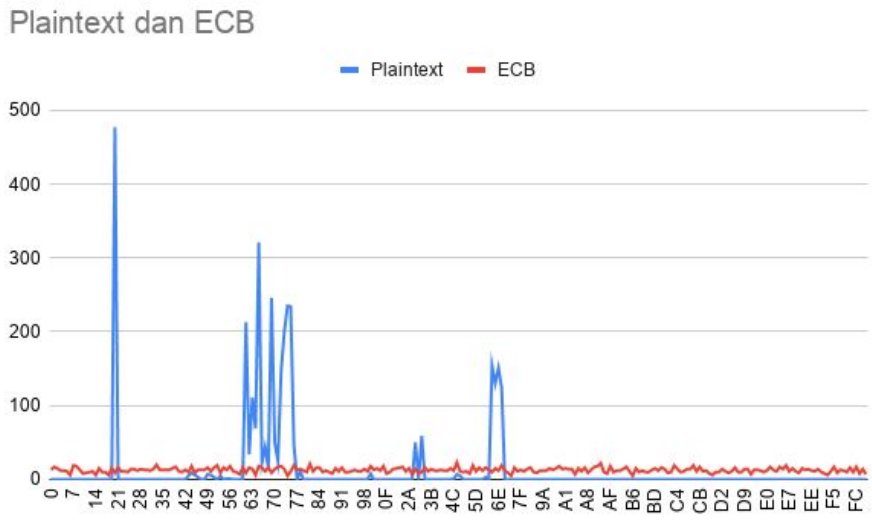
Gambar 3. Grafik analisis frekuensi kemunculan bigram pada *plaintext* dan *ciphertext*



Gambar 4. Grafik analisis frekuensi kemunculan bigram pada *plaintext* dan *ciphertext* hasil enkripsi dengan mode CBC



Gambar 5. Grafik analisis frekuensi kemunculan bigram pada *plaintext* dan *ciphertext* hasil enkripsi dengan mode Counter



Gambar 6. Grafik analisis frekuensi kemunculan bigram pada *plaintext* dan *ciphertext* hasil enkripsi dengan mode ECB

Dapat dilihat dari frekuensi kemunculan karakter pada setiap *ciphertext* bahwa distribusi bigram berbeda jauh dengan *plaintext*. Hal ini menunjukkan bahwa Incognita sudah memiliki *diffusion* yang baik.

5. Kesimpulan dan Saran Pengembangan

Incognita merupakan algoritma *block cipher* "baru" yang banyak menerapkan fungsi pengacakan sebagai implementasi konsep *confusion* dan *diffusion*. Proses pembangunan setiap tabel yang digunakan pada Incognita sangat tergantung dengan key yang dimasukkan oleh pengguna. Dengan menggunakan tabel-tabel (tabel permutasi, *shift table*, dan kotak-S) yang unik tergantung terhadap key, analisis terhadap *ciphertext* yang dihasilkan juga akan lebih sulit, sehingga *ciphertext* Incognita akan lebih sulit untuk didekripsi oleh kriptanalis tanpa mengetahui kunci yang digunakan.

Sebagai saran pengembangan Incognita (*future works*), karena dasar inspirasi Incognita adalah algoritma DES, GOST, dan AES yang memiliki operasi yang relatif sedikit, operasi yang dilakukan di dalam jaringan feistel masih tergolong sedikit. Walaupun hal ini membuat proses enkripsi dan dekripsi pada Incognita menjadi lebih cepat, kompleksitas algoritma juga menjadi lebih rendah daripada algoritma-algoritma *block cipher* baru yang diusulkan beberapa tahun belakangan ini. Dengan demikian, perlu ditambahkan operasi ringan seperti penambahan modulo, pergeseran *plaintext*, dan sebagainya.

6. Daftar Referensi

[1] Munir, Rinaldi. 2020. Slide Kuliah IF4020 Kriptografi: Kriptografi Modern

[2] Munir, Rinaldi. 2020. Slide Kuliah IF4020 Kriptografi: Review Beberapa Block Cipher dan Stream Cipher

[3] Salih, Hilal Hadi; Sadiq, Ahmed Tariq; Frhan, Alaa K. Proposal of New Block Cipher Algorithm

[4] Cruz, Bryan F.; Domingo, Keinaz N.; De Guzman, Froilan E.; Cotiangco, Jhinia B.; Hilario, Christopher B. 2017. Expanded 128-bit Data Encryption Standard

[5] Knuth, Donald. The Art of Computer Programming: Sorting and Searching. Vol. 3, Second Edition.

Acknowledgments

Pertama-tama, kami mengucapkan terima kasih yang sebesar-besarnya kepada Tuhan Yang Maha Esa atas segala berkat-Nya yang memungkinkan kami dalam melakukan proses perancangan dan pembuatan algoritma *block cipher* baru yang kami usulkan pada dokumen ini. Kami juga mengucapkan terima kasih kepada dosen pengajar mata kuliah Kriptografi kami, Dr. Ir. Rinaldi Munir, MT., atas segala ilmu yang telah beliau berikan sebagai dasar kami untuk melakukan proses perancangan dan implementasi algoritma. Ada banyak pihak lain yang juga memiliki peran besar dalam pembuatan dokumen ini, seperti anggota keluarga kami yang selalu memberikan dukungan baik fisik maupun mental selama proses pengerjaan, para penulis buku dan *paper* yang menjadi acuan tambahan dalam merancang dan mengimplementasi algoritma, serta teman-teman kami yang setia mendukung dan membantu kami di setiap kesulitan yang kami hadapi, baik di dalam proses pengerjaan maupun di luar proses pengerjaan.