PEMBANGKIT KEY POLYALPHABETIC CIPHER PADA KRIPTOGRAFI SIMETRI MENGGUNAKAN JAVA

Nuniek Fahriani, Yoedo Agung Suryo , Putri Aisyiyah R. Devi

Abstrak: Kriptografi adalah usaha untuk mengirim pesan rahasia ke penerima dengan menggunakan sistem kode untuk membuat pesan tersebut tidak bisa dipahami oleh pihak ketiga. Didalam kriptografi ada dua macam proses yaitu : proses enkripsi dan proses deskripsi. Enkripsi modern berbeda dengan enkripsi konvensional. Enkripsi modern sudah menggunakan komputer untuk pengoperasiannya, berfungsi untuk mengamankan data baik yang ditransfer melalui jaringan komputer maupun yang bukan. Hal ini sangat berguna untuk melindungi kerahasiaan, integritas data, autentikasi dan non-repudiasi. Salah satu metode yang digunakan yaitu kriptografi simetri atau algoritma simetri. Algoritma ini sering disebut dengan algoritma klasik karena memakai kunci yang sama untuk proses enkripsi dan deskripsi. Ada berbagai serangan keamanan enkripsi untuk dapat mendekrip ciphertext. Salah satu bentuk metode yang dapat mengatasi keamanan data adalah menggunakan pembangkit key polyalpabetic cipher, karena dibuat dari sejumlah cipher abjad-tunggal, masing – masing dengan kunci yang berbeda.

Kata Kunci: Kriptogafi, Venam, Polyalphabetic, Cipher, Enkripsi, Deskripsi

Kriptografi secara umum adalah ilmu dan seni untuk menjaga kerahasiaan berita atau informasi (Sneider, 1996). Pengertian *kriptografi* yang lain adalah ilmu yang mempelajari teknik–teknik matematika yang berhu-bungan dengan aspek keamanan in-for-masi seperti kerahasiaan data, keabsahan data, integritas data, serta autentikasi data (Knused, 1994).

Ada empat tujuan mendasar dari ilmu *kriptografi* yang juga merupakan aspek keamanan informasi, yaitu :

1. Kerahasiaan adalah layanan yang digunakan untuk menjaga isi dari informasi dari siapapun kecuali yang memiliki otoritas atau kunci rahasia untuk membuka informasi yang telah disandi
2. Integritas data adalah berhubungan dengan penjagaan dari perubahan data secara tidak sah. Untuk menjaga integritas data, sistem harus memiliki kemampuan untuk mendeteksi manipulasi data oleh pihak-pihak yang tidak berhak. Antara lain, penyisipan, penghapusan, dan pensubstitusian data lain kedalam data yang sebenarnya.
3. Autentikasi adalah berhubungan dengan identifikasi/pengenalan baik secara kesatuan system maupun informasi itu sendiri. Dua pihak yang saling berkomunikasi harus saling memperkenalkan diri. Informasi yang dikirimkan melalui kanal harus diautentikasi keaslian, isi datanya, waktu pengiriman.
4. Non repudiasi atau nirpenyangkalan adalah usaha untuk mencegah terjadinya penyangkalan terhadap pengiriman/terciptanya suatu informasi oleh yang mengirim-kan/membuat.

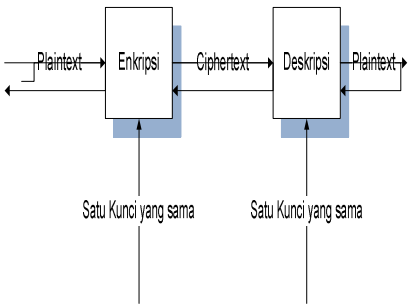
Didalam kriptografi ada *algoritma* yang dipakai. Ditinjau dari asal usulnya, kata *algoritma* mempunyai sejarah yang menarik. Kata ini muncul didalam kamus *Webster* sampai akhir tahun 1957. Kata *algorism* mempunyai arti proses perhitungan dalam bahasa arab yang terkenal yaitu Abu Ja'far Muhammad Ibnu Musa

Nuniek Fahriani, Yoedo Agung Suryo , dan Putri Aisyiyah R. Devi adalah Dosen Teknik Informatika Universitas Muhammadiyah Gresik

Al-Khuwarizmi (al-khuwarizmi dibaca oleh orang barat sebagai algorism). Kata *algorism* lambat laun berubah menjadi *algorithm* (Ariyus Doni, 2008:43).

Kriptografi bertujuan untuk menjaga kerahasiaan informasi atau data supaya tidak dapat diketahui oleh pihak ketiga yang tidak berhak (*unaunthorized person*) dengan meng-gunakan *kriptografi* data asli yang dikirim diubah kedalam bentuk data tersandi. Suatu data yang tidak disandikan disebut *plaintext*, sedangkan data yang telah tersandikan disebut *ciphertext*. Proses yang dilakukan untuk merubah *plaintext* menjadi *ciphertext* disebut *enkripsi*. Sebaliknya proses untuk merubah *ciphertext* menjadi *plaintext* disebut *deskripsi*. Jadi, didalam *kriptografi* ada dua macam proses yaitu: proses *enkripsi* dan proses *deskripsi*. Keamanan dari algoritma *kriptografi* tergantung pada bagaimana algoritma itu bekerja. Oleh sebab itu algoritma semacam ini disebut algoritma terbatas. Algoritma terbatas merupakan algoritma yang dipakai sekelompok orang untuk merahasiakan pesan yang mereka kirim. Jika salah satu dari anggota kelompok itu keluar dari kelompoknya maka algoritma yang dipakai diganti dengan yang baru. Jika tidak maka hal itu bisa menjadi ma-salah dikemudian hari.

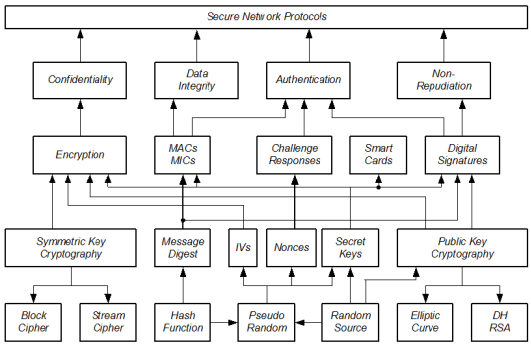
Diperlukan parameter untuk proses kon-versi data yaitu suatu set kunci dan dekripsi data dikontrol oleh sebuah kunci. Secara sederhana dapat digambarkan sebagai berikut :



Gambar 1. Proses *Enkripsi* dan *Deskripsi* Sederhana

Algoritma kriptografi terdiri dari tiga fungsi dasar, yaitu :

1. *Enkripsi* merupakan hal yang sangat penting dalam kriptografi, merupakan pengamanan data yang dikirimkan agar terjaga kerahasiaannya. Pesan asli disebut *plaintext*, yang diubah menjadi kode-kode yang tidak dimengerti. *Enkripsi* bisa diartikan dengan *cipher* atau kode. Sama halnya dengan kita tidak mengerti akan sebuah kata maka kita akan melihatnya didalam kamus atau daftar istilah. Beda halnya dengan *enkripsi*, untuk mengubah *text* asli ke bentuk *text* kode kita meng-gunakan algoritma yang dapat meng-kodekan data yang kita inginkan.
2. *Deskripsi* merupakan kebalikan dari *enkripsi*. Pesan yang dienkripsi dikembalikan ke bentuk asalnya (teks asli), disebut deng-an *deskripsi* pesan. Algoritma yang digu-nakan untuk deskripsi tentu berbeda deng-an algoritma yang digunakan untuk *en-kripsi*.
3. Kunci yang dimaksud disini adalah kunci yang dipakai untuk melakukan *enkripsi* dan *deskripsi*. Kunci terbagi menjadi dua bagian, kunci rahasia (*private key*) dan kunci umum (*public key*)



Gambar 2. Diagram Blok *Kriptografi* Modern

Enkripsi modern berbeda dengan *enkripsi* konvensional. *Enkripsi* modern su-

dah meng-gunakan komputer untuk peng-operasiannya, berfungsi untuk mengaman-kan data baik yang ditransfer melalui jaringan komputer maupun yang bukan. Hal ini sangat berguna untuk me-lindungi kerahasiaan, integritas data, auten-tikasi dan non-repudiasi. Dibawah ini akan di-gambarkan bagaimana *enkripsi* modern saling mendukung satu dengan lainnya.

ALGORITMA SIMETRI

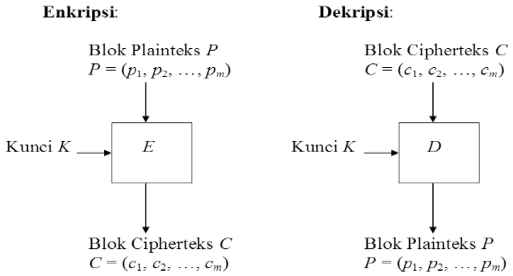
Algoritma ini sering disebut dengan algoritma klasik karena memakai kunci yang sama untuk kegiatan *enkripsi* dan *deskripsi*. Algoritma ini sudah ada sejak lebih dari 4000 tahun yang lalu. Bila mengirim pesan dengan menggunakan al-goritma ini, si penerima pesan harus diberitahu kunci dari pesan tersebut agar bisa mendekripsikan pesan yang dikirim. Keamanan dari pesan yang menggunakan algoritma ini tergantung pada kunci. Jika kunci tersebut diketahui oleh orang lain maka orang tersebut akan dapat me-lakukan *enkripsi* dan deskripsi terhadap pesan. Algoritma simetri menggunakan *stream cipher* yang beroperasi pada bit tunggal dan enkripsi/deskripsi bit per bit (1 bit setiap kali transformasi) atau byte per byte (1 byte setiap kali transformasi). Diperkenalkan oleh Vernam melalui algoritmanya, **Vernam Cipher**. Langkah-langkahnya sebagai berikut : (Knused, 1994)

1. *Enkripsi* pada **Vernam Cipher**
   $C_i = (P_i + K_i) \bmod 2 = P_i \oplus K_i$
   P_i = bit plaintext
   K_i = bit kunci
   C_i = bit ciphertext
2. *Deskripsi* pada **Vernam Cipher**
   $P_i = (C_i + K_i) \bmod 2 = C_i \oplus K_i$

Skema algoritma sandi akan disebut kunci simetri apabila untuk setiap proses en-kripsi data secara keseluruhan digunakan kunci yang sama. Skema ini berdasarkan jumlah data per proses dan alur peng-olahan data didalamnya dibedakan men-jadi dua kelas, yaitu *block cipher* dan *stream cipher*.

BLOCK CIPHER

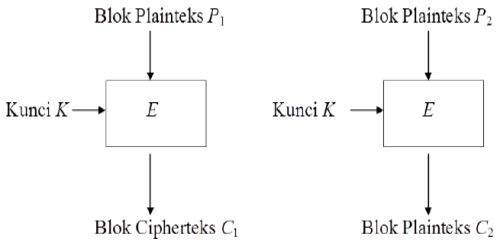
Block cipher adalah skema algoritma sandi yang akan membagi-bagi *text* terang yang akan dikirimkan dengan ukuran ter-tentu (disebut blok) dengan panjang kunci *enkripsi*, dan setiap blok dienkripsi deng-an menggunakan kunci yang sama. Pada umumnya, *block cipher* memproses *text* terang dengan blok yang relative panjang lebih dari 64 bit, untuk mempersulit peng-gunaan pola-pola serangan yang ada untuk membongkar kunci.



Gambar 3. Skema *Enkripsi Block Cipher*

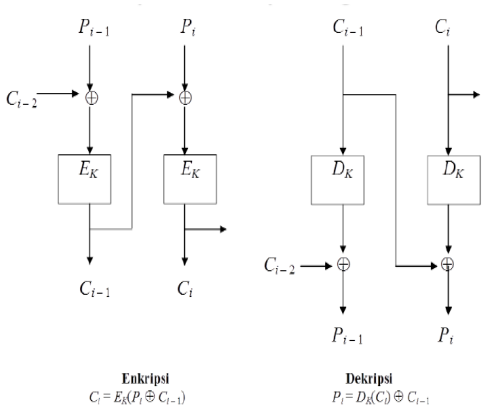
Ada empat macam mode operasi *cipher block* :

1. *Electronic Code Book* (ECB)
   Setiap blok *plaintext* P_i dienkripsi se-cara individual dan independen menjadi blok *ciphertext* C_i. yang dalam hal ini, P_i dan C_i masing-masing blok *plaintext* dan *ciphertext* ke-*i*



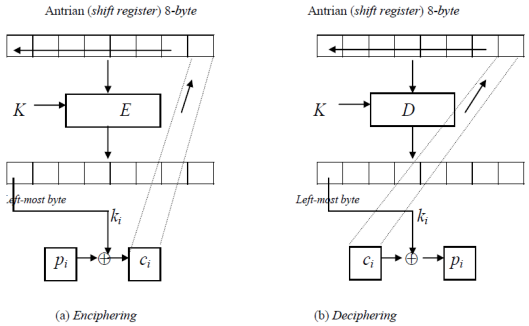
Gambar 4. Skema *Enkripsi* dan *Deskripsi* Mode *ECB*

2. *Cipher Block Chaining* (CBC)
   Tujuan dari *CBC* adalah membuat ketergantungan antar blok. Setiap blok *ciphertext* bergantung tidak hanya pada blok *plaintext*nya tetapi juga pada seluruh blok *plaintexts* sebelumnya. Hasil *enkripsi* blok sebelumnya diumpan balikkan kedalam *enkripsi* blok yang *current*.

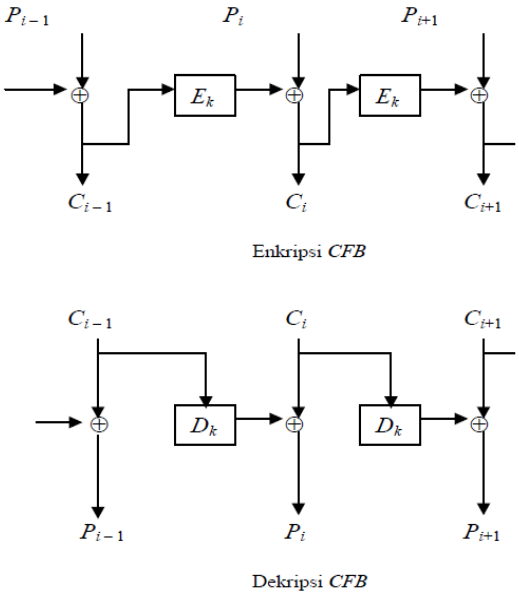


Gambar 5. Skema *Enkripsi* dan *Deskripsi* Mode *CBC*

3. *Cipher Feedback* (CFB)
   Pada mode *CFB* data dienkripsikan dalam unit yang lebih kecil daripada ukuran blok. Unit yang dienkripsikan dapat berupa bit per bit, 2 bit, 3 bit dan seterusnya. Bila unit yang dienkripsikan satu karakter setiap kalinya, maka mode *CFB*nya disebut *CFB* 8 bit. Secara umum *CFB* n-bit mengenkripsikan *plaintext* sebanyak n bit setiap kalinya, yang mana n ≤ m (m = ukuran blok). Mode *CFB* membutuhkan sebuah antrian (*queue*) yang berukuran sama dengan ukuran blok masukan.
   Algoritma *enkripsi CFB* adalah sebagai berikut :
   1. Antrian diisi dengan *initialization vector*
   2. *Enkripsi* antrian dengan kunci k. n bit paling kiri dari hasil *enkripsi* berlaku sebagai keystream (k i) yang kemudian di-XOR-kan dengan n-bit dari *plaintext* menjadi n bit pertama dari *ciphertext*. Salinan n-bit dari *ciphertext* ini dimasukkan kedalam antrian digeser ke kiri menggantikan n-bit pertama yang sudah digunakan
   3. m-n bit *plaintext* berikutnya dienkripsikan dengan cara yang sama seperti pada langkah kedua.



Gambar 6. Skema Mode *CFB* pada 8 Bit



Gambar 7. Skema Mode *CFB* pada n-Bit jika m = n

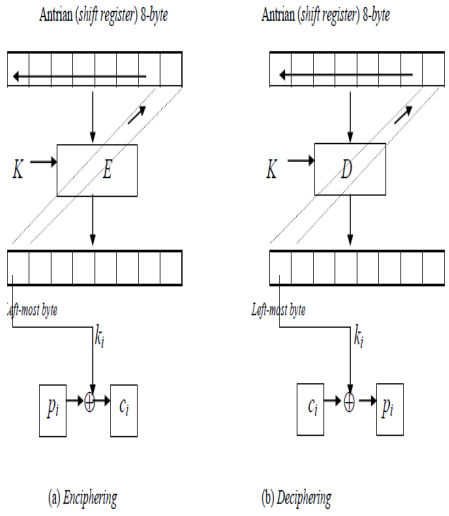
Algoritma *deskripsi CFB* adalah sebagai berikut :

1. Antrian diisi dengan *initialization vector*

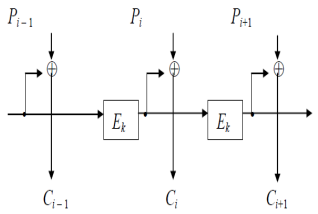
2. *Deskripsi* antrian dengan kunci k. n bit paling kiri dari hasil *deskripsi* berlaku sebagai *keystream* (k i) yang kemudian di-XOR-kan dengan n-bit dari *plaintext* menjadi n bit pertama dari *ciphertext*. Salinan n-bit dari *ciphertext* ini dimasukkan kedalam antrian digeser ke kiri menggantikan n-bit pertama yang sudah digunakan
3. m-n bit *plaintext* berikutnya dideskripsikan dengan cara yang sama seperti pada langkah kedua.

4. *Output Feedback* (OFB)
   Pada mode *OFB*, data dienkripsikan dalam unit yang lebih kecil daripada ukuran blok. Unit yang dienkripsikan dapat berupa bit per bit, 2 bit, 3 bit dan seterusnya. Bila unit yang dienkripsikan satu karakter setiap kalinya, maka mode *OFB* nya disebut *OFB* 8 bit. Secara umum *OFB* n-bit mengenkripsi *plaintext* sebanyak n bit setiap kalinya, yang mana n ≤ m (m = ukuran blok). Mode *OFB* membutuhkan sebuah antrian (*queue*) yang berukuran sama dengan ukuran blok masukan. Tinjau mode *OFB* n-bit yang bekerja pada blok berukuran m-bit.
   Algoritma *enkripsi OFB* adalah sebagai berikut :
   1. Antrian diisi dengan *initialization vector*
   2. *Enkripsi* antrian dengan kunci k. n bit paling kiri dari hasil *enkripsi* dimasukkan kedalam antrian (menempati n posisi bit paling kanan antrian), dan m – n bit lainnya didalam antrian digeser kekiri menggantikan n bit pertama yang sudah digunakan. n bit paling kiri dari hasil enkripsi juga berlaku sebagai *keystream* (ki) yang kemudian di XOR kan dengan n bit dari *plaintext* menjadi n bit pertama dari *ciphertext*
   3. m-n bit *plaintext* berikutnya dienkripsikan dengan cara yang sama seperti pada langkah kedua.

   Algoritma *deskripsi OFB* adalah sebagai berikut :
   1. Antrian diisi dengan *initialization vector*
   2. *Deskripsi* antrian dengan kunci k. n bit paling kiri dari hasil *deskripsi* dimasukkan kedalam antrian (menempati n posisi bit paling kanan antrian), dan m – n bit lainnya didalam antrian digeser kekiri menggantikan n bit pertama yang sudah digunakan. n bit paling kiri dari hasil *deskripsi* juga berlaku sebagai *keystream* (ki) yang kemudian di XOR kan dengan n bit dari *plaintext* menjadi n bit pertama dari *ciphertext*
   3. m-n bit *plaintext* berikutnya dideskripsikan dengan cara yang sama seperti pada langkah kedua.



Gambar 8. Skema Mode *OFB* pada 8 Bit

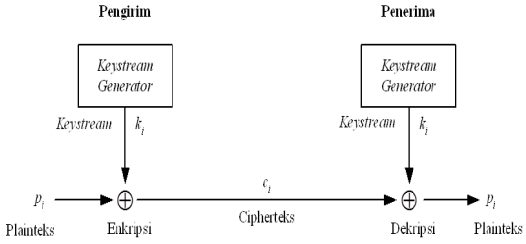


Enkripsi *OFB*

Gambar 9. *Enkripsi* Mode *OFB* n-bit untuk blok n-bit

STREAM CIPHER

Stream cipher adalah algoritma sandi yang mengenkripsikan data persatuan data, seperti bit, byte, nible atau per lima bit (saat data yang dienkripsi berupa data *boundout*). Setiap mengenkripsi satu kesatuan data digunakan kunci yang merupakan hasil pembangkitan dari kunci sebelum.



Gambar 10. Skema *Stream Cipher*

SERANGAN PADA ENKRIPSI

Serangan ini secara umum diklasifikasikan dalam enam kategori. Tujuan dari penyerang dalam semua kasus adalah untuk dapat mendekrip sebuah *ciphertext* baru tanpa informasi tambahan. Yang menjadi tujuan dari penyerang adalah untuk mengekstrak kunci rahasia.

1. Serangan *ciphertext only* adalah salah satu serangan dimana penyerang mendapatkan contoh dari *ciphertext*, tanpa *plaintext* yang berhubungan dengannya. Data ini relative mudah didapatkan dalam banyak skenario, tetapi serangan yang berhasil biasanya sulit dan membutuhkan contoh *ciphertext* yang sangat besar.
2. Serangan *known plaintext* adalah salah satu serangan dimana peyerang mendapatkan contoh *ciphertext* dan *plaintext* yang berhubungan.
3. Serangan *chosen plaintext* adalah salah satu serangan dimana penyerang dapat memilih kuantitas *plaintext* dan kemudian mendapatkan *ciphertext* terenkripsi yang berhubungan.
4. Serangan *adaptive chosen plaintext* adalah kasus khusus dari serangan *chosen plaintext* dimana penyerang dapat memilih secara dinamis dan mengubah pilihannya berdasar dari hasil enkripsi sebelumnya.
5. Serangan *chosen ciphertext* adalah salah satu serangan dimana penyerang dapat memilih *ciphertext* dan mencoba mendapatkan *plaintext* terdekripsi yang berhubungan. Tipe serangan ini biasanya banyak dilakukan pada *public key cryptosystems.*
6. *Adaptive chosen ciphertext* adalah versi adaptif dari serangan diatas. Penyerang dapat memuat serangan dari tipe ini dalam skenario dimana ia memiliki penggunaan bebas dari sebuah *hardware* dekripsi tetapi tidak dapat mengekstrak kunci dekripsi darinya.

CRYPTOGRAPHIC ATTACKS

Pada dasarnya serangan terhadap *protocol kriptografi* dapat dibedakan menjadi dua jenis yaitu :

1. Serangan pasif adalah serangan dimana penyerang hanya memonitor saluran komunikasi. Penyerang pasif hanya mengancam kerahasiaan data
2. Serangan aktif adalah serangan dimana penyerang mencoba untuk menghapus, menambahkan, atau dengan cara lain mengubah transmisi pada saluran. Penyerang aktif mengancam integritas data dan otentikasi juga kerahasiaan.

PEMROGRAMAN JAVA

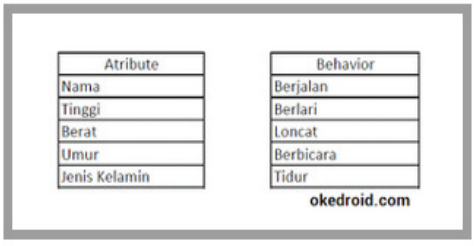
Java adalah nama sebuah bahasa pemrograman yang sangat terkenal. Sebagai bahasa pemrograman, java dapat digunakan untuk menulis program. Sebagaimana diketahui, program adalah kumpulan intruksi yang ditujukan untuk komputer. Melalui program, komputer dapat diatur agar melaksanakan tugas tertentu sesuai yang ditentukan oleh pemrogram (orang yang membuat program). Bahasa java dikembangkan di Sun Microsystems dan mulai diperkenalkan kepada publik pada tahun 1995. Seperti halnya C++, java juga merupakan bahasa yang berorientasi objek. Dengan demikian, java juga memudahkan dalam pembuatan aplikasi yang berskala besar. Sebagai bahasa yang universal, java bisa dijumpai diberbagai platform (linux, Unix, Windows, Mac) hasil kompalasi java yang dinamakan *bytecode* dapat dijalankan di berbagai platform sepanjang di system target memiliki *Java Runtime Environment* (JRE) (Hartono, 1999)

Beberapa keunggulan bahasa pemrograman Java antara lain:

1. Mudah dipelajari
   Java memiliki salah satu keunggulan , yaitu mudah dipelajari oleh semua orang, karena *syntax* (tata bahasa) yang mirip dengan bahasa manusia. Dengan banyaknya artikel atau tutorial yang membahas bahasa pemrograman Java di Internet. Semakin memudahkan kita untuk pemula, untuk mempelajari semua hal yang ada di bahasa pemrograman Java.
2. Merupakan bahasa OOP
   Salah satu alasan ,yang membuat Java begitu populer karena mempunyai konsep bahasa *(OOP) Object Oriented Programming.* Oleh karena itu , untuk mengembangkanya jauh lebih mudah , serta menjaga *system* tetap *modular*, *flexible* and *extensible*.
3. Banyaknya API
   Java memiliki banyaknya *API (Application programming interface) ,* yang siap dikembangkan untuk keperluan para programer Java.

Object Oriented Programming memiliki beberapa konsep yaitu:

1. ***Class*** merupakan sebuah kerangka/ model yang berfungsi untuk tempat menaruh, dan mendeksripsikan *variabel, method* (perilaku) dari sebuah obyek. Penamaan nama class diprogram ,harus sama dengan nama di struktur *file extention* **.java**. Contoh : class binatang , class mobil , class buah
2. ***Object*** merupakan sebuah representasi dan instance dari *Class*. *Object* adalah sebuah inti dan wujud real dari sebuah *Class.* Contoh : dari *class* binatang, kita bisa mengambil obyek, yaitu kucing, ayam, burung.
3. ***Atribute*** merupakan sebuah unsur data yang ada di *class*, Atribut biasanya terdiri dari sebuah data, variabel, *propertie* dan *field*. Contoh dari *class* binatang bntng, class mobil mbl.
4. ***Method*** merupakan sebuah perilaku (*behavior*) dari sebuah *class*. *Method* terbagi menjadi dua jenis, *method void* dan *method* non *void*. *Method void* artinya *method* yang tidak mengembalikan nilai. *Method* non *void* sebaliknya *method* yang mengembalikan nilai. Contoh *method void* seperti : *public static void main(String[] args)*



Gambar 11. *Method* dan *Behaviour*

5. ***Encapsulation*** (Enkapsulasi) adalah suatu mekanisme membungkus suatu data (variabel), agar tidak dapat di akses oleh class lain, dengan menggunakan modifier private atau protected (untuk Class Turunan). Didalam konsep ini beberapa variabel akan disembunyikan oleh class lain, dan hanya bisa di akses di main class , dengan menggunakan method modifier public.
6. ***Inheritance*** (Pewarisan) adalah suatu proses dimana, suatu class yang bisa disebut *super class,* dapat mewarisi sifat turunan ke dalam class turunanya yaitu *sub class*. Super class akan mewarisi nilai dari atribut atau behavior ke Class turunananya.
7. ***Polymorphism*** (Banyak bentuk) adalah suatu kemampuan yang dimiliki sebuah method, yang memiliki nama sama, namun dengan perilaku yang berbeda-beda. Kemampuan objek agar melakukan perilaku atau tindakan yang secara konsep sama namun dengan cara yang berbeda-beda.

Modifier

Modifier merupakan sebuah ijin hak akses untuk penggunaan suatu atribut atau method, pada suatu *Package* dan *Class*. Terdapat empat 4 jenis Modifier yaitu public, private, protected, no modifier. Contohnya bisa melihat gambar berikut:

Modifiers Akses	Class Sama	Package sama	Subclass	Package lain
Public	✓	✓	✓	✓
Protected	✓	✓	✓	-
Private	✓	-	-	-
No Modifiers	✓	✓	-	-

Gambar 12. Modifier Akses Java

Keterangan :

- **Public** dapat di akses dari class yang sama, package yang sama, subclass, dan package lain.
- **Protected** dapat diakses dari class yang sama, package yang sama, subclass, tapi tidak dapat diakses dari package lain.
- **Private** hanya dapat di akses oleh class yang sama.
- **No Modifiers** dapat diakses dari class dan package yang sama.

POLYALPHABETIC CIPHER

Teknik *alphabetic cipher* merupakan teknik *kriptografi* yang menggunakan kunci yang berupa kata dan juga menggunakan rumus *Caesar cipher*. Metode kriptografi klasik *polyalphabetic cipher* ditemukan pertamakali oleh Leon Battista pada tahun 1467. Metode ini digunakan oleh tentara AS selama perang sipil Amerika. *Cipher* ini melibatkan penggunaan kunci berbeda. *Polialphabetic cipher* dibuat dari sejumlah cipher abjad-tunggal, masing-masing dengan kunci yang berbeda. Kebanyakan cipher abjad-majemuk adalah *cipher* substitusi periodic yang didasarkan pada periode m.

Plaintext P adalah $P = P_1P_{2....}P_mP_{m+1}$
Maka hasil *ciphertext* hasil enkripsi adalah
$E_k(P) = f_1(P_1)f_2(P_2).....f_m(P_m)f_{m+1}(P_{m+1})$

Salah satu algoritma *polialphabetic cipher* adalah *vigenere cipher* yang ditemukan oleh kriptologi perancis, Blaise de Vigenere pada abad 16. Pada algoritma menggunakan string, misalkan K adalah deretan kunci. Kebanyakan penerapan *polyalphabetic cipher* adalah mengulang kunci *monoalphabetic* selama n periode.
n periode = panjang *plaintext*. Untuk lebih jelasnya ditunjukkan pada Gambar 13.

	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
C1	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
C2	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
C3	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
C4	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
C5	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
C6	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

Gambar 13. Tabel *Polyalphabet*

Keterangan Gambar 13 :
Plaintext = kriptografi
Key = n u n i e k
1 2 3 4 5 6
Ciphertext = xlvxxytlnnm

ANALISA

Bentuk pelayanan keamanan dalam kriptografi adalah autentikasi dan kerahasiaan pesan. Untuk itu William Stalling (Stalling, 2003) menyarankan bahwa dalam men-disain kunci publik hendaknya memenuhi kedua jenis layanan keamanan tersebut, hal ini mengingat bahwa umumnya kunci publik banyak yang didisain hanya me-menuhi layanan kerahasiaan data. Se-dangkan Simmons (simmons, 1993) menyarankan bah-wa dalam mendisain kunci publik, selain harus menjamin keamanan data, sistem kunci publik juga harus efisien, khususnya dalam hal penggunaan ruang kunci, kompleksitas enkripsi dan dekripsi, dan ekspansi pesan.

Secara matematis, proses atau fungsi enkripsi (E) dapat dituliskan sebagai:

$$E(M) = C \qquad (1)$$

dimana:
M adalah plaintext (message) dan *C* adalah ciphertext.
Proses atau fungsi dekripsi dapat dituliskan sebagai:

$$D(C) = M \qquad (2)$$

Keamanan sebuah algoritma yang digunakan dalam enkripsi atau dekripsi bergantung kepada beberapa aspek. Salah satu aspek yang cukup penting adalah sifat algoritma yang digunakan. Apabila kekuatan dari sebuah algoritma sangat tergantung kepada pengetahuan (tahu atau tidaknya) orang terhadap algoritma yang digunakan, maka algoritma tersebut disebut "*restricted algorithm*". Apabila algoritma tersebut bocor atau ketahuan oleh orang banyak, maka pesan-pesan dapat terbaca. Tentunya hal ini masih bergantung kepada adanya kriptografer yang baik. Jika tidak ada yang tahu, maka sistem tersebut dapat dianggap aman (meskipun semu).

Salah satu cara untuk menambah tingkat keamanan sebuah algoritma enkripsi dan dekripsi adalah dengan menggunakan sebuah kunci (*key*) yang biasanya disebut *K*. Kunci *K* ini dapat memiliki rentang (*range*) yang cukup lebar. Rentang dari kemungkinan angka (harga) dari kunci *K* ini disebut *keyspace*. Kunci K ini digunakan dalam proses enkripsi dan dekripsi sehingga persamaan matematisnya menjadi:

$$E_K(M) = (C) \qquad (3)$$
$$D_K(C) = (M) \qquad (4)$$

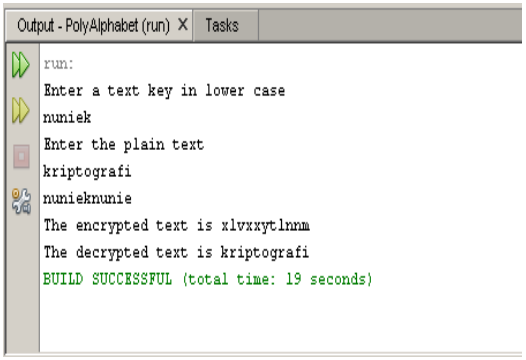
Keamanan sistem yang digunakan kemudian tidak bergantung kepada pengetahuan algoritma yang digunakan, melainkan bergantung kepada kunci yang digunakan. Artinya, algoritma dapat diketahui oleh umum atau dipublikasikan. Usaha untuk memecahkan keamanan sistem menjadi usaha untuk memecahkan atau mencari kunci yang digunakan.

Usaha mencari kunci sangat bergantung kepada *keyspace* dari kunci *K*. Apabila *keyspace* ini cukup kecil, maka cara *brute force* atau mencoba semua kunci dapat dilakukan. Akan tetapi apabila *keyspace* dari kunci yang digunakan cukup besar, maka usaha untuk mencoba semua kombinasi kunci menjadi tidak realistis.

IMPLEMENTASI HASIL PENGUJIAN

Pembangkit key pada kriptografi simetris ini menggunakan *polyalphabetic cipher* yang menggunakan pembangkit satu kunci yang sama yang diimplementasikan menggunakan Java NetBeans IDE 7.1.1 pada operating *windows*. Hasil dari implementasi keamanan data *kriptografi polyalphabetic cipher* sesuai dengan gambar 13 tabel *polyalphabet* dapat dilihat pada Gambar 14.

Penggunaan kunci yang mempunyai panjang sama dengan *plaintext* dan kunci ini hanya digunakan 1 kali. Karena itu, cara enkripsi ini cukup sulit dipecahkan karena kunci hanya digunakan satu kali sehingga tidak ada suatu pola tertentu. Satu-satunya cara melakukan deskripsi adalah dengan mengetahui kunci yang digunakan. Enkripsi dilakukan dengan cara sama dengan enkripsi Caesar, tetapi karena panjang kunci sama dengan *plaintext*, maka setiap huruf pada *plaintext* akan mengalami pergeseran yang berbeda. Banyaknya kunci dari suatu cipher cukup merespons para *cryptanalyst*, akan membutuhkan waktu sampai ditemukannya kunci dari *ciphertext*. Akan tetapi, cara tersebut selalu membutuhkan waktu yang lama.



Gambar 14. Hasil Implementasi *Polyalphabetic Cipher*

PEMBAHASAN

Ada beberapa cara yang dilakukan oleh *ctyptanalyst* dengan melihat frekuensi kemunculan huruf seperti tabel frekuensi kemunculan huruf yang ditunjukkan oleh Tabel 1 dan Tabel 2.

Tabel 1. Frekuensi Kemunculan Huruf Dalam Bahasa Inggris

Karakter	Peluang	Karakter	Peluang
A	0.082	N	.067
B	.015	O	.075
C	.028	P	.019
D	.043	Q	.001
E	.127	R	.060
F	.022	S	.063
G	.020	T	.091
H	.061	U	.028
I	.070	V	.010
J	.002	W	.023
K	.008	X	.001
L	.040	Y	.020
M	.024	Z	.001

Tabel 2. Sepuluh Huruf yang sering Muncul dalam Bahasa Indonesia

Letter	Frequency Kemunculan (%)
A	17.50
N	10.30
I	8.70
E	7.50
K	5.65
T	5.10
R	4.60
D	4.50
S	4.50
M	4.50

Berikut ini adalah *syntax* untuk proses enkripsi pada java programming

```
int index;
        String              key             =
```

```
kunci.getText().replaceAll("[^a-zA-Z]",
"").toUpperCase();
        String kata = masukan.getText();
        if (key.length() == 0) {
            if (kunci.getText().length() >
0) {

JOptionPane.showMessageDialog(null, "Kunci
harus mengandung alphabet!", "Informasi",
JOptionPane.INFORMATION_MESSAGE);
            }             else             {
JOptionPane.showMessageDialog(null,
"Masukkan         kunci!",         "Error",
JOptionPane.ERROR_MESSAGE);
            }
        } else {
            kunci.setText(key);
            if (kata.length() > 0) {
                Proses polyalphabet = new
Proses(key);

keluaran.setText(polyalphabet.enkripsi(kat
a));
            }             else             {
keluaran.setText("");
            }
            modelTabel.setRowCount(0);
            for   (int   i   =   0;   i   <=
(key.length() - 1); i++) {
                dataTabel[0]              =
key.substring(i, i + 1);
                index                     =
Arrays.binarySearch(alphabet,
dataTabel[0]) - 1;
                for (int j = 1; j <= 26;
j++) {                    index = index <
26 ? index + 1 : 1;
                    dataTabel[j]          =
alphabet[index];
                }

modelTabel.addRow(dataTabel); }  }    }
```

Berikut ini adalah *syntax* untuk proses deskripsi pada java programming

```
int index;
        String             key            =
kunci.getText().replaceAll("[^a-zA-Z]",
"").toUpperCase();
        String kata = masukan.getText();
        if (key.length() == 0) {
            if (kunci.getText().length() >
0)                                         {
JOptionPane.showMessageDialog(null, "Kunci
harus mengandung alphabet!", "Informasi",
JOptionPane.INFORMATION_MESSAGE);
            } else {

JOptionPane.showMessageDialog(null,
"Masukkan         kunci!",         "Error",
JOptionPane.ERROR_MESSAGE);
            }
        } else {
            kunci.setText(key);
            if   (kata.length()   >   0)   {
Proses polyalphabet = new Proses(key);

keluaran.setText(polyalphabet.deskripsi(ka
ta));
            }             else             {
keluaran.setText("");
            }
            modelTabel.setRowCount(0);
            for   (int   i   =   0;   i   <=
(key.length()     -     1);     i++)     {
dataTabel[0] = key.substring(i, i + 1);
                index      =      27      -
Arrays.binarySearch(alphabet,
dataTabel[0]);
                for (int j = 1; j <= 26;
j++) {
                    index = index < 26 ?
index + 1 : 1;
                    dataTabel[j]          =
alphabet[index];
                }

modelTabel.addRow(dataTabel);  }   }   }
```

KESIMPULAN

Masing-masing dari 26 cipher disusun secara horisontal dan alphabet normal disusun secara vertikal. Untuk meng-enkripsi sebuah pesan, sebuah kunci di-butuhkan seusai dengan jumlah huruf dari *plaintext*. Biasanya, kunci tersebut merupakan kata-kata kunci yang diulang-ulang sesuai dengan panjang plaintext. Sebagai contoh kata kuncinya adalah *nuniek* dan pesannya adalah "*kriptografi*". Kekuatan dari *cipher* ini adalah adanya banyak huruf *ciphertext* untuk setiap huruf *plaintext*, satu untuk setiap huruf unik dari huruf kunci. Jadi, informasi frekuensi huruf menjadi tidak jelas.

DAFTAR RUJUKAN

Ariyus Dony.(2008).”Pengantar ilmu kriptografi : teori analisis & implementasi” STMIK Amikom. Penerbit Andi

Hartono Jogiyanto. (1999). “Pengenalan computer dasar Ilmu Komputer, pemrograman, system informasi dan intelegensi buatan” Andi Yogkarta

Kadir Abdul. (2014). “Buku Pertama Belajar Pemrograman Java Untuk Pemula” Penerbit Mediakom Yogyakarta

Knused. (1994). “Block Cipher – Analysis Design and Application” Phd Dissertation : Aarhus University

Menezes J. Alfred, Oorschot Van C. Paul. (1996). "The handbook of Applied Cryptography".

Munir Rinaldi. (2006). "Kriptografi" Penerbit Informatika Bandung

Patrick W. Dowd, and John T. McHenry, "Network Security: It's Time To Take It Seriously," *IEEE Computer*, pp. 24-28, September 1998.

Ramachandran Jay.(2002). "Designing security architecture Solution" John Wiley & Sons

Simmons, G.J. (1993). "Contemporary Cryptology : The Science of Information Integrity, New York : IEEE Press.

Sneider Bruce. (1996). "Applied Cryptography : Protocol, Algorithms, and Source Code in C".

Stalling William. (2003). "Cryptography and Network Security : Principles and Practice, 3rd Edition" Pearson Education International

Stephen Northcutt and Judy Novak. (2001) "Network Intrusion Detection: ananalyst's handbook," 2nd edition, New Riders Publishing.

Wahana. (2003). "Memahami model enkripsi dan security data". Andi Offset Yogyakarta

Warwick Ford, and Michael Baum. (1997) "Secure Electronic Commerce: building infrastructure for digital signatures & encryption," Prentice Hall PTR.