P-ISSN: 2548-9704
E-ISSN: 2686-0880

PENERAPAN ALGORITMA RIVEST SHAMIR ADLEMAN (RSA) UNTUK MENGAMANKAN DATABASE PROGRAM KELUARGA HARAPAN (PKH)

Andika Cahya Putra [1], Magdalena Simanjuntak[2], Nurhayati [3]

Mahasiswa Program Studi Teknik Informatika,
STMIK KAPUTAMA
Jl. Veteran No. 4A-9A, Binjai 20714, Sumatera Utara

ABSTRACT

To maintain security, encryption techniques are used so that the confidentiality of the data is guaranteed. One of the encryption algorithms that is often used is the RSA (Rivest Shamir Adleman) algorithm. On this occasion the author is interested in studying the sql server database security application. In this study, the RSA (Rivest Shamir Adleman) algorithm is used to protect the PKH (Family Hope Program) database. The system will generate public keys and private keys. To secure the PKH database is encrypted with a public key. All data will be encrypted, while the private key will decrypt or restore the original state with the RSA algorithm. The application of the RSA cryptographic algorithm is a good solution for the sql server database security system that will be used to secure the PKH database. To ensure the confidentiality of PKH data stored in the database, by using the RSA algorithm into the system, the data is stored in the database so that the contents of the data cannot be understood by other parties.

Keywords: RSA Algorithm, Security, PKH, Sql Server.

ABSTRAK

Untuk menjaga keamanan digunakan teknik enkripsi agar kerahasiaan data terebut terjamin. Salah satu algoritma enkripsi yang sering digunakan adalah algoritma RSA (*Rivest Shamir Adleman*). Pada kesempatan ini penulis tertarik mengkaji tentang aplikasi pengamanan *database sql server*. Pada penelitian ini, algoritma RSA (*Rivest Shamir Adleman*) digunakan sebagai pelindung database PKH (Program Keluarga Harapan), Sistem akan membangkitkan kunci *public* dan kunci *private*. Untuk mengamankan database PKH dienkripsi dengan kunci *public* Seluruh data akan dienkripsi, Sedangkan kunci *private* akan melakukan dekripsi atau mengembalikan dalam keadaan asli dengan algoritma RSA. Penerapan algoritma kriptografi RSA menjadi solusi yang baik pada sistem pengamanan database sql server yang akan digunakan untuk mengamankan *database* PKH. Untuk menjamin kerahasiaan data-data PKH yang disimpan didalam *database*, dengan penggunaan algoritma RSA ke dalam sistem tersebut maka data yang disimpan di dalam *database* sehingga isi datanya tidak dapat dimengerti oleh pihak lain.

Kata Kunci: Algoritma RSA, Keamanan, PKH, Sql Server.

1. PENDAHULUAN

Masalah Keamanan dan kerahasian data merupakan suatu hal yang sangat penting terutama di dalam dunia pekerjaan. Database yang bersifat rahasia perlu dibuatkan penyimpanan yang aman. Database yang telah di enkripsi tidak dapat di buka oleh pihak-pihak yang tidak mengetahui kunci yang digunakan untuk mengenkripsi dan mendekripsi database tersebut. Dengan begitu database akan terproteksi dan akan terhindar dari pihak-pihak yang tidak berkepentingan dan tidak bertanggung jawab yang ingin merusak atau melihat isi data tersebut.

Dalam pengaplikasian kriptografi ini, mengambil referensi dari jurnal berjudul "Pengamanan Database Pada Aplikasi Test Masuk Karyawan Baru Berbasis Web Menggunakan Algoritma Kriptografi AES-128 Dan RC4" hasil penelitian

P-ISSN: 2548-9704
E-ISSN: 2686-0880

ini dapat membantu dan mempermudah untuk melakukan proses enkripsi dan dekripsi yang akan mengamankan database (Grehasen & Mulyati, 2017).

Algoritma *Rivest Shamir Adleman (RSA)* merupakan salah satu metode kriptografi yang menggunakan kunci private dan kunci public untuk proses enkripsi dan dekripsi atau dalam istilahnya sering disebut teknik asimetris, algoritma ini menggunakan kunci private untuk melakukan enkripsi dan untuk dekripsinya menggunakan kunci public. Algoritma *Rivest Shamir Adleman (RSA)* digunakan karena melakukan pemfaktoran bilangan yang sangat besar, Oleh karena alasan tersebut RSA dianggap aman. Untuk membangkitkan dua kunci, dipilih dua bilangan prima acak yang besar.

2. METODOLOGI PENELITIAN

2.1 Kriptografi

Kriptografi (*cryptography*) berasal dari bahasa Yunani yang terdiri dari dua suku kata, yaitu *cryptós* yang berarti rahasia dan *gráphein* yang berarti kata tulisan. Karena itu secara umum kriptografi diartikan sebagai tulisan rahasia. Terdapat beberapa definisi kriptografi dalam berbagai literatur. Definisi pada tahun 80-an menyatakan kriptografi adalah ilmu dan seni untuk menjaga keamanan pesan. Kata seni dalam definisi ini berasal dari fakta sejarah bahwa pada awal sejarah kriptografi, setiap orang mempunyai cara yang unik untuk merahasiakan pesan.

Sedangkan definisi dalam buku-buku terbaru menyatakan kriptografi merupakan ilmu mengenai metode untuk mengirimkan pesan secara rahasia sehingga hanya penerima yang dimaksud yang dapat menghapus dan membaca pesan tersebut atau memahaminya. Pengertian lain kriptografi yaitu ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi seperti kerahasiaan, integritas data, serta otentikasi. Kata *graphy* dalam kata *cryptography* itu sendiri sudah menyiratkan sebuah seni.

Jadi, kriptografi adalah suatu ilmu sekaligus seni yang bertujuan untuk menjaga keamanan suatu pesan (*cryptography is the art and science of keeping messages secure*). Secara umum, kriptografi adalah teknik pengamanan informasi dimana informasi diubah dengan kunci tertentu melalui enkripsi sehingga menjadi informasi baru yang tidak dapat dimengerti oleh orang yang tidak berhak menerimanya, dan informasi tersebut hanya dapat di ubah kembali oleh orang yang berhak menerimanya melalui dekripsi.

2.2 Algoritma RSA(*Rivest Shamir Adleman)*

RSA merupakan algoritma kriptografi kunci publik (asimetris). Ditemukan pertama kali pada tahun 1977 oleh Ron Rivest, Adi Shamir, dan Len Adleman. Nama RSA sendiri diambil dari ketiga penemunya tersebut. Sebagai algoritma kunci publik, RSA mempunyai dua kunci, yaitu kunci publik dan kunci rahasia. RSA mendasarkan proses enkripsi dan dekripsinya pada konsep bilangan prima dan aritmetika modulo. Baik kunci enkripsi maupun dekripsi keduanya merupakan bilangan bulat. Kunci enkripsi tidak dirahasiakan dan diberikan kepada umum (sehingga disebut dengan kunci publik), namun kunci untuk dekripsi bersifat rahasia (kunci privat). Untuk menemukan kunci dekripsi, dilakukan dengan memfaktorkan suatu bilangan bulat menjadi faktor-faktor primanya. Kenyataannya, memfaktorkan bilangan bulat menjadi faktor primanya bukanlah pekerjaan yang mudah. Karena belum ditemukan algoritma yang efisien untuk melakukan pemfaktoran. Cara yang bisaa digunakan dalam pemfaktoran adalah dengan menggunakan pohon faktor. Jika semakin besar bilangan yang akan difaktorkan, maka semakinlama waktu yang dibutuhkan. Jadi semakin besar bilanganyang difaktorkan, semakin sulit pemfaktorannya, semakin kuat pula algoritma RSA (Ginting et al., 2015).

Tingkat keamanan algoritma penyandia RSA sangat tergantung pada ukuran kunci sandi (dalam bit). Semakin besar ukuran kunci, maka semakin sulit juga penyadap terjadi. Kombinasi kunci ini lebih dikenal dengan istilah *brute fore attack* yang bila panjang kuncinya 256 bit, maka menjadi tidak ekonomis dan sia-sia jika hacker pun meyadap sandi (Brink et al., 2014).

2.3 Proses Pembuatan Kunci

Algoritma pembangkit kunci ada 2 bilangan prima besar yaitu n = p x q sangat sulit

P-ISSN: 2548-9704
E-ISSN: 2686-0880

untuk difaktorisasikan. Di rekomendasikan besar p dan q adalah 512 bit sehingga n berukuran 1024 bit. Karena p dan q adalah bilangan prima, maka n =(p – 1) (q – 1). Kemudian pilih sebuah integer e dipilih secara acak dari Ƶϕ(n) yang memenuhi gcd(e,ϕ(n)) sehingga e merupakan generator pada Ƶϕ(n). Selanjutnya algoritma pembangkit kunci RSA menghitung d invers perkalian e pada Ƶϕ(n). Pada akhirnya algoritma pembangkit kunci RSA menetapkan (e,n) sebagai kunci publik dan d sebagai kunci privat atau tetap dirahasiakan.

Langkah-langkah dalam pembangkit kunci RSA:

1. Pilih dua buah bilangan prima sembarang p dan q. nilai p dan q harus dirahasiakan.
2. Hitung nilai n dari rumus, $n – p \; x \; q$. Besaran n tidak perlu dirahasiakan
3. Hitung $m = (p – 1) (q – 1)$. Bersaran m perlu dirahasiakan
4. Dipilih sebuah bilangan bulat sebgai kunci publik, disebut namanya e, yaitu relativ prima terhadap m. e relative prima terhadap m artinya factor pembagi terbesar keduanya adalah 1, secara matematis disebut gcd (e,m)=1. Untuk mencarinya dapat digunakan algoritma Euclid. Nilai e bersifat tidak rahasia.

Hitung kunci privat, disebut namanya d sedemikian agar (d x e) mod m=1. Untuk mencari d yang sesuai dapat juga digunakan algoritma *Extended Euclid* Maka hasil dari algoritma tersebut diperoleh Kunci public adalah pasangan (e,n) bersifat tidak rahasia, Kunci privat adalah pasangan (d,n) bersifat rahasia (Brink et al., 2014).

2.4 Enkripsi dan Dekripsi

Enkripsi merupakan data yang disandikan dengan kunci publiknya (n,e) agar tidak dapat diketahui data (informasi) yang akan dikirimkan (P).kemudian menghitung ciphertext (c) sesuai dengan :

$$C=p^e mod\ n$$

Deskripsi adalah mendapatkan data maupun informasi yang telah dienkripsi atau disandikan (n,e) dari pengirim (p) untuk membaca daa asli menggunakan kunci privatnya, dapat dienkripsi dengan :

$$P=c^d mod\ n$$

Enkripsi dan dekripsi harus mengetahui nilai e dan nilai n, satu diantaranya harus memiliki d untuk melakukan dekripsi terhadap hasil enkripsi dengan menggunakan *public key* e. Syarat nilai e dan d adalah gcd(d,e)=1 (Brink et al., 2014).

2.5 Kekuatan dan Keamanan RSA

Keamanan algoritma RSA terletak pada tingkat kesulitan dalam memfaktorkan bilangan non prima menjadi faktor primanya, yang dalam hal ini $r = p$ x q. Sekali r berhasil difaktorkan menjadi p dan q, maka $\phi(r) = (p – 1) (q – 1)$ dapat dihitung. Selanjutnya, karena kunci enkrispi *PK* diumumkan (tidak rahasia), maka kunci dekripsi *SK* dapat dihitung dari persamaan $PK.SK = 1 \pmod{\phi(r)}$.

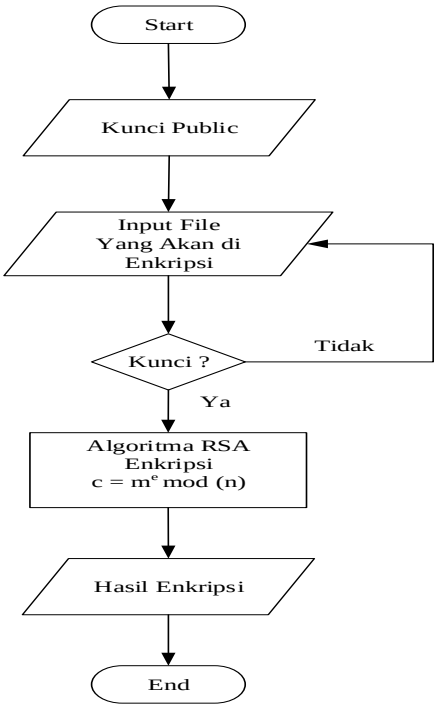
Penemu algoritma RSA menyarankan nilai p dan q panjangnya lebih dari 100 digit. Dengan demikian hasil kali $r = p$ x q akan berukuran lebih dari 200 digit. Menurut Rivest dan kawan-kawan, uasaha untuk mencari faktor bilangan 200 digit membutuhkan waktu komputasi selama 4 milyar tahun (dengan asumsi bahwa algoritma pemfaktoran yang digunakan adalah algoritma yang tercepat saat ini dan komputer yang dipakai mempunyai kecepatan 1 milidetik).

Untuk memfaktorkan bilangan yang besar belum ditemukan, Inilah yang membuat algoritma RSA tetap dipakai hingga saat ini. Selagi belum ditemukan algoritma yang mangkus untuk memfaktorkan bilangan bulat menjadi faktor primanya, maka algoritma RSA tetap direkomendasikan untuk menyandikan pesan (Ginting et al., 2015).

3.3 HASIL DAN PEMBAHASAN

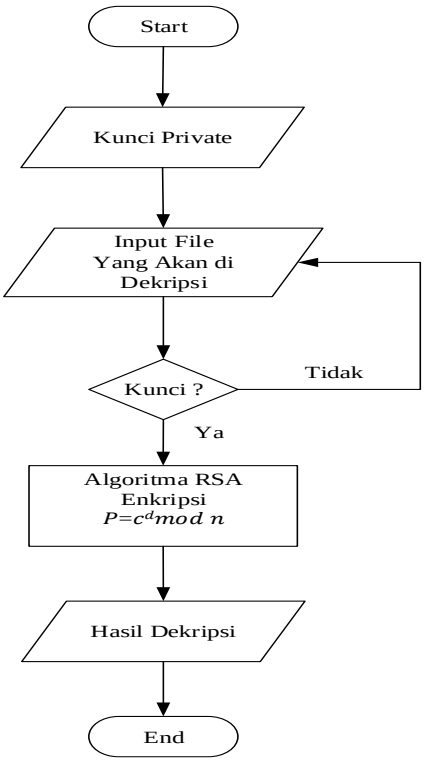
3.1 *Flowchart* Sistem Enkripsi RSA

P-ISSN: 2548-9704
E-ISSN: 2686-0880



Gambar 1. Flowchart Sistem Enkripsi RSA

3.2 *Flowchart* Sistem Dekripsi RSA



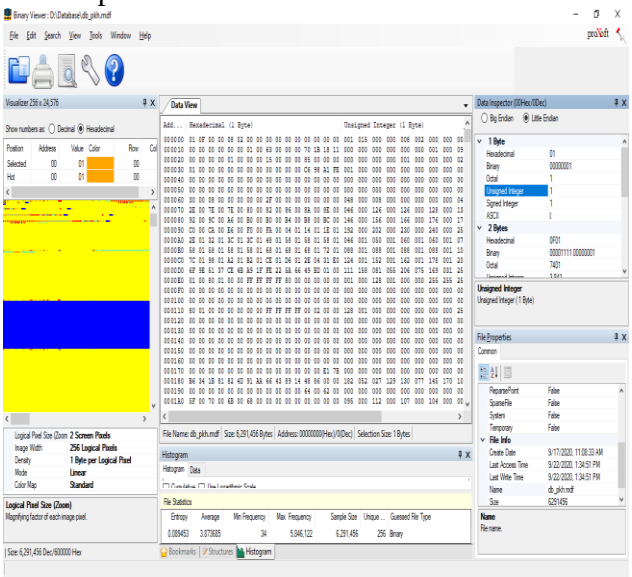
Gambar 2. Flowchart Sistem Enkripsi RSA

3.3 Perhitungan Enkripsi Database Algoritma RSA

Dari hasil enkripsi database dengan algoritma *RSA (Rivest Shamir Adleman)* dilakukan perhitungan:

p = 5
q = 83
n = p x q = 5 x 83 = 415
t(n) = (p - 1) x (q - 1) = (5 - 1) x (83 - 1) = 328
e = 215
Cari nilai d =
d . e mod t(n) = 1
d . 215 mod 328 = 1
119 . 215 mod 328 = 1
25585 mod 328 = 1
1 = 1
maka d = 119
Kunci Public yaitu : E = 215 N = 415
Kunci Private yaitu : D = 119 N = 415

Sebelum melakukan perhitungan secara manual, file tersebut diubah terlebih dahulu menggunakan *binary viewer*. File db_pkh.mdf akan seperti berikut:



Gambar 3. Database PKH

Dimana mencari Enkripsi adalah dengan rumus:

$C=p^e mod\ n$

Maka proses enkripsi *database* dengan algoritma *RSA (Rivest Shamir Adleman)* Sebagai Berikut:
Byte ke-1 = 1 = (1 ^ 215) mod 415 = 1

P-ISSN: 2548-9704
E-ISSN: 2686-0880

Byte ke-2 = 15 = (15 ^ 215) mod 415 = 185
Byte ke-3 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-4 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-5 = 8 = (8 ^ 215) mod 415 = 292
Byte ke-6 = 2 = (2 ^ 215) mod 415 = 138
Byte ke-7 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-8 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-9 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-10 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-11 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-12 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-13 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-14 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-15 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-16 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-17 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-18 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-19 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-20 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-21 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-22 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-23 = 1 = (1 ^ 215) mod 415 = 1
Byte ke-24 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-25 = 99 = (99 ^ 215) mod 415 = 124
Byte ke-26 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-27 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-28 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-29 = 112 = (112 ^ 215) mod 415 = 363
Byte ke-30 = 27 = (27 ^ 215) mod 415 = 93
Byte ke-31 = 24 = (24 ^ 215) mod 415 = 54
Byte ke-32 = 17 = (17 ^ 215) mod 415 = 78
Byte ke-33 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-34 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-35 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-36 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-37 = 1 = (1 ^ 215) mod 415 = 1
Byte ke-38 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-39 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-40 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-41 = 21 = (21 ^ 215) mod 415 = 341
Byte ke-42 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-43 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-44 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-45 = 118 = (118 ^ 215) mod 415 = 267
Byte ke-46 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-47 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-48 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-49 = 1 = (1 ^ 215) mod 415 = 1
Byte ke-50 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-51 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-52 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-53 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-54 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-55 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-56 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-57 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-58 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-59 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-60 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-61 = 247 = (247 ^ 215) mod 415 = 28
Byte ke-62 = 78 = (78 ^ 215) mod 415 = 177
Byte ke-63 = 112 = (112 ^ 215) mod 415 = 363
Byte ke-64 = 117 = (117 ^ 215) mod 415 = 223
Byte ke-65 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-66 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-67 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-68 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-69 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-70 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-71 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-72 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-73 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-74 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-75 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-76 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-77 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-78 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-79 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-80 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-81 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-82 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-83 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-84 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-85 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-86 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-87 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-88 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-89 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-90 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-91 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-92 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-93 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-94 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-95 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-96 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-97 = 48 = (48 ^ 215) mod 415 = 397

P-ISSN: 2548-9704
E-ISSN: 2686-0880

Byte ke-98 = 0 = (0 ^ 215) mod 415 = 0
Byte ke-99 = 8 = (8 ^ 215) mod 415 = 292
Byte ke-100 = 0 = (0 ^ 215) mod 415 = 0

3.6 Perhitungan Dekripsi Database Dengan Algoritma RSA

Selanjutnya mendekripsikan dengan algoritma *RSA (Rivest Shamir Adleman)* dengan melakukan perhitungan, Dimana mencari dekripsi adalah dengan rumus :

$P=c^d mod\ n$

Kunci Private yaitu : D = 119 N = 415

Maka proses dekripsi *database* dengan algoritma *RSA (Rivest Shamir Adleman)* Sebagai Berikut:

Byte ke-1 = 1 = (1 ^ 119) mod 415 = 1
Byte ke-2 = 185 = (185 ^ 119) mod 415 = 15
Byte ke-3 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-4 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-5 = 292 = (292 ^ 119) mod 415 = 8
Byte ke-6 = 138 = (138 ^ 119) mod 415 = 2
Byte ke-7 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-8 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-9 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-10 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-11 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-12 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-13 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-14 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-15 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-16 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-17 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-18 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-19 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-20 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-21 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-22 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-23 = 1 = (1 ^ 119) mod 415 = 1
Byte ke-24 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-25 = 124 = (124 ^ 119) mod 415 = 99
Byte ke-26 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-27 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-28 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-29 = 363 = (363 ^ 119) mod 415 = 112
Byte ke-30 = 93 = (93 ^ 119) mod 415 = 27
Byte ke-31 = 54 = (54 ^ 119) mod 415 = 24
Byte ke-32 = 78 = (78 ^ 119) mod 415 = 17
Byte ke-33 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-34 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-35 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-36 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-37 = 1 = (1 ^ 119) mod 415 = 1
Byte ke-38 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-39 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-40 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-41 = 341 = (341 ^ 119) mod 415 = 21
Byte ke-42 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-43 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-44 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-45 = 267 = (267 ^ 119) mod 415 = 118
Byte ke-46 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-47 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-48 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-49 = 1 = (1 ^ 119) mod 415 = 1
Byte ke-50 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-51 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-52 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-53 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-54 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-55 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-56 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-57 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-58 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-59 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-60 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-61 = 28 = (28 ^ 119) mod 415 = 247
Byte ke-62 = 177 = (177 ^ 119) mod 415 = 78
Byte ke-63 = 363 = (363 ^ 119) mod 415 = 112
Byte ke-64 = 223 = (223 ^ 119) mod 415 = 117
Byte ke-65 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-66 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-67 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-68 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-69 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-70 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-71 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-72 = 0 = (0 ^. 119) mod 415 = 0
Byte ke-73 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-74 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-75 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-76 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-77 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-78 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-79 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-80 = 0 = (0 ^ 119) mod 415 = 0

P-ISSN: 2548-9704
E-ISSN: 2686-0880

Byte ke-81 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-82 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-83 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-84 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-85 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-86 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-87 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-88 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-89 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-90 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-91 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-92 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-93 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-94 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-95 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-96 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-97 = 397 = (397 ^ 119) mod 415 = 48
Byte ke-98 = 0 = (0 ^ 119) mod 415 = 0
Byte ke-99 = 292 = (292 ^ 119) mod 415 = 8
Byte ke-100 = 0 = (0 ^ 119) mod 415 = 0

4. Pembahasan

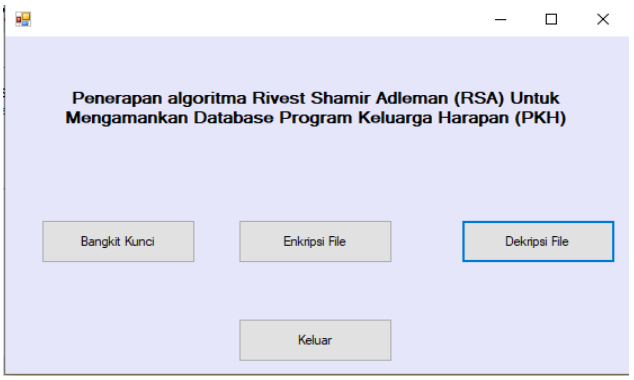
Aplikasi pengamanan database menggunakan metode RSA (*Rivest Shamir Adleman*) ini di bangun dengan tujuan menjaga kepemilikan agar tetap aman dari tindakan pencurian. Hal ini di lakukan dengan cara mengenkrip data tersebut dan dapat di dekripsi sebagai pembuktian kepemilikan dari database tersebut. Proses enkripsi dan dekripsi harus menggunakan aplikasi dan kunci yang sama.

4.1 Pembahasan Langkah Kerja Aplikasi

Tampilan dari sistem yang telah dirancang menggunakan aplikasi pemerograman *Visual Basic 2010*, dengan penerapan RSA (*Rivest Shamir Adleman*) dalam pengamanan database, yaitu sebagai berikut :

a. Tampilan Form Menu Utama

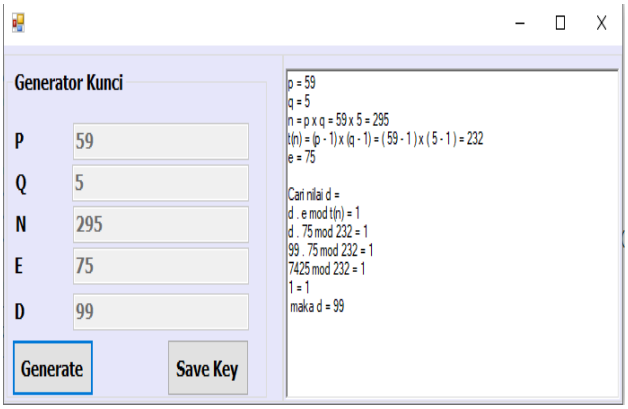
Tampilan dibawah ini merupakan tampilan menu utama Algoritma RSA. Adapun tampilanya yaitu seperti pada gambar dibawah ini:



Gambar 4. Menu Utama

b. Tampilan Menu Bangkit Kunci

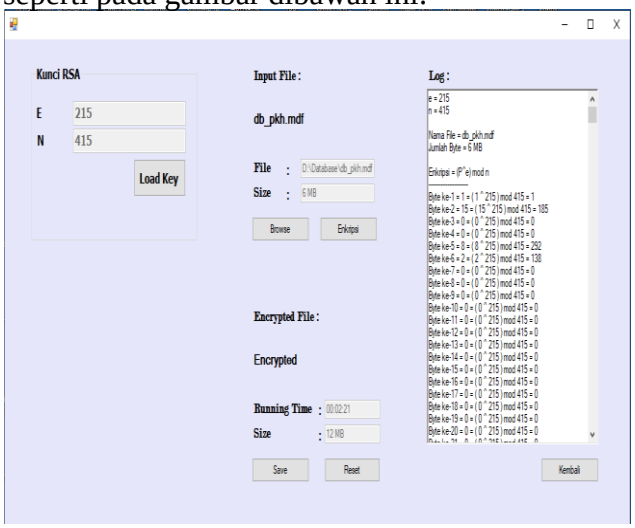
Pada tampilan bangkit kunci menampilkan beberapa baris untuk proses bangkit kunci:



Gambar 5. Bangkit Kunci

c. Tampilan Menu Enkripsi

Tampilan form Menu Algoritma RSA seperti pada gambar dibawah ini:

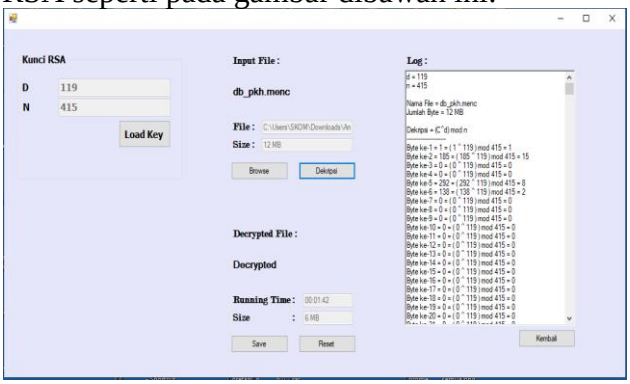


Gambar 6. Menu Enkripsi

P-ISSN: 2548-9704
E-ISSN: 2686-0880

d. Tampilan Tampilan Menu Dekripsi

Tampilan form Menu Dekripsi Algoritma RSA seperti pada gambar dibawah ini:



Gambar 7. Menu Dekripsi

4. KESIMPULAN

Berdasarkan hasil perancangan dan pembuatan program aplikasi kriptografi menggunakan metode RSA (*Rivest Shamir Adleman*) ini dapat di ambil kesimpulan sebagai berikut:

1. Aplikasi file database menggunakan algoritma RSA dapat digunakan untuk enkripsi dan mengembalikan data dengan cara mendekripsi dan tidak mengalami perubahan.
2. Tidak mudah untuk mengetahui isi file sulit untuk dipecahkan, bila tidak mengetahui metodenya, kecuali orang melakukan pelacakan kombinasi.
3. Aplikasi ini hanya dapat mengamankan file berekstensi .mdf dan .ldf karena pada pembuatan database sql server.

5. SARAN

Adapun saran penulis usulkan dalam pengembangan aplikasi ini adalah sebagai berikut:

1. Sistem yang di buat dalam penelitian ini hanya berbasis desktop, di harapkan pada penelitian berikutnya mampu menerapkan pada perangkat lain seperti website dan yang lainnya.
2. Aplikasi enkripsi dan dekripsi masih jauh dari kata sempurna, maka di harpakan kepada pembaca untuk menambah fitur lainnya maka program ini lebih baik dan sempurna.
3. Untuk penelitian lebih lanjut di harapkan dapat mengembangkan sistem yang telah ada untuk dapat melakukan enkripsi dan dekripsi.

DAFTAR PUSTAKA

[1]. Brink, J. van den (Pieter J. C. M., Monumentenstichting Baet en Borgh., N. B., Historische Vereniging Tweestromenland., J., & Azanuddin, A. (2014). Mensen van Maas en Waal. *Jurnal SAINTIKOM (Jurnal Sains Manajemen Informatika Dan Komputer)*, *18*(1), 30–34. https://ojs.trigunadharma.ac.id/index.php/jis/article/view/100

[2]. Ginting, A., Isnanto, R. R., & Windasari, I. P. (2015). Implementasi Algoritma Kriptografi RSA untuk Enkripsi dan Dekripsi Email. *Jurnal Teknologi Dan Sistem Komputer*, *3*(2), 253. https://doi.org/10.14710/jtsiskom.3.2.2015.253-258

[3]. Grehasen, G., & Mulyati, S. (2017). *Pengamanan Database Pada Aplikasi Test Masuk Karyawan Baru Berbasis Web Menggunakan Algoritma Kriptografi AES-128 Dan RC4 Geri*. *14*(1), 52–60.

[4]. Ladjamudin, A.-b. B. (2006). *Analisis Desain Sistem Informasi* . Yogyakarta: Graha Ilmu.

[5]. Nuryana, M., & Sulistiyono. (2014). Analisa dan Perancangan Sistem Front Office Hotel Bidakara Serang. *Jurnal Protekinfo*, *1*(September), 1–5.

[6]. Oetomo, B. D. (2006). *Perencanaan & Pembangunan Sistem Informasi.* Yogyakarta: C.V ANDI OFFSET.

[7]. Rini, B. 2011. *Microsoft Visual Basic 2010 Dan Mysql Untuk Aplikasi Point Of Sales.* Yogyakarta: Wahana Komputer.

[8]. Sugiarti, Y. 2013. *Analisis dan Perancangan UML(Unifed Modeling Language).* Yogyakarta: Graha Ilmu.

[9]. Supriyono. (2008). Pengujian sistem enkripsi-dekripsi dengan metode rsa untuk pengamanan dokumen. *Jurnal Sekolah Tinggi Teknologi Nuklir – BATAN*, *2*, 179–194. https://doi.org/10.1111/j.1365-

P-ISSN: 2548-9704
E-ISSN: 2686-0880

2923.2010.03863x.

[10]. Zainuddin, M. A., & Mulyana, D. I. (2016). Penerapan Algoritma Rsa Untuk Keamanan Pesan Instan Pada Perangkat Android. *Jurnal CKI On SPOT*, *9*(2), 105–114.