UAS

RESUME EBOOK CISSP

Nama : Rahmat Hikmatullah

Nim : 1310652047

Fakultas Teknik

Universitas Muhammadiyah Jember

Domain 1: Access Control 1

EXAM OBJECTIVES IN THIS CHAPTER
- Cornerstone Access Control Concepts
- Access Control Models
- Access Control Defensive Categories and Types
- Authentication Methods
- Access Control Technologies
- Assessing Access Control

INTRODUCTION
The purpose of access control is to allow authorized users access to appropriate data and deny access to unauthorized users

CORNERSTONE INFORMATION SECURITY CONCEPTS
These concepts provide the foundation upon which the 10 domains of the Common Body of Knowledge are built.

1. **Confidentiality, integrity, and availability**
   Confidentiality, Integrity, and Availability are the "CIA triad are the "CIA triad," forms the three-legged
   stool information security is built upon.
   a. Confidentiality
      unauthorized read access to data
   b. Integrity
      Integrity seeks to prevent unauthorized modification of information. There are two types of integrity: data integrity and system integrity.
   c. Availability
      Availability ensures that information is available when needed

2. **Disclosure, alteration, and destruction**
   Disclosure is the unauthorized disclosure of information; alteration is the unauthorized modification of data, and destruction is making systems unavailable.

3. **Identity and authentication, authorization, and accountability**
   a. Identity and authentication
      Identity is a claim: if your name is "Person X," you identify yourself by saying "I am Person X."
   b. Authorization
      Authorization describes the actions you can perform on a system once you have identified and authenticated.
   c. Accountability
      Accountability holds users accountable for their actions
4. **Nonrepudiation**
   Nonrepudiation means a user cannot deny (repudiate) having performed a transaction.
5. **Least privilege and need to know**
   Least privilege means users should be granted the minimum amount of access (authorization) required to do their jobs, but no more

6. **Subjects and objects**
   A subject is an active entity on a data system.
7. **Defense-in-depth**
   Applies multiple safeguards (also called controls: measures taken to reduce risk)

ACCESS CONTROL MODELS

the primary models are Discretionary Access Control (DAC), Mandatory Access Control (MAC), and nondiscretionary access control.

1. Discretionary access controls
   Discretionary Access Control (DAC) gives subjects full control of objects they have been given access
2. Mandatory access controls
   Mandatory Access Control (MAC) is system-enforced access control based on subject's clearance and object's labels.
3. Nondiscretionary access control
   Role-Based Access Control (RBAC) defines how information is accessed on a system based on the role of the subject.
4. Rule-based access controls
   A rule-based access control system uses a series of defined rules, restrictions, and filters for accessing objects within a system.
5. Centralized access control
   Centralized access control concentrates access control in one logical point for a system or organization.
6. Access control lists
   Access control lists (ACLs) are used throughout many IT security policies, procedures, and technologies.
7. Access provisioning lifecycle
   IBM describes the following identity lifecycle rules:
   • "Password policy compliance checking
   • Notifying users to change their passwords before they expire
   • Identifying life cycle changes such as accounts that are inactive for more than 30 consecutive days
   • Identifying new accounts that have not been used for more than 10 days following their creation
   • Identifying accounts that are candidates for deletion because they have been suspended for more than 30 days
   • When a contract expires, identifying all accounts belonging to a business partner or contractor's employees and revoking their access rights
8. User entitlement, access review, and audit
   Access aggregation occurs as individual users gain more access to more systems.
9. Access control protocols and frameworks
   Both centralized and decentralized models may support remote users authenticating to local systems. A number of protocols and frameworks may be used to support this need, including RADIUS, Diameter, TACACS/TACACSþ, PAP, and CHAP.
   RADIUS

ACCESS CONTROL DEFENSIVE CATEGORIES AND TYPES

There are six access control types: Preventive, Detective, Corrective, Recovery, Deterrent, Compensating

a. Preventive
   Preventive controls prevent actions from occurring.
b. Detective
   Detective controls are controls that alert during or after a successful attack.
c. Perbaikan
   Kontrol korektif bekerja dengan "memperbaiki" sistem atau proses rusak.
d. Recovery
   order to restore functionality of the system and organization.
e. Deterrent
   Deterrent controls deter users from performing actions on a system.
f. Compensating
   A compensating control is an additional security control put in place to compensate for weaknesses in other controls.

AUTHENTICATION METHODS

A key concept for implementing any type of access control is controlling the proper authentication of subjects within the IT system

1. Type 1 authentication: something you know
   a. Passwords
      Passwords have been the cornerstone for access control to IT systems.
   b. Password hashes and password cracking
      . Hashing is one-way encryption using an algorithm and no key.
   c. Dictionary attacks
      A dictionary attack uses a word list: a predefined list of words, and then runs each word through a hash algorithm.
   d. Hybrid attacks
      A hybrid attack appends, prepends, or changes characters in words from a dictionary before hashing, to attempt the fastest crack of complex passwords.
   e. Brute-force attacks
      Brute-force attacks take more time but are more effective. The attacker calculates the hash outputs for every possible password.
   f. Rainbow tables
      A rainbow table is a precomputed compilation of plaintexts and matching ciphertexts (typically passwords and their matching hashes).
   g. Salts
      A salt allows one password to hash multiple ways.

2. Type 2 authentication: something you have
   requires that users possess something, such as a token, which proves they are an authenticated user
   a. Synchronous dynamic token
      use time or counters to synchronize a displayed
   b. Asynchronous dynamic token
      Asynchronous dynamic tokens are not synchronized with a central server.

3. Type 3 authentication: something you are
   is biometrics, which uses physical characteristics as a means of identification or authentication.

a. Biometric enrollment and throughput
   Enrollment describes the process of registering with a biometric system
b. Accuracy of biometric systems
   The accuracy of biometric systems should be considered before implementing a biometric control program.
   - False reject rate
     A false rejection occurs when an authorized subject is rejected by the biometric system as unauthorized.
   - False accept rate
     A false acceptance occurs when an unauthorized subject is accepted as valid.
   - Crossover Error Rate
     The Crossover Error Rate (CER) describes the point where the False Reject Rate (FRR) and False Accept Rate (FAR) are equal.
c. Types of biometric controls
   - Fingerprints
   - Retina scan
     A retina scan is a laser scan of the capillaries that feed the retina of the back of the eye.
   - Iris scan
     An iris scan is a passive biometric control. A camera takes a picture of the iris (the colored portion of the eye) and then compares photos within the authentication database.
   - measurements are taken from specific points on the subject's hand:
   - Keyboard dynamics
     Keyboard dynamics refers to how hard a person presses each key and the rhythm
   - Dynamic signature
     Dynamic signatures measure the process by which someone signs his or her name
   - Voiceprint
     A voiceprint measures the subject's tone of voice while stating a specific sentence or phrase
   - Facial scan
     Facial scan technology has greatly improved over the past few years.

4. Someplace you are
   Someplace you are describes location-based access control using technologies such as the global positioning system (GPS), IP address-based geolocation, or the physical location for a point-of-sale purchase.

ACCESS CONTROL TECHNOLOGIES

There are several technologies used for the implementation of access controls.

a. Single sign-on
   Single Sign-On (SSO) allows multiple systems to use a central authentication server (AS).
b. Federated identity management
   Federated Identity Management (FIdM) applies Single Sign-On at a much wider scale: ranging from cross organization to Internet scale.
c. Kerberos
   Kerberos is a third-party authentication service that may be used to support Single Sign-On. Kerberos uses symmetric encryption and provides mutual authentication of both clients and servers.

d. SESAME
   SESAME is Secure European System for Applications in a multivendor environment, a single sign-on system that supports heterogeneous environments.

ASSESSING ACCESS CONTROL

A number of processes exist to assess the effectiveness of access control.

a. Penetration testing
   A penetration tester is a white hat hacker who receives authorization to attempt to break into an organization's physical or electronic perimeter (and sometimes both).
b. Assessing Access Control 17
   black hat hackers could do the same. They are a narrow, but often useful, test, especially if the penetration tester is successful.
   Penetration tests may include the following tests:
   • Network (Internet)
   • Network (internal or DMZ)
   • War dialing
   • Wireless
   • Physical (attempt to gain entrance into a facility or room)
   • Wireless
c. Vulnerability testing
   Vulnerability scanning (also called vulnerability testing) scans a network or system for a list of predefined vulnerabilities such as system misconfiguration, outdated
   software, or a lack of patching.
d. Security audits
   A security audit is a test against a published standard.
e. Security assessments
   Security assessments are a holistic approach to assessing the effectiveness of access control.

TUGAS

SOURCE CODE ENKRIPSI

Nama :Rahmat Hikmatullah

3. Dekripsi Algoritma Blowfish

3.1. Struktur Algoritma Blowfish

Blowfish adalah algoritma kunci simetri, yang berarti menggunakan kunci yang sama untuk melakukan enkripsi dan dekripsi file. Blowfish juga merupakan *cipher* blok, yang berarti selama proses enkripsi dan dekripsi, Blowfish akan membagi pesan menjadi blok-blok dengan ukuran yang sama panjang. Panjang blok untuk algoritma Blowfish adalah 64-bit. Pesan yang bukan merupakan kelipatan delapan *byte* akan ditambahkan bit-bit tambahan (*padding*) sehingga ukuran untuk tiap blok sama.

Algoritma dalam Blowfish terbagi menjadi dua bagian, yaitu *key expansion* dan *data encryption*.

Proses *key expansion* akan melakukan konversi sebuah kunci mulai dari 56 byte sampai beberapa *array sub* kunci dengan total mencapai 4168 byte.
Proses *data encryption* terjadi pada jaringan feistel, mengandung fungsi pengulangan sederhana sebanyak enam belas kali. Setiap iterasi, terdiri dari sebuah permutasi yang tidak bergantung pada kunci dan sebuah substitusi yang tidak bergantung pada data dan kunci. Semua operasi merupakan penambahan dan

XOR pada word 32-bit. Operasi penambahan yang dilakukan hanya merupakan empat indeks array data lookup pada setiap iterasi.

Pada algoritma Blowfish, digunakan banyak *subkey*. Kunci-kunci ini harus dihitung atau dibangkitkan terlebih dahulu sebelum dilakukan enkripsi atau dekripsi data.

Kunci- kunci yang digunakan antara lain terdiri dari, 18 buah 32-bit *subkey* yang tergabung dalam P -array (P1, P2, ..., P18). Selain itu, ada pula empat 32-bit S-box yang masing-masingnya memiliki 256 entri : S1,0, S1,1,..., S1,255; S2,0, S2,1,..,, S2,255; S3,0, S3,1,..., S3,255; S4,0, S4,1,..,, S4,255.

Pada jaringan feistel, Blowfish memiliki 16 iterasi, masukannya adalah 64-bit elemen data, X. Untuk melakukan proses enkripsi:

1. Bagi X menjadi dua bagian yang masing-masing terdiri dari 32-bit: X_L, X_R.
2. For i = 1 to 16:

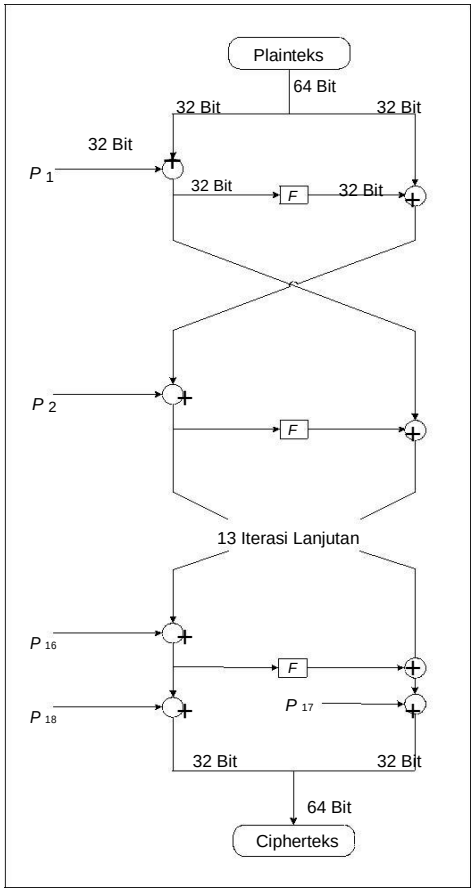
   $X_L = X_L$ XOR Pi
   $X_R = F(X_L)$ XOR X_R
   Tukar X_L dan X_R

3. Setelah iterasi ke-enam belas, tukar X_L dan X_R lagi untuk melakukan *undo* pertukaran terakhir.
4. Lalu lakukan

   $X_R = X_R$ XOR P17
   $X_L = X_L$ XOR P1

5. Terakhir, gabungkan kembali X_L dan X_R untuk mendapatkan cipherteks.

Untuk lebih jelasnya, gambaran tahapan pada jaringan feistel yang digunakan Blowfish adalah seperti pada Gambar 4.

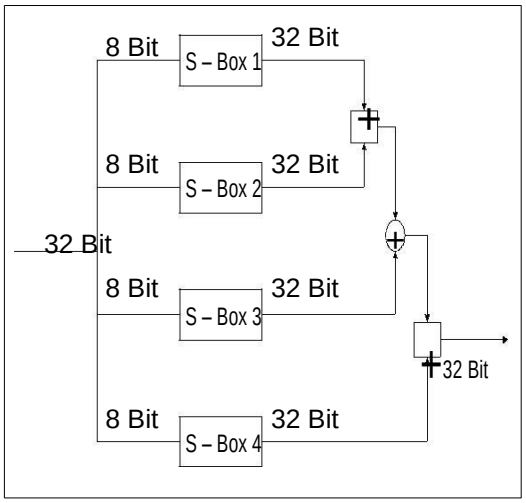


Gambar 4. Jaringan Feistel

Pada langkah kedua, telah dituliskan mengenai penggunaan fungsi F. Fungsi F adalah:

Bagi X_L menjadi empat bagian 8-bit: a,b,c dan d. $F(X_L) = ((S1,a + S2,b \bmod 2^{32}) \text{ XOR } S3,c) + S4,d \bmod 232$.

Agar dapat lebih memahami fungsi F, tahapannya dapat dilihat pada Gambar 5.



Gambar 5. Fungsi F

Algoritma Blowfish memiliki keunikan dalam hal proses dekripsi, yaitu proses dekripsi dilakukan dengan urutan yang sama persis dengan proses enkripsi, hanya saja pada proses dekripsi P1, P2, …, P18 digunakan dalam urutan yang terbalik.

Sebelumnya, telah dijelaskan mengenai penggunaan subkey didalam Blowfish. Sekarang, akan dijelaskan mengenai cara menghitung atau membangkitkan subkey:

1. Inisialisasi P-array yang pertama dan juga empat S-box, berurutan, dengan string yang telah pasti. String tersebut terdiri dari digit-digit heksadesimal dari pi, tidak termasuk angka tiga diawal.

   Contoh :

   P1 = 0x243f6a88
   P2 = 0x85a308d3
   P3 = 0x13198a2e
   P4 = 0x03707344

2. XOR pi dengan 32-bit pertama dari kunci, XOR p2 dengan 32-bit kedua dari kunci , dan seterusnya untuk seluruh bit dari kunci (sampai p18). Ulangi siklus seluruh bit kunci secara berurutan sampai seluruh P-array telah di-XOR-kan dengan bit-bit kunci.
3. Enkripsikan string yang seluruhnya nol (*all-zero*) dengan algoritma Blowfish, menggunakan *subkey* yang telah dideskripsikan di langkah (1) dan (2).
4. Gantikan p1 dan p2 dengan hasil dari langkah (3).
5. Enkripsikan hasil dari tahap (3) menggunakan algoritma Blowfish dengan *subkey* yang telah dimodifikasi.
6. Gantikan p3 dan p4 dengan hasil dari langkah (5)
7. Lanjutkan tahapan-tahapan diatas, gantikan seluruh elemen dari P- array dan kemudian keempat S-box secara berurutan, dengan hsil keluaran algoritma Blowfish yang terus menerus berubah.

Secara keseluruhan, 521 iterasi dibutuhkan untuk membangkitkan seluruh *subkey*. Aplikasi dapat menyimpan seluruh *subkey*, agar tidak perlu mengeksekusi proses ini secara berulang kali setiap iterasi.

3.2. Informasi Lain yang Terkait

Karena algoritma ini merupakan algoritma yang sudah termasuk lama, tentu saja ada beberapa usaha kriptanalisis yang dilakukan terhadap algoritma ini, antara lain adalah:

1. John Kelsey membuat sebuah *attack* yang dapat mematahkan 3 iterasi Blowfish, namun tidak dapat mengembangkannya lebih lanjut. *Attack* ini melakukan eksploitasi pada fungsi f dan fakta bahwa penambahan mod232 dan XOR tidak *commute*.
2. Vikramjit Singh Chhabra mencari cara untuk menerapkan *brute-force key search machine*
3. Serge Vaudenay melakukan penelitian pada varian Blowfish yang telah disederhanakan, dengan S-aBox yang diketahui, dan tidak *key-dependent*. Untuk varian ini, *attack* yang berbeda dapat menemukan P-array dengan 28r+1 plainteks yang telah dipilih (r merupakan jumlah iterasi). *Attack* ini tidak mungkin dilakukan pada Blowfish dengan 8-iterasi dan lebih, karena lebih banyak plainteks yang dibutuhkan daripada yang dapat dibangkitkan dengan 64-bit *cipher* blok.
4. Tesis Ph.D milik Vincent Rijmen mencantumkan second-order differential attack pada 4 iterasi Blowfish. Namun, attack tersebut tidak dapat dilanjutkan lagi untuk iterasi selanjutnya.

Untuk beberapa *weak key* (*weak key* merupakan kunci yang tidak baik yang jika digunakan dalam proses enkripsi akan mengakibatkan hasil enkripsi dengan tingkat keamanan yang buruk), yang membangkitkan S-Box yang lemah (kemungkinan untuk mendapatkannya adalah 1 banding 2^{14}) *attack* yang sama hanya membutuhkan 24r+1 plainteks yang telah diketahui untuk menemukan P-array (dengan asumsi S-box telah diketahui).

Dengan S-Box yang tidak diketahui, *attack* ini dapat mendeteksi apakah *weak key* sedang digunakan, tetapi tidak dapat mengetahui apa *weak key* tersebut (S-Box, P-array, dan kunci semuanya tidak diketahui). *Attack* ini hanya dapat berfungsi pada varian dengan jumlah iterasi yang dikurangi. Dengan kata lain, *attack* ini sama sekali tidak efektif untuk digunakan terhadap Blowfish dengan 16 iterasi.

Meskipun demikian, penemuan akan adanya *weak key* didalam Blowfish signifikan. *Weak key* pada Blowfish akan memberikan kedua masukan untuk S-Box sama persis. Sama sekali tidak ada cara untuk menangani pengecekan terhadap penggunaan *weak key* sebelum tahapan *key expansion*.

4. Penerapan Algoritma Blowfish

Algoritma Blowfish merupakan algoritma yang cukup sederhana. Namun, untuk dapat lebih memahami konsep algoritma Blowfish, penulis akan melakukan penerapan algoritma Blowfish dalam bentuk aplikasi yang akan melakukan enkripsi dan dekripsi file secara otomatis

4.1. Strategi Penerapan Perangkat Lunak

Perangkat lunak ini diharapkan dapat melakukan proses enkripsi secara optimal, maka penerapan yang dilakukan harus mempertimbangkan beberapa aspek yang dianggap mempengaruhi kecepatan dan keamanannya.

4.1.1. Strategi Perancangan Perangkat Lunak

Hal-hal yang harus dipertimbangkan dalam perancangan perangkat lunak yang menerapkan algoritma Blowfish antara lain adalah prinsip perancangan algoritma Blowfish (sifat dasar algoritma Blowfish), yaitu:

1. lebih optimal jika digunakan untuk aplikasi yang tidak sering berganti kunci.
2. lebih cepat jika diterapkan pada 32-bit mikroprosessor dengan *data cache* yang besar. Namun, karena hal ini merupakan batasan teknis sehingga tidak akan dipakai dalam strategi perancangan.
3. Lebih aman jika diterapkan tanpa adanya pengurangan jumlah iterasi (dengan asumsi *user* tidak menggunakan *weak key*).

Setelah mencari beberapa referensi, ternyata Blowfish tidak memiliki karakteristik lainnya yang harus dipenuhi. Ternyata selama aplikasi yang digunakan tidak sering berganti kunci, Blowfish merupakan algoritma yang cepat, dan selama tidak dilakukan pengurangan jumlah iterasi (dengan asumsi tidak ada penggunaan *weak key* oleh *user*) Blowfish merupakan algoritma enkripsi yang aman.

Berikut perbandingan kecepatan algoritma Blowfish dengan beberapa algoritma *cipher* blok lainnya:

Algoritma	Clock cycles tiap iterasi	Jumlah iterasi	Jumlah clock cycles per byte terenkripsi
Blowfish	9	16	18
Khufu/ Khafre	5	32	20
RC5	12	16	23
DES	18	16	45
IDEA	50	8	50
Triple-DES	18	48	108

Tabel 1. Perbandingan kecepatan

Tingkat keamanan Blowfish terbukti dengan belum adanya *attack* yang mampu mematahkan algoritma enkripsi ini.

4.2. Deskripsi Singkat Perangkat Lunak

Berdasarkan pertimbangan diatas, maka penulis memutuskan perangkat lunak yang akan dikembangkan adalah sebuah enkripsi file teks otomatis, agar kunci yang digunakan untuk melakukan suatu enkripsi pada suatu saat tidak berubah -ubah. Pada aplikasi ini, untuk melakukan proses enkripsi *user* diharuskan memasukkan kata kunci yang diinginkan, yang kemudian akan digunakan untuk melakukan proses enkripsi. Untuk enkripsi satu file atau satu teks, hanya

dibutuhkan satu kunci. Lebih lengkapnya, dapat dilihat pada deskripsi perangkat lunak berikut ini:

Perangkat lunak yang akan dibuat adalah perangkat lunak untuk melakukan enkripsi file teks (hanya dengan ekstensi txt, bukan doc) dengan pemanfaatan algoritma Blowfish.

Perangkat lunak ini diharapkan dapat menerapkan algoritma Blowfish secara optimal sehingga baik dalam kecepatan maupun keamanan dapat dianggap baik.

Perangkat lunak ini akan dapat membuka file teks yang ingin dienkripsi dan menampilkan data dalam file tersebut dalam bentuk teks biasa, namun perlu diperhatikan sekali lagi hanya file dengan ekstensi txt yang dapat dibuka. Selain itu, aplikasi ini juga dapat melakukan enkripsi pada kata-kata yang ditulis secara langsung pada 'textbox' yang ada pada aplikasi.

Aplikasi ini akan melakukan proses enkripsi dan dekripsi secara otomatis. *User* hanya perlu memasukkan kata kunci *(password*) yang diinginkan, kemudian tinggal menekan tombol 'encrypt' dan 'decrypt' yang terdapat pada aplikasi ini. Sedangkan untuk proses dekripsi, *user* akan diminta memasukkan kembali kata kunci yang diketikkannya ketika file tersebut dienkripsi.

Setelah proses enkripsi dilakukan *user* akan diminta untuk menyimpan hasil enkripsi tersebut agar dapat didekripsi jika diperlukan.

Aplikasi ini penulis beri nama 'Ikan Kembung' yang penulis ambil dari terjemahan langsung nama algoritma Blowfish ke dalam bahasa Indonesia.

Aplikasi ini akan melakukan proses enkripsi dengan mode operasi ECB (*Electronic Code Book*)

4.2.1. Proses Penerapan Perangkat Lunak

Proses perancangan perangkat lunak ikan kembung tidak mengikuti suatu metode tertentu, melainkan langkah-langkah yang penulis terapkan sendiri.

Tahapan penerapan perangkat lunak Ikan Kembung:

1. Penentuan spesifikasi perangkat lunak, yaitu:
   a. Dapat menerima tulisan *user* secara langsung pada 'textbox' dan langsung melakukan enkripsi
   b. Mode operasi enkripsi menggunakan ECB (*electronic code book*)
   c. Semua hasil enkripsi akan ditampilkan terlebih dahulu pada 'textbox' yang ada pada aplikasi
   d. Setelah proses enkripsi dilakukan, aplikasi akan secara otomatis menghapus plainteks dan *key* yang dimasukkan *user*
2. Perancangan antarmuka yang dapat memudahkan *user* dan memenuhi seluruh fungsi yang dibutuhkan oleh aplikasi ini (dapat dilihat pada bagian antarmuka aplikasi).
3. Proses konstruksi perangkat lunak (*coding)*. Meskipun dikatakan *coding*, namun sebenarnya penulis hanya menerjemahkan kode yang diberikan oleh Bruce Schneier dalam bukunya, *Applied Cryptography*. Pada bagian belakang buku teresbut, terdapat *source code* Blowfish didalam bahasa C.

   Pada kode yang dibuat oleh Bruce Schneier, merupakan kode standard yang tidak menggunakan tampilan aplikasi, melainkan hanya menggunakan main program yang akan dijalankan ketika keseluruhan program tersebut dieksekusi.

   Untuk menerapkan algoritma ini kedalam bentuk aplikasi yang memiliki antarmuka, dan bahasa basic diperlukan penanganan lebih lanjut.

   Contoh kode Ikan kembung, pada bagian algoritma Blowfish saja:

```
Private Function blf_F(x As Long) As Long
   Dim a As Byte, b As Byte, C As Byte, d As
   Byte Dim y As Long

   Call uwSplit(x, a, b, C, d)
   y = uw_WordAdd(blf _S(0, a), blf_S(1,
   b)) y = y Xor blf_S(2, C)
   y = uw_WordAdd(y, blf_S(3,
   d)) blf_F = y
End Function
```

   Kode diatas merupakan fungsi F, yang digunakan pada jaringan feistel.

Kode yang digunakan oleh Bruce Schneier:

```
unsigned long F(blf_ctx
*bc, unsigned long x)
{
   unsigned long a;
   unsigned long b;
   unsigned long c;
   unsigned long d;
   unsigned long y;

   d = x & 0x00FF;
   x >>= 8;
   c = x & 0x00FF;
   x >>= 8;
   b = x & 0x00FF;
   x >>= 8;
   a = x & 0x00FF;
   y = bc->S[0][a] +
bc->S[1][b];
   y = y ^ bc->S[2][c];
   y = y + bc->S[3][d];
   return y;
}
```

Dapat dilihat bahwa kodenya menjadi berbeda.

Berikut adalah kode untuk proses enciphering dan dechipering

```
Public Function blf_EncipherBlock(xL As
Long, xR As Long) 'fungsi enchiper
   Dim i As Integer
   Dim temp As Long

   For i = 0 To ncIterasi - 1 'jaringan feistel
      xL = xL Xor blf_P(i)
      xR = blf_F(xL) Xor
      xR temp = xL
      xL = xR
      xR = temp
   Next

   temp = xL
   xL = xR
   xR = temp

   xR = xR Xor blf_P(ncIterasi) xL
   = xL Xor blf_P(ncIterasi + 1)

End Function

Public Function blf_DecipherBlock(xL As
Long, xR As Long) 'fungsi dechiper

Dim i As Integer
   Dim temp As Long

   For i = ncIterasi + 1 To 2 Step -1
      xL = xL Xor blf_P(i)
      xR = blf_F(xL) Xor
      xR temp = xL
      xL = xR
      xR = temp
   Next

   temp = xL
   xL = xR
   xR = temp

   xR = xR Xor blf_P(1)
   xL = xL Xor blf_P(0)

End Function
```

Pada kode diatas, ncIterasi merupakan konstanta jumlah iterasi yang digunakan pada aplikasi Ikan Kembung, yaitu sebanyak 16 kali mengikuti algoritma Blowfish yang dikembangkan oleh Bruce schneier.

Berikut kode yang digunakan oleh Bruce Schneier.

```
void Blowfish_encipher(blf_ctx
*bc, unsigned long *xl,
unsigned long *xr)
{
   unsigned long Xl;
   unsigned long Xr;
   unsigned long temp;
   short i;

   Xl = *xl;
   Xr = *xr;

   for (i = 0; i < N; ++i)
   {
      Xl = Xl ^ bc->P[i];
      Xr = F(bc, Xl) ^ Xr;
      temp = Xl;
      Xl = Xr;
      Xr = temp;
   }

   temp = Xl;
   Xl = Xr;
   Xr = temp;
   Xr = Xr ^ bc->P[N];
   Xl = Xl ^ bc->P[N + 1];

   *xl = Xl;
   *xr = Xr;
}

void Blowfish_decipher(blf_ctx
*bc, unsigned long *xl,
unsigned long *xr)
{
   unsigned long  Xl;
```

```
    unsigned long  Xr;
    unsigned long  temp;
    short          i;

    Xl = *xl;
    Xr = *xr;

    for (i = N + 1; i > 1; --i)
    {
       Xl = Xl ^ bc->P[i];
       Xr = F(bc, Xl) ^ Xr;
       /* Exchange Xl and Xr
       */ temp = Xl;
       Xl = Xr;
       Xr = temp;
    }

    /* Exchange Xl and Xr
    */ temp = Xl;
    Xl = Xr;
    Xr = temp;
    Xr = Xr ^ bc->P[1];
    Xl = Xl ^ bc->P[0];

    *xl = Xl;
    *xr = Xr;
}
```

Dapat dilihat bahwa kedua kode memiliki kemiripan dalam hal algoritma namun berbeda cara penerapannya.

Lebih lengkapnya kode aplikasi Ikan Kembung dapat dilihat pada *source code* yang saya sertakan pada CD makalah ini.

5. Penjelasan singkat aplikasi

Aplikasi Ikan Kembung merupakan aplikasi berbasis desktop yang dikembangkan pada kakas pengembangan Visual Basic 6.0 dengan bahasa pengembangan Basic. Aplikasi ini berbasis GUI, dengan kata lain sudah memiliki *user interface* dalam bentuk grafis. Hal ini bertujuan untuk memudahkan *user*. Tampilan awal aplikasi ini dapat dilihat pada halaman Antarmuka Aplikasi pada Gambar 6.

Untuk melakukan enkripsi dengan aplikasi Ikan Kembung, ada beberapa tahapan yang harus dilakukan.

Tahapan proses enkripsi pada Ikan Kembung, yaitu:

1. Mengetikkan kata yang ingin dienkripsi pada ’textbox’ plainteks yang terdapat pada aplikasi.
2. Mengetikkan key pada ’textbox’ key aplikasi. Jika *user* tidak memasukkan key, maka aplikasi akan mengeluarkan peringatan dalam bentuk ’message box’.
3. Menekan tombol ‘encrypt’

Contoh langkah proses enkripsi dapat dilihat pada halaman Antarmuka Aplikasi pada Gambar 7.

Setelah langkah-langkah tersebut dilakukan, aplikasi akan memulai proses enkripsi. Kemudian akan menampilkan cipherteks di dalam ’textbox’ untuk cipherteks. Contoh hasil enkripsi dapat dilihat pada halaman Antarmuka Aplikasi pada Gambar 8.

Pada Gambar 8 dapat dilihat bahwa setelah proses enkripsi dilakukan, plainteks dan kunci dihilangkan.

Setelah melakukan enkripsi, *user* dapat langsung melakukan proses dekripsi dengan cara:

1. Mengetikkan key pada ’textbox’ key yang terdapat pada aplikasi.
2. menekan tombol ’decrypt’

Contoh langkah proses dekripsi dapat dilihat pada halaman Antarmuka Aplikasi pada Gambar 9.

Jika *user* tidak memasukkan key pada ’textbox’ key, maka apliksi akan mengeluarkan pesan kesalahan seperti dapat dilihat pada Gambar 10.

Setelah itu, aplikasi akan melakukan proses dekripsi pada cipherteks yang terdapat pada ’textbox’ cipherteks. Contoh hasil dekripsi dapat dilihat pada halaman Antarmuka Aplikasi pada Gambar 11.

Keterangan mengenai aplikasi dapat dilihat pada form ’About’ dan ’Help’

6. Pengujian Perangkat Lunak

Perangkat lunak Ikan Kembung akan diuji untuk mengetahui kecepatannya jika dibandingkan dengan algoritma lain. Aplikasi pembandingnya menggunakan algoritma buatan mahasiswa teknik informatika yang digunakan sebagai contoh pada tugas kriptografi.

Aplikasi tersebut bernama Diencrypt, yang dibuat oleh :

1. 13501006 Rizki Yulianto
2. 13501027 Widhiyo Sudiyono

3. 13501072 Anugrah Redja kusuma

Sebenarnya penulis kurang mengetahui mengenai algoritma yang diterapkan pada aplikasi Diencrypt, namun penulis yakin algoritma yang digunakan berbeda dengan algoritma Blowfish. Selain itu alasan memilih aplikasi Diencrypt adalah, karena sudah ada data perbandingan dengan algoritma cipher blok lainnya (lihat Tabel 1) maka menurut penulis lebih menarik jika membandingkan dengan algoritma yang belum ada datanya.

Tetapi perlu diperhatikan perbandingan yang penulis lakukan berdasarkan pengamatan penulis semata. Tidak ada dasar ilmiah maupun alat untuk mengukur kecepatan kedua aplikasi secara tepat.

6.1. Perancangan Kasus Uji Pengujian Perangkat Lunak

Berdasarkan teknik pengujian yang diterapkan penulis, maka dirancang kasus-kasus uji sebagai berikut:

1. Kasus Uji 1
   Kasus Uji 1 bertujuan untuk mengetahui seberapa cepat aplikasi Ikan Kembung dan Diencrypt melakukan proses enkripsi terhadap teks yang telah disediakan

Teks yang akan digunakan untuk menguji kedua aplikasi :

Vigènere Cipher

- Termasuk ke dalam *cipher* abjad-majemuk (*polyalpabetic substitution cipher*).
- Dipublikasikan oleh diplomat (sekaligus seorang kriptologis) Perancis, Blaise de Vigènere pada abad 16 (tahun 1586).
- Tetapi sebenarnya Giovan Batista Belaso telah menggambarkannya pertama kali pada tahun 1553 seperti ditulis di dalam bukunya *La Cifra del Sig. Giovan Batista Belaso*
- Algoritma tersebut baru dikenal luas 200 tahun kemudian yang oleh penemunya *cipher* tersebut kemudian dinamakan *Vigènere Cipher*
- *Cipher* ini berhasil dipecahkan oleh Babbage dan Kasiski pada pertengahan Abad 19.
- *Vigènere Cipher* digunakan oleh Tentara Konfiderasi (*Confederate Army*) pada Perang Sipil Amerika (*American Civil war*).
- Perang Sipil terjadi setelah *Vigènere Cipher* berhasil dipecahkan.

Aplikasi Ikan Kembung memiliki banyak perbedaan dengan aplikasi Diencrypt, antara lain:

1. Aplikasi Diencypt hanya dapat melakukan enkripsi pada file, sedangkan aplikasi Blowfish melakukan enkripsi pada teks yang ditulis langsung pada aplikasi.
2. Aplikasi Diencrypt akan meminta *user* menyimpan file hasil dekripsi terlebih dahulu sebelum menampilkan hasil dekripsi
3. Proses dekripsi harus dilakukan dari file.

6.2. Evaluasi Hasil Pengujian Perangkat Lunak

Berdasarkan pengujian yang telah dilakukan, penulis memutuskan bahwa kedua aplikasi memiliki kecepatan yang hampir sama.

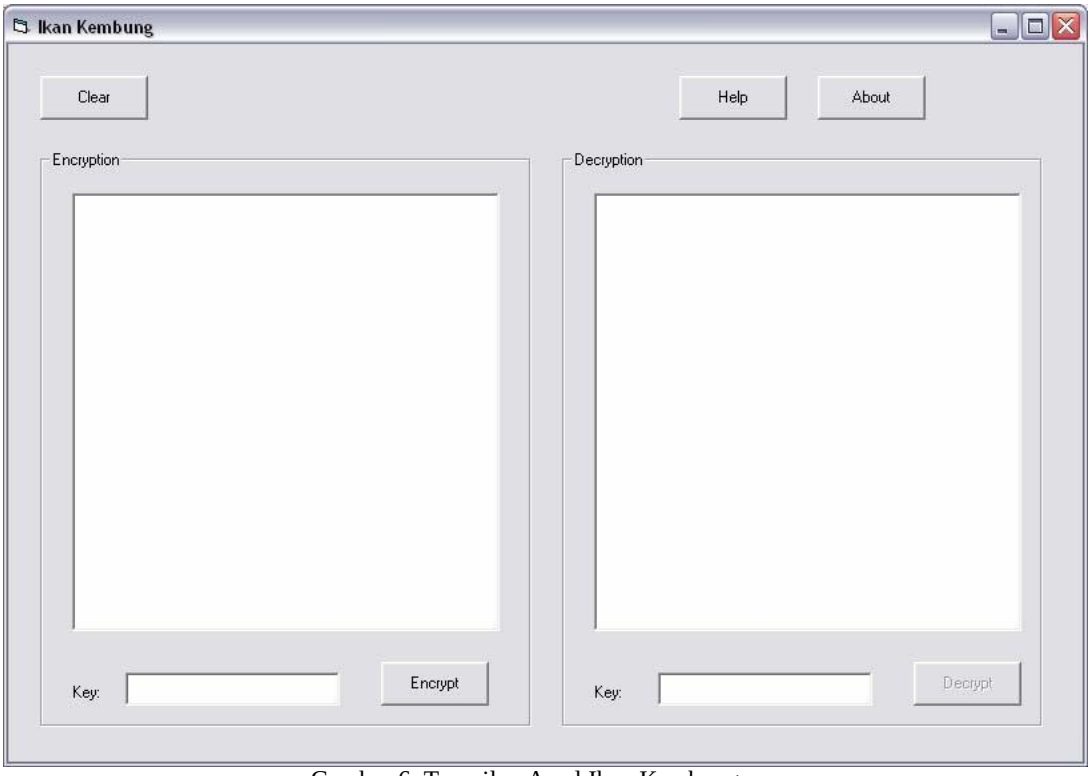
Namun hasil tersebut tetap kurang dapat dipercaya, karena hanya merupakan pengamatan penulis, ada juga pengaruh dari berbedanya kakas pengembangan yang digunakan.

7. Kesimpulan

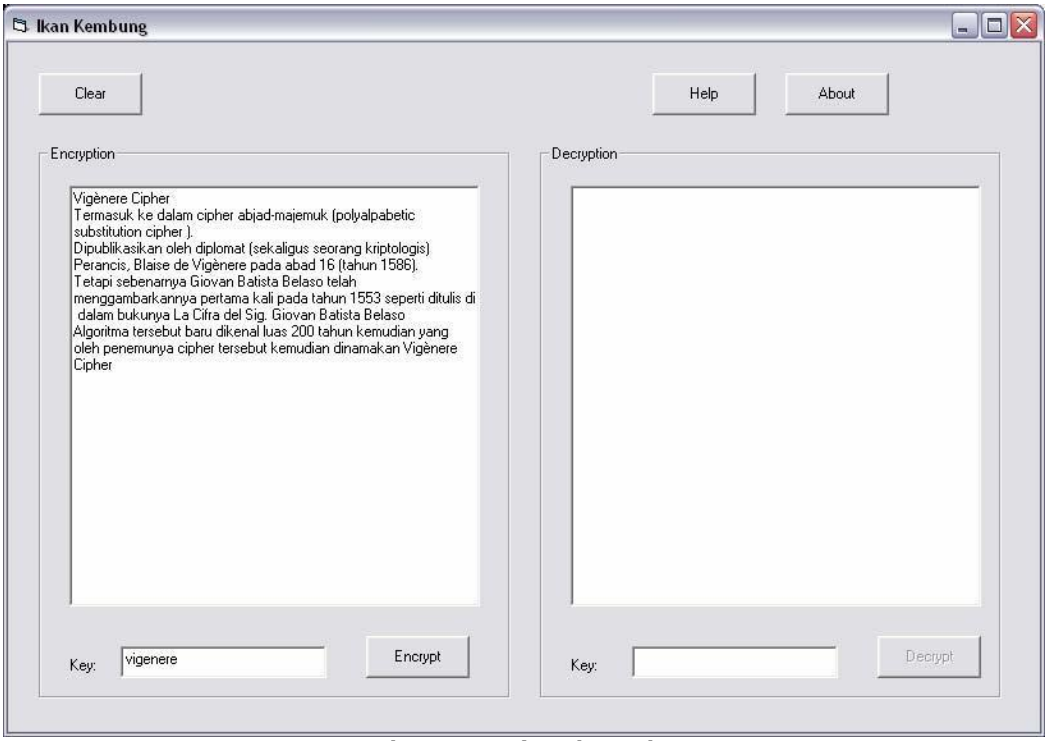
Kesimpulan yang dapat diambil dari studi dan implementasi algoritma Blowfish untuk aplikasi enkripsi dan dekripsi file ini adalah:

1. *Blowfish* merupakan salah satu solusi yang baik untuk mengatasi masalah keamanan dan kerahasiaan data yang pada umumnya diterapkan dalam saluran komunikasi dan file.
2. Algoritma Blowfish merupakan algoritma yang sederhana dan menggunakan jaringan feistel
3. Algoritma Blowfish dapat diimplementasikan dengan bahasa apapun (selain C) namun harus disesuaikan cara penulisan maupun tipe-tipe datanya.
4. Implementasi algoritma Blowfish yang optimal dapat dilakukan dengan aplikasi yang tidak sering berubah-ubah kunci.
5. Tingkat keamanan algoritma blowfish ditentukan oleh jumlah iterasi dan panjang kunci yang digunakan
6. Blowfish memiliki *Weak Key* meskipun amat jarang kemungkinan terjadinya.

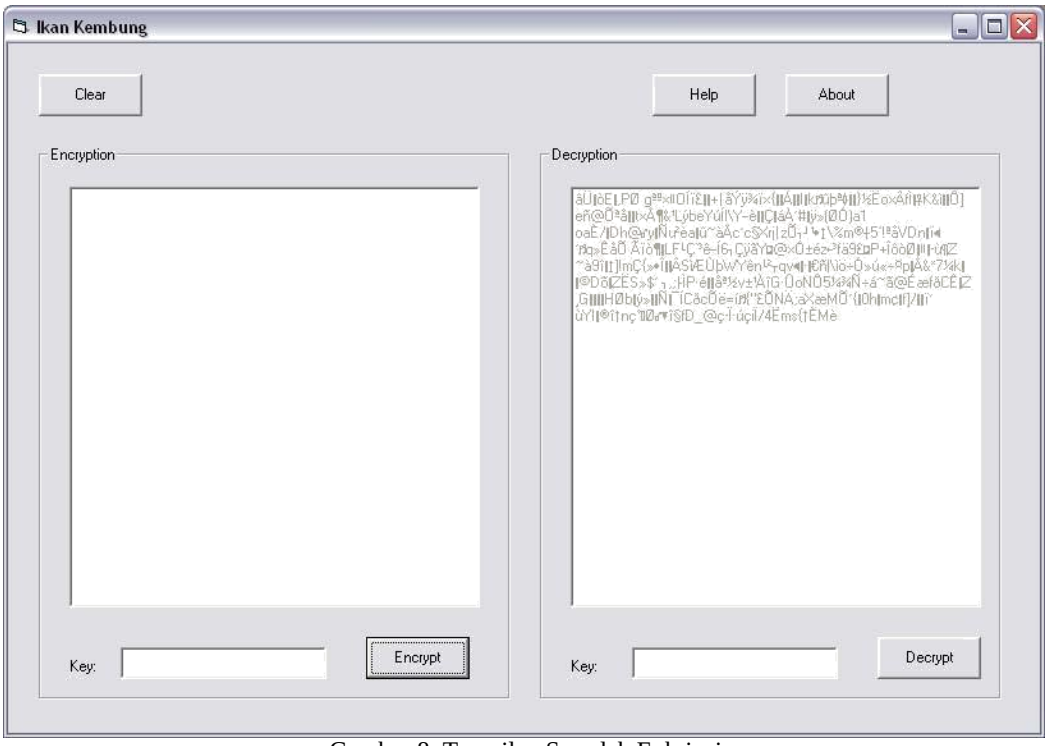
Antarmuka Aplikasi



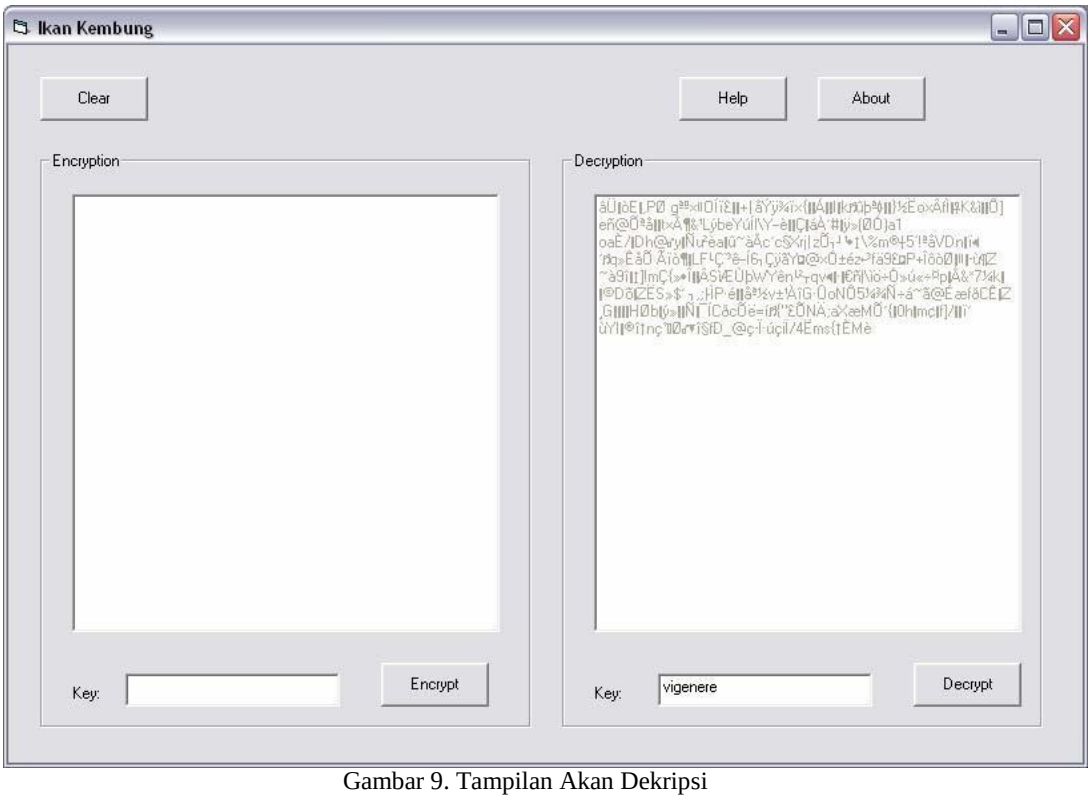
Gambar 6. Tampilan Awal Ikan Kembung



Gambar 7. Tampilan Akan Enkripsi

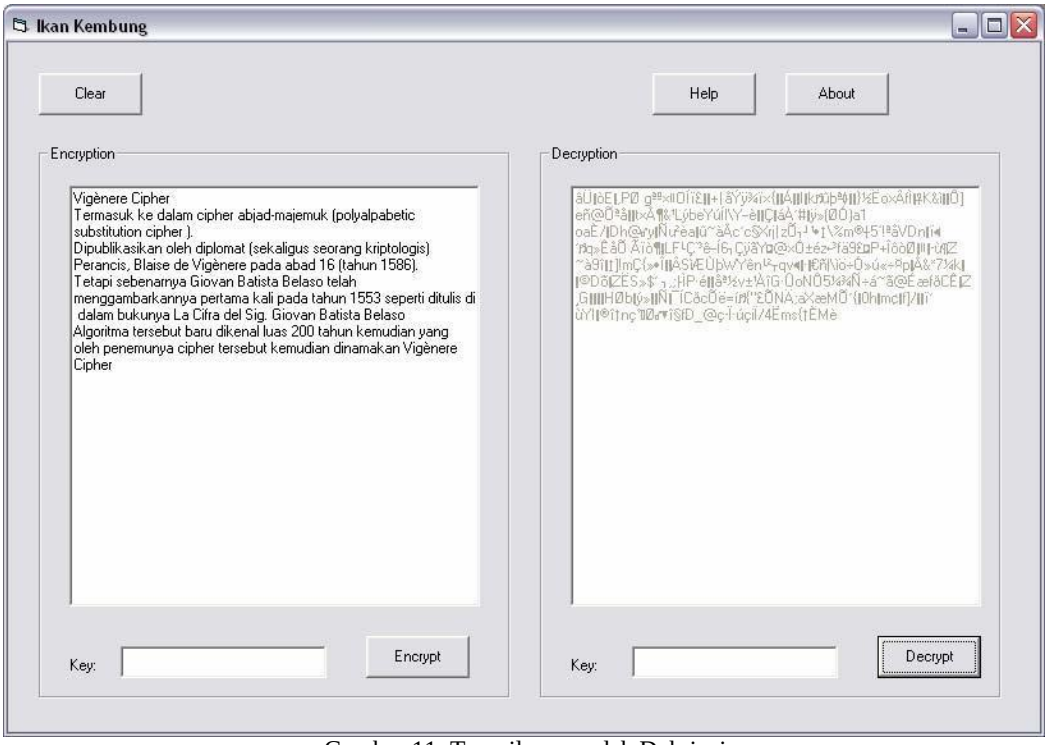


Gambar 8. Tampilan Sesudah Enkripsi

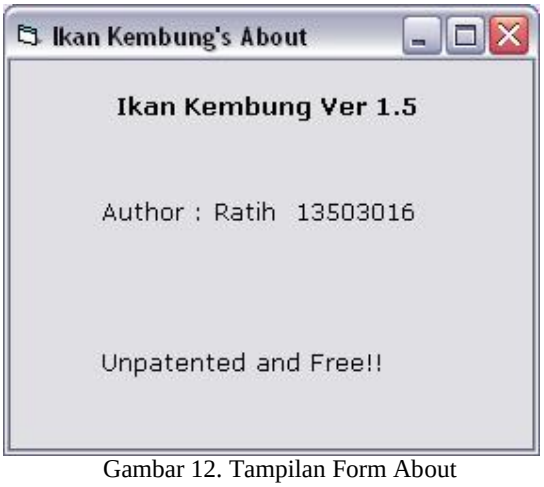


Gambar 9. Tampilan Akan Dekripsi

Gambar 10. Peringatan Memasukkan *Key*



Gambar 11. Tampilan sesudah Dekripsi



Gambar 12. Tampilan Form About

DAFTAR PUSTAKA

[1] Munir, Rinaldi. (2004). Bahan Kuliah IF5054 Kriptografi. Departemen Teknik Informatika, Institut Teknologi Bandung.

[2] Schneier, Bruce. (1996). Applied Cryptography 2nd. John Wiley & Sons.

[3] Wikipedia (2006), *http://en .wikipedia.org/wiki/Cryptograpy*. Tanggal akses : 5 Oktober 2006 pukul 12.00

[4] Modern Private Key Ciphers (1996), *http://williamstallings.com/Extras/Security-Notes/lectures/index.html*. Tanggal akses: 5 Oktober 2006 pukul 12.00

[5] Wikipedia (2006), *http://en.wikipedia.org/wiki/Blowfish_(cipher)*. Tanggal akses : 5 Oktober 2006 pukul 12.00

[6] Bruce Schneier (1994), *http://www.schneier.com/paper-blowfish-fse.html*. Tanggal akses : 5 Oktober 2006 pukul 12.00