Terbit *online* pada laman : https://journal.fkpt.org/index.php/BIT

BULLETIN OF INFORMATION TECHNOLOGY (BIT)

ISSN : 2722-0524 (Media Online)

Implementasi Algoritma Mars Pada Penyandian Record Database

Alimah[1], Nelly Astuti Hasibuan.[2], Taronisokhi Zebua[3]

Program Studi Teknik Informatika STMIK Budi Darma Medan
Email: etekalimah@gmail.com

INFORMASI ARTIKEL

Sejarah Artikel:
Diterima Redaksi : 25 Desember 2019
Revisi Akhir : 02 Januari 2020
Diterima : 08 Januari 2020
Diterbitkan *Online* : 10 Maret 2020

KATA KUNCI

Cryptography, records, databases, Mars

KORESPONDENSI

etekalimah@gmail.com

A B S T R A C T

Catatan basis data dalam bentuk teks sangat banyak digunakan, sehingga sangat rentan terhadap pencurian data oleh pihak yang tidak berwenang. Untuk menjaga keamanan catatan database, itu bisa dilakukan dengan menggunakan teknik kriptografi. Teknik kriptografi dapat menyandikan catatan database dengan mengenkripsi mereka dalam bentuk kata sandi yang tidak dipahami. Algoritma Mars adalah algoritma yang menggunakan kunci 128-bit dan proses enkripsi terdiri dari 32 putaran. Algoritma simetri ini akan menghasilkan tingkat keamanan yang lebih tinggi terhadap catatan basis data karena dapat menyandikannya ke dalam kata sandi dengan proses yang cukup rumit yang akan mempersulit cryptanalyst untuk mengakses gambar. Penelitian ini menggunakan Algoritma Mars untuk proses enkripsi dan dekripsi, sehingga dalam proses itu perlu melalui beberapa langkah panjang untuk menghasilkan sandi akhir. Penelitian ini menjelaskan proses mengamankan catatan basis data dengan menyandikannya berdasarkan algoritma Mars, dalam bentuk kata sandi yang sulit dipahami dan dipahami orang lain. Ini dilakukan sebagai upaya untuk meminimalkan tindakan penyalahgunaan catatan basis data

1. PENDAHULUAN

Keamanan data adalah suatu hal yang sangat penting dalam menjaga kerahasiaan informasi terutama data yang hanya boleh diketahui oleh pihak yang berhak saja, terutama *record database*. Data tersebut bias bersifat penting atau bersifat rahasia, sehingga perlu diamankan agar tidak jatuh kepada pihak yang tidak bertanggungjawab.

Berdasarkan penelitian sebelumnya yang dilakukan oleh Yanti bahwa keamanan atau kerahasiaan data sangatlah penting[1]. Berdasarkan penelitian lainnya yang dilakukan oleh Neti menyatakan bahwa *database* merupakan kumpulan data yang disimpan dalam tabel yang saling bekaitan dan dapat dimanfaatkan sebagai sumber informasi bagi para pengguna[2]. Zebua juga menyatakan dalam penelitiannya bahwa *database* merupakan susunan data atau informasi dari suatu perusahahan yang tersimpan dalam media penyimpanan menjadi satu kesatuan yang utuh dan dijadikan sebagai sumber informasi yang optimal dan dapat digunakan oleh pengguna[3]. *Record databa* masih sering ditampilkan dalam bentuk teks sebagai informasi bagi pengguna, sehingga dapat dengan mudah diakses oleh pihak lain. Agar *record database* tidak mudah diakses, maka perlu diamankan berdasarkan algoritma kriptografi.

Kriptografi adalah suatu teknik yang digunakan untuk mengatasi kebocoran data dengan cara merubahnya kedalam kode tertentu[4]. Salah satu cara yang dapat digunakan untuk mengatasinya adalah dengan menggunakan algoritma *Mars* dimana algoritma ini dapat mengenkripsi dan dekripsi data dimasa sekarang danakan datang. *Mars* merupakan algoritma kriptografi *block chiper* kunci simetris yang memilikiukuran *block* 128 bit dankunci 128 bit sampai 400 bit[5]. *Mars* terdiridari 4 *word*, 128 bit memiliki ukuran 32 *word*. Operasi xor pada *mars* melibatkan penjumlahan, perkalian, dan pembagian untuk mengabungkan nilai data dan nilai kunci. Adapun keunggulan algoritma mars dapat mengenkripsi berbagai jenis *file* dan dikembalikan pada asal semula setelah didekripsi.

2. TEORITIS

Keamanan data merupakan permasalahan yang sangat penting, karena data yang belum terjamin keamanannya akan sangat mudah disalahgunakan oleh pihak yang tidak bertanggungjawab[6]. Keamanan sangat dibutuhkan oleh orang untuk melindungi dari hal-hal yang tidak diinginkan.

Kriptografi (*cripthographi*) berasal dari bahasa yunani "*cryptos*" artinya "*secret*" (rahasia), sedangkan "*graphein*" artinya "*writing*" (tulisan). Jadi kriptografi adalah ilmu dan seni yang bertujuan untuk menjaga kerahasiaan data dengan cara mengubahnya kedalam bentuk sandi yang tidak dimengerti lagi maknanya[7].

Kriptografi memiliki tujuan untuk melindungi data dari ancaman yang disengaja atau tidak disengaja dengan mengubah data menjadi sebuah sandi yang hanya dimengerti oleh pihak yang berhak saja[7]. Ada empat aspek yang terdapat pada pada kriptografi yaitu sebagai berikut:

1. Kerahasiaan (*confidentiality*)
   Kerahasiaan adalah layanan yang ditujukan untuk menjaga agar pesan tidak dapat dibaca oleh pihak-pihak yang tidak berhak. Layanan ini direalisasikan dengan menyadikan pesan menjadi *chiperteks*.
2. Integritas data (*Data integrity*)
   Integritas data merupakan layanan yang menjamin bahwapesan masih asli atau utuh belum pernah dimanipulasi selama pengiriman. Dengan kata lain, aspek keamanan ini dapat diungkapkan sebagai pertanyaan "Apakah pesan yang diterima masih asli atau atau tidak mengalami perubahan (modifikasi). Proses menjaga integritas data, sistem harus memiliki kemampuan untuk mendeteksi manipulasi pesan oleh pihak-pihak yang tidak berhak, antara lain penyisipan, penghapusan, dan pensubtitusian data lain kedalam pesan yang sebenarnya.
3. Otentikasi *(Authentication*)
   Authentication merupakan layanan yang terkait dengan identifikasi, baik mengidentifikasi kebenaran pihak-pihak yang berkomunikasi (*user authentication* atau *entity authentication*).Maupun mengidentifikasi kebenaran sumber pesan (*data origin authencitation*).
4. Nirpenyangkalan (*non repudation*)
   Nir penyangkalan adalah layanan untuk mencegah entitas yang berkomunikasi melakukan penyangkalan, yaitu pengirim pesan menyangkal melakukan pengiriman atau penerima pesan menyangkal telah menerima pesan.

Berdasarkan perkembangannya algoritma kriptografi dapat dibagi dua[8]yaitu:

1. Algoritma kriptografi klasik
   Kriptografi klasik merupakan algoritma yang hanya menggunakan satu kunci untuk mengamankan data. Teknik ini sudah digunakan beberapa abad yang lalu. *chipher*, *hill chipher transposisi columnar* dan lain-lain.
   Kriptografi klasik memiliki beberapa kategori:
   a. Algoritma kriptogrsfi klasik berbasis karakter.
   b. Menggunakan pena dan kertas saja, belum ada komputer.
   c. Termasuk kedalam kunci simetris.
2. Algoritma kriptografi modern
   Enkripsi kriptografi modern berbeda dengan enkripsi kriptografi klasik, karena enkripsi kriptografi modern sudah menggunakan komputer dalam pengoperasianya yang berfungsi untuk menggamankan data baik yang ditransfer melalui jaringan komputer maupun yang tidak baik. Hal ini sangat berguna untuk melindungi privasi, integrasi data, autentikasi dan anti penyangkala. Beberapa contoh algoritma kriptografi modern adalah *Data Encryption Standard* (DES), *Advance Encryption Standard* (AES), *International Data Encryption Algorithm* (IDEA), A5,RC4 dan lain-lain.

2.1. Algoritma Mars

Multivariate Adaptive Regression Splines (MARS) merupakan metode dengan pendekatan regresi nonparametrik yang pertama kali diperkenalkan oleh Friedman pada tahun 1991.Model MARS berguna untuk mengatasi permasalahan data berdimensi tinggi dan menghasilkan prediksi variabel respon yang akurat, dan menghasilkan model kontinu dalam *knot* berdasarkan nilai *Generalized Cross Validation* (GCV) terkecil.Permasalahan berdimensi tinggi adalah suatu permasalahan dengan jumlah variabel yang banyak serta ukuran sampel yang besar sehingga memerlukan perhitungan yang rumit. Data berdimensi tinggi yang dimaksud adalah data dengan ukuran $3 \leq v \leq 20$, dimana v adalah banyak variabel prediktor dan sampel data yang berukuran $50 \leq N \leq 1000$, dimana N untuk ukuran sampel[9].

Proses enkripsi MARS dibagi menjadi 3 tahap yaitu:

3. *Forward mixing*. Tahapan ini berfungsi untuk mencegah serangan terhadap *chosenplaintext*. Terdiri dari penambahan sub kunci pada setiap *word* data,diikuti dengan delapan iterasi *mixing* tipe-3 feitsal dengan berbasis S-box.
4. *Cryptographic core* dan *chipher*, terdiri dari enam belas iterasi transformasi kunci tipe-3 feitsel. Untuk menjamin bahwa proses enkripsi dan dekripsi mempunyai kekuatan yang sama, delapan iterasi pertama ditunjukkan dalam"*forward* mode" dan delapan iterasi terakhir ditunjukkan dalam"*backward* mode".
5. *Backward mixing*, berfungsi umtuk melindungi serangan kembali terhadap *chosen chipertext.* Tahap ini merupakan invers dari tahap pertama, terdiri dari delapan iterasi mixingtipe-3 feitsel(dalam *backward* mode) dengan berbasis S-box, diikuti dengan penggurangan sub kunci dari word data.

Volume 1, No. 1, Maret 2020, pp 37 - 51
ISSN 2722-0524 (media online)

2.2. Proses Dekripsi

Dekripsi adalah kebalikan dari proses enkripsi. Urutan proses yang dilakukan adalah *backward mixing*, *cryptographic core*, *forward mixing*. Pada proses dekripsi data yang dilakukan adalah menginput *file* yang telah dienkripsi sebelumnya atau disebut *chipertext*. Kemudian yang menginputkan *key* pertama untuk dilakukan dekripsi file menggunakan metode MARS.

Dibawah ini s-box algoritma Mars

0x09d0c479, 0x28c8ffe0, 0x84aa6c39, 0x9dad7287, 0x7dff9be3, 0xd4268361,
0xc96da1d4, 0x7974cc93, 0x85d0582e, 0x2a4b5705, 0x1ca16a62, 0xc3bd279d,
0x0f1f25e5, 0x5160372f, 0xc695c1fb, 0x4d7ff1e4, 0xae5f6bf4, 0x0d72ee46,
0xff23de8a, 0xb1cf8e83, 0xf14902e2, 0x3e981e42, 0x8bf53eb6, 0x7f4bf8ac,
0x83631f83, 0x25970205, 0x76afe784, 0x3a7931d4, 0x4f846450, 0x5c64c3f6,
0x210a5f18, 0xc6986a26, 0x28f4e826, 0x3a60a81c, 0xd340a664, 0x7ea820c4,
0x526687c5, 0x7eddd12b, 0x32a11d1d, 0x9c9ef086, 0x80f6e831, 0xab6f04ad,
0x56fb9b53, 0x8b2e095c, 0xb68556ae, 0xd2250b0d, 0x294a7721, 0xe21fb253,
0xae136749, 0xe82aae86, 0x93365104, 0x99404a66, 0x78a784dc, 0xb69ba84b,
0x04046793, 0x23db5c1e, 0x46cae1d6, 0x2fe28134, 0x5a223942, 0x1863cd5b,
0xc190c6e3, 0x07dfb846, 0x6eb88816, 0x2d0dcc4a, 0xa4ccae59, 0x3798670d,
0xcbfa9493, 0x4f481d45, 0xeafc8ca8, 0xdb1129d6, 0xb0449e20, 0x0f5407fb,
0x6167d9a8, 0xd1f45763, 0x4daa96c3, 0x3bec5958, 0xababa014, 0xb6ccd201,
0x854b3e95, 0x05bb9b43, 0x7dcd5dcd, 0xa02e926c, 0xfae527e5, 0x36a1c330,
0x3412e1ae, 0xf257f462, 0x3c4f1d71, 0x30a2e809, 0x68e5f551, 0x9c61ba44,
0x5ded0ab8, 0x75ce09c8, 0x9654f93e, 0x698c0cca, 0x243cb3e4, 0x2b062b97,
0x0f3b8d9e, 0x00e050df, 0xfc5d6166, 0xe35f9288, 0xc079550d, 0x0591aee8,
0x8e531e74, 0x75fe3578, 0x2f6d829a, 0xf60b21ae, 0x95e8eb8d, 0x6699486b,
0x901d7d9b, 0xfd6d6e31, 0x1090acef, 0xe0670dd8, 0xdab2e692, 0xcd6d4365,
0xe5393514, 0x3af345f0, 0x6241fc4d, 0x460da3a3, 0x7bcf3729, 0x8bf1d1e0,
0x14aac070, 0x1587ed55, 0x3afd7d3e, 0xd2f29e01, 0x29a9d1f6, 0xefb10c53,
0xcf3b870f, 0xb414935c, 0x664465ed, 0x024acac7, 0x59a744c1, 0x1d2936a7,
0xdc580aa6, 0xcf574ca8, 0x040a7a10, 0x6cd81807, 0x8a98be4c, 0xaccea063,
0xc33e92b5, 0xd1e0e03d, 0xb322517e, 0x2092bd13, 0x386b2c4a, 0x52e8dd58,
0x58656dfb, 0x50820371, 0x41811896, 0xe337ef7e, 0xd39fb119, 0xc97f0df6,
0x68fea01b, 0xa150a6e5, 0x55258962, 0xeb6ff41b, 0xd7c9cd7a, 0xa619cd9e,
0xbcf09576, 0x2672c073, 0xf003fb3c, 0x4ab7a50b, 0x1484126a, 0x487ba9b1,
0xa64fc9c6, 0xf6957d49, 0x38b06a75, 0xdd805fcd, 0x63d094cf, 0xf51c999e,
0x1aa4d343, 0xb8495294, 0xce9f8e99, 0xbffcd770, 0xc7c275cc, 0x378453a7,
0x7b21be33, 0x397f41bd, 0x4e94d131, 0x92cc1f98, 0x5915ea51, 0x99f861b7,
0xc9980a88, 0x1d74fd5f, 0xb0a495f8, 0x614deed0, 0xb5778eea, 0x5941792d,
0xfa90c1f8, 0x33f824b4, 0xc4965372, 0x3ff6d550, 0x4ca5fec0, 0x8630e964,
0x5b3fbbd6, 0x7da26a48, 0xb203231a, 0x04297514, 0x2d639306, 0x2eb13149,
0x16a45272, 0x532459a0, 0x8e5f4872, 0xf966c7d9, 0x07128dc0, 0x0d44db62,
0xafc8d52d, 0x06316131, 0xd838e7ce, 0x1bc41d00, 0x3a2e8c0f, 0xea83837e,
0xb984737d, 0x13ba4891, 0xc4f8b949, 0xa6d6acb3, 0xa215cdce, 0x8359838b,
0x6bd1aa31, 0xf579dd52, 0x21b93f93, 0xf5176781, 0x187dfdde, 0xe94aeb76,
0x2b38fd54, 0x431de1da, 0xab394825, 0x9ad3048f, 0xdfea32aa, 0x659473e3,
0x623f7863, 0xf3346c59, 0xab3ab685, 0x3346a90b, 0x6b56443e, 0xc6de01f8,
0x8d421fc0, 0x9b0ed10c, 0x88f1a1e9, 0x54c1f029, 0x7dead57b, 0x8d7ba426,
0x4cf5178a, 0x551a7cca, 0x1a9a5f08, 0xfcd651b9, 0x25605182, 0xe11fc6c3,
0xb6fd9676, 0x337b3027, 0xb7c8eb14, 0x9e5fd030,
0x6b57e354, 0xad913cf7, 0x7e16688d, 0x58872a69, 0x2c2fc7df, 0xe389ccc6,
0x30738df1, 0x0824a734, 0xe1797a8b, 0xa4a8d57b, 0x5b5d193b, 0xc8a8309b,
0x73f9a978, 0x73398d32, 0x0f59573e, 0xe9df2b03, 0xe8a5b6c8, 0x848d0704,
0x98df93c2,0x720a1dc3,0x684f259a,0x943ba848, 0xa6370152, 0x863b5ea3,
0xd17b978b,0x6d9b58ef,0x0a700dd4,0xa73d36bf, 0x8e6a0829, 0x8695bc14,
0xe35b3447,0x933ac568,0x8894b022,0x2f511c27, 0xddfbcc3c, 0x006662b6,
0x117c83fe,0x4e12b414,0xc2bca766,0x3a2fec10, 0xf4562420, 0x55792e2a,
0x46f5d857,0xceda25ce,0xc3601d3b,0x6c00ab46, 0xefac9c28, 0xb3c35047,

Volume 1, No. 1, Maret 2020, pp 36 - 49
ISSN 2722-0524 (media online)
0x611dfee3, 0x257c3207, 0xfdd58482, 0x3b14d84f, 0x23becb64, 0xa075f3a3, 0x088f8ead, 0x07adf158, 0x7796943c, 0xfacabf3d, 0xc09730cd, 0xf7679969, 0xda44e9ed, 0x2c854c12, 0x35935fa3, 0x2f057d9f, 0x690624f8, 0x1cb0bafd, 0x7b0dbdc6, 0x810f23bb, 0xfa929a1a, 0x6d969a17, 0x6742979b, 0x74ac7d05, 0x010e65c4, 0x86a3d963, 0xf907b5a0, 0xd0042bd3, 0x158d7d03, 0x287a8255, 0xbba8366f, 0x096edc33, 0x21916a7b, 0x77b56b86, 0x951622f9, 0xa6c5e650, 0x8cea17d1, 0xcd8c62bc, 0xa3d63433, 0x358a68fd, 0x0f9b9d3c, 0xd6aa295b, 0xfe33384a, 0xc000738e, 0xcd67eb2f, 0xe2eb6dc2, 0x97338b02, 0x06c9f246, 0x419cf1ad, 0x2b83c045, 0x3723f18a, 0xcb5b3089, 0x160bead7, 0x5d494656, 0x35f8a74b, 0x1e4e6c9e, 0x000399bd, 0x67466880, 0xb4174831, 0xacf423b2, 0xca815ab3, 0x5a6395e7, 0x302a67c5, 0x8bdb446b, 0x108f8fa4, 0x10223eda, 0x92b8b48b, 0x7f38d0ee, 0xab2701d4, 0x0262d415, 0xaf224a30, 0xb3d88aba, 0xf8b2c3af, 0xdaf7ef70, 0xcc97d3b7, 0xe9614b6c, 0x2baebff4, 0x70f687cf, 0x386c9156, 0xce092ee5, 0x01e87da6, 0x6ce91e6a, 0xbb7bcc84, 0xc7922c20, 0x9d3b71fd, 0x060e41c6, 0xd7590f15, 0x4e03bb47, 0x183c198e, 0x63eeb240, 0x2ddbf49a, 0x6d5cba54, 0x923750af, 0xf9e14236, 0x7838162b, 0x59726c72, 0x81b66760, 0xbb2926c1, 0x48a0ce0d, 0xa6c0496d, 0xad43507b, 0x718d496a, 0x9df057af, 0x44b1bde6, 0x054356dc, 0xde7ced35, 0xd51a138b, 0x62088cc9, 0x35830311, 0xc96efca2, 0x686f86ec, 0x8e77cb68, 0x63e1d6b8, 0xc80f9778, 0x79c491fd, 0x1b4c67f2, 0x72698d7d, 0x5e368c31, 0xf7d95e2e, 0xa1d3493f, 0xdcd9433e, 0x896f1552, 0x4bc4ca7a, 0xa6d1baf4, 0xa5a96dcc, 0x0bef8b46, 0xa169fda7, 0x74df40b7, 0x4e208804, 0x9a756607, 0x038e87c8, 0x20211e44, 0x8b7ad4bf, 0xc6403f35, 0x1848e36d, 0x80bdb038, 0x1e62891c, 0x643d2107, 0xbf04d6f8, 0x21092c8c, 0xf644f389, 0x0778404e, 0x7b78adb8, 0xa2c52d53, 0x42157abe, 0xa2253e2e, 0x7bf3f4ae, 0x80f594f9, 0x953194e7, 0x77eb92ed, 0xb3816930, 0xda8d9336, 0xbf447469, 0xf26d9483, 0xee6faed5, 0x71371235, 0xde425f73, 0xb4e59f43, 0x7dbe2d4e, 0x2d37b185, 0x49dc9a63, 0x98c39d98, 0x1301c9a2, 0x389b1bbf, 0x0c18588d, 0xa421c1ba, 0x7aa3865c, 0x71e08558, 0x3c5cfcaa, 0x7d239ca4, 0x0297d9dd, 0xd7dc2830, 0x4b37802b, 0x7428ab54, 0xaeee0347, 0x4b3fbb85, 0x692f2f08, 0x134e578e, 0x36d9e0bf, 0xae8b5fcf, 0xedb93ecf, 0x2b27248e, 0x170eb1ef, 0x7dc57fd6, 0x1e760f16, 0xb1136601.

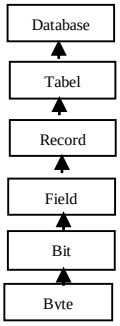
2.3. Database

Database merupakan susunan data atau informasi dari suatu perusahahan yang tersimpan dalam media penyimpanan menjadi satu kesatuan yang utuh dan dijadikan sebagai sumber informasi yang optimal dan dapat digunakan oleh para pengguna[10].

Dalam suatu pengolahan data agar dapat berjalan dengan baik suatu data, maka data tersebut perlu disusun secatar teratur. Kontribusi *database* dibagi dalam enam tingkatan[11], yaitu:

1. *Bit* adalah bagian terkecil yang dapat diwakili dengan angka biner yang terdiri atas dua macam nilai saja, yaitu 0 dan 1. Sistem angka biner ini dapat digunakan sebagai dasar komunikasi antar manusia dan komputer.
2. *Byte* adalah kumpulan dari delapan buah bit yang sejenis yang mewakili sebuah karakter. Dengan kombinasi 8 bit, dapat diperoleh 256 karakter.
3. *Field* atau kolom adalah kumpulan beberapa *byte* yang sejenis yang juga disebut dengan data. Jadi f*ield* ibarat kumpulan karakter yang membentuk suatu kata.
4. *Record* atau baris adalah kumpulan data yang saling berhubungan diantara satu sama lainnya. Suatu *record* dapat dikenali dengan *field* kunci.
5. *File* atau tabel adalah kumpulan *record* yang sejenis yang memiliki elemen yang sama, atribut yang sama, namun berbeda data valuenya.
6. *Database* merupakan kumpulan tabel yang teritegrasi secara logis yang dapat digunakan secara terus menerus dan juga sebagai tempat penyimpanan data.

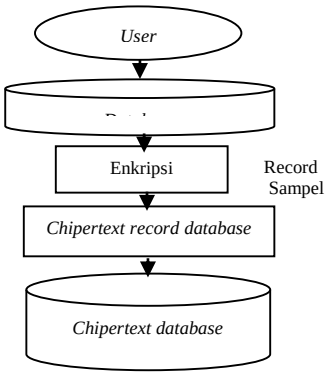
BULLETIN OF INFORMATION TECHNOLOGY (BIT)
Volume 1, No. 1, Maret 2020, pp 37 - 51
ISSN 2722-0524 (media online)



Gambar 1 Hierarki *Database*

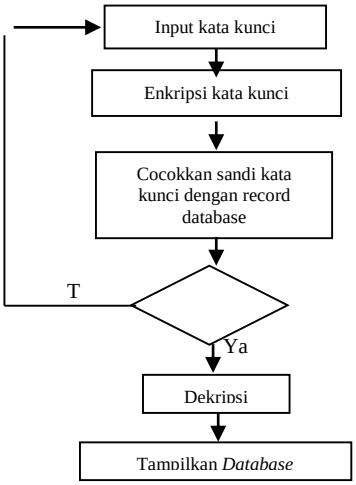
2.4. Analisa

Keamanan *record database* merupakan salah satu data yang perlu diamankan agar tidak jatuh ketangan orang yang salah atau tidak bertanggung jawab. Sesuai dengan perumusan masalah bagaimana prosedur penggamanan *record databse* menggunakan teknik kriptografi Mars dapat dijelaskan pada gambar di bawah ini.



Gambar 2 Analisa penyandian *record database* berdasarkan kriptografi

Penggamanan *record database* pada penelitian ini dimulai dengan membuka *database* tersebut dan memilih salah satu tabel kemudian dienkripsikan berdasarkan teknik kriptografi yang mana menghasilkan *record database* yang telah berubah dalam bentuk sandi dan tidak dipahami. Proses dekripsi terjadi disaat pengaksessan ulang *database*, dimana dimulai dengan menginputkan kata kunci yang kemudian kata kunci tersebut dienkripsi dan dicocokan pada *chiper* yang tersimpan dalam *database.* Bila *chipertext* kata kunci cocok dengan *chiper* yang tersimpan dalam *database* maka data yang berhubungan dengan kata kunci akan didekripsida kemudian ditampilkan.

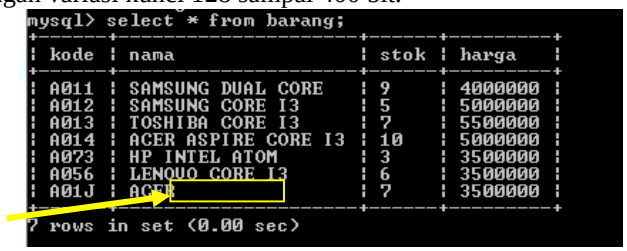


Gambar 3 Skema dekripsi

BULLETIN OF INFORMATION TECHNOLOGY (BIT)
Volume 1, No. 1, Maret 2020, pp 36 - 49
ISSN 2722-0524 (media online)

2.5. Contoh Kasus

Terdapat nama barang yang tersimpan di dalam suatu *database* yang dibuat menggunakan MySQL dengan nama Penjualan barang elektronik. Dimana *record database* ini akan diamankan menggunakan teknik kriptografi agar kerahasiaan data lebih terjamin. *Record database* diamankan dengan mengekripsikan *record* ke dalam bentuk sandi menggunakan algoritma Mars dengan variasi kunci 128 sampai 400 bit.



```
mysql> select * from barang;
+------+----------------------+------+----------+
| kode | nama                 | stok | harga    |
+------+----------------------+------+----------+
| A011 | SAMSUNG DUAL CORE    | 9    | 4000000  |
| A012 | SAMSUNG CORE I3      | 5    | 5000000  |
| A013 | TOSHIBA CORE I3      | 7    | 5500000  |
| A014 | ACER ASPIRE CORE I3  | 10   | 5000000  |
| A073 | HP INTEL ATOM        | 3    | 3500000  |
| A056 | LENOVO CORE I3       | 6    | 3500000  |
| A01J | ACER                 | 7    | 3500000  |
+------+----------------------+------+----------+
7 rows in set (0.00 sec)
```

Gambar 4 *Database* penjualan barang elektronik

Record sampel : ACER
KEY: Skripsi_ALIMAH24
K[0] = Skri (01010011 01101011 01110010 01101001)
K[1] = psi_ (01110000 01110011 01101001 01011111)
K[2] = ALIM (01000001 01001100 01001001 01001101)
K[3] = AH24 (01000001 01001000 00110010 00110100)
Lalu inisialisasi T[] dengan kunci data, dimana T[] adalah *array* sementara yang terdiri dari 15 words dan n adalah jumlah words di kunci *buffer* k[], $(4 \leq n \leq 14)$ Misalkan n = 4
T[0n - 1] = k[0..n - 1],
T[n] = n
T[n + 1..14] = 0
T[0..3] = k[0..3],
T[4] = 4
T[5..14] = 0
Empat iterasi, menghitung 10 words untuk tiap K[]
for j = 0 to 3 do
for i = 0 to 14 do
Transformasi Linear
T[i] = ((T[i-7 mod 15] $\oplus$ T[i-2 mod 15])<<< 3) $\oplus$ (4i + j)
Untuk i = 0 to 14
j = 0
T[0] = ((T[0-7 mod 15] $\oplus$ T[0-2 mod 15])<<< 3) $\oplus$ (4.0 + 0)
= ((T[8] $\oplus$ T[13])<<< 3) $\oplus$ 0
= ((0 $\oplus$ 0<<< 3) $\oplus$ 0
= (00000000 00000000 $\oplus$00000000 00000000 <<< 3) $\oplus$ 0
= (0000000 00000000 <<<3) $\oplus$ 0
= (00000000 00000000) $\oplus$ 0
= 00000000 00000000
T[1] = ((T[1-7 mod 15] $\oplus$ T[1-2 mod 15])<<< 3) $\oplus$ (4.1 + 0)
= ((T[9] $\oplus$ T[14])<<< 3) $\oplus$ 4
= ((0 $\oplus$ 0<<< 3) $\oplus$ 4
= (00000000 00000000 $\oplus$00000000 00000000 <<< 3) $\oplus$ 4
= (00000000 00000000 <<<3) $\oplus$ 4
=00000000 00000000 $\oplus$ 00000000 00000100
= 00000000 00000100
T[14] = ((T[14-7 mod 15] $\oplus$ T[14-2 mod 15])<<< 3) $\oplus$ (4.14 + 3)
= ((T[7] $\oplus$ T[12])<<< 3) $\oplus$ 59
= ((0 $\oplus$ 0)<<< 3) $\oplus$ 59
= (0000000 $\oplus$ 00000000 00000000 00000000 00000000 <<< 3) $\oplus$ 59
= (00000000 00000000 00000000 00000000 <<<3) $\oplus$ 59
=0$\oplus$ 00000000 00000000 00000000 00111011
= 00111011

Volume 1, No. 1, Maret 2020, pp 37 - 51
ISSN 2722-0524 (media online)

Setelah itu dilakukan empat pengulangan lagi untuk mendapatkan T[i] baru.

For i = 0 to 14 do
T[i] = (T[i] + S[low 9 bits of T[i - 1 mod 15]])<<< 9
Dimulai dari i = 0
T[0] = (T[0] + S[low 9 bits of T[0 - 1 mod 15]])<<< 9
= (00000000 + S[low 9 bits of T[14]])<<< 9
= (00000000+ S[59]) <<< 9
= (00000000+ S[0x1863cd5b]) <<< 9
= (00000000+ 0001 1000 0110 0011 1100 1101 0101 1011) <<< 9
= 0001 1000 0110 0011 1100 1101 0101 1011 <<<9
= 110 0011 1100 1101 0101 1011 0001 1000 0
T[14] = (T[14]+ S[low 9 bits of T[14 - 1 mod 15]])<<< 9
= (00111011+ S[low 9 bits of T[13]]) <<< 9
= (00111011+ S[55]) <<< 9
= (00111011+ S[0x23db5c1e]) <<< 9
= (00111011+10 0011 1101 1011 0101 1100 0001 1110)<<< 9
= 10 0011 1101 1011 0101 1100 0101 1001<<<9
= 101 1011 0101 1100 0101 1001 0010 0011 1

Dilakukan proses untuk mendapatkan *expansion key* dari K[0] sampai dengan K[39]

Untuk i = 0.....9 dan j = 0
K[10j+i] = T[4i mod 15]
K[10.0 + 0] = T[4.0 mod 15]
K[0] = T[0]
K[1] = T[4]
K[2] = T[8]
K[3] = T[12]
K[4] = T[1]
K[5] = T[5]
K[6] = T[9]
K[7] = T[13]
K[8] = T[2]
K[9] = T[6]

proses enkripsi

Proses enkripsi terdiri dari tiga proses yaitu *fordware mixing, cryptoghraphic core* , dan *backwards mixing*.

a. *Forward mixing*

D = ACER
= D[0], D[1], D[2], D[3]
= 01000001, 01000011, 01000101, 01010010
D[i] = D[i] + K[i]
D[0] = D[0] + K[0]
= 01000001 + 011 1111 0110 0000 1010 0101 1011 0011 1
= 111 1110 1100 0001 0100 1011 1010 1000
D[1] = D[1] + K[1]
= 01000011+ 111 1100 1100 0111 0100 0101 0101 1111 0
= 1111 1001 1000 1110 1000 1011 0000 0001
D[2] = D[2] + K[2]
= 01000101+ 101 0010 1110 1111 1110 0100 1100 1110 0
= 1010 0101 1101 1111 1100 1001 1110 0001
D[3] = D[3] + K[3]
= 01010010 + 001 0101 1100 0010 0010 1110 1100 0110 1
= 10 1011 1000 0100 0101 1101 1101 1111
For i= 0 to 7 do
D[1] =D[1] ⊕ S0[low byte of D[0]]
= 1111 1001 1000 1110 1000 1011 0000 0001 ⊕ S0[111 1110 1100 0001 0100 1011 1010 1000]
= 1111 1001 1000 1110 1000 1011 0000 0001⊕ 0x09d0c479
=1111 1001 1000 1110 1000 1011 0000 0001⊕ 1001 1101 0000 1100 0100 0111 1001
= 1111 0000 0101 1110 0100 1111 0111 1000

Volume 1, No. 1, Maret 2020, pp 36 - 49
ISSN 2722-0524 (media online)

D[1] =D[1] +S1[2nd byte of D[0]]
= 1111 0000 0101 1110 0100 1111 0111 1000+S0[111 1110 1100 0001 0100 1011 1010 1000]
= 1111 0000 0101 1110 0100 1111 0111 1000+ 0x09d0c479
= 1111 0000 0101 1110 0100 1111 0111 1000 +1001 1101 0000 1100 0100 0111 1001
= 1111 1010 0010 1111 0001 0011 1111 0001

b. *Cryptographyc core*

For i = 0 to 15 do
For i = 0
(out1; out2; out3) = E-function(D[0];K[2i+4];K[2i+ 5])
= E-Function (11 1001 1011 0001 0010 0011 1100 0110 0110; K[4]; K[5])
= E-Function (111 1110 1100 0001 0100 1011 1010 1000; 0110 0011 0100 1000 1110 1110 1 1001 0100; 100 1111 0011 0101 0101 1010 0101 0000 0
Out1 = 11 1001 1011 0001 0010 0011 1100 0110 0110
Out2 = 0110 0011 0100 1000 1110 1110 1 1001 0100;
Out3 = 100 1111 0011 0101 0101 1010 0101 0000 0
D[0] = D[0] <13
= 11 1001 1011 0001 0010 0011 1100 0110 0110<<<13
= 1 0010 0011 1100 0110 0110 11 1001 1011 000

D[2] =D[2] +out2
= 1 0001 1100 0001 1101 1000 1111 1001 0000 + 0110 0011 0100 1000 1110 1110 1 1001 0100
= 1 1110 0010 1010 1111 0110 1101 0010 0100
If I < 8 then
D[1] =D[1] +out1
= 1 0100 1011 1010 1001 1101 0010 0100 0001+ 11 1001 1011 0001 0010 0011 1100 0110 0110
= 100 1110 0110 1011 1100 0000 1110 1010 0111

D[3] =D[3] ⊕out3
= 1110 1011 0100 1000 1010 1101 1 0011 1111 ⊕ 100 1111 0011 0101 0101 1010 0101 0000 0
= 1 0100 1000 1111 1011 1110 1111 1001 1111
(D[3];D[2];D[1];D[0])← (D[0];D[3];D[2];D[1])

Backward mixing
For i = 1
(out1; out2; out3) = E-function(D[0];K[2.1+4];K[2.1+ 5])
= E-Function (1 0010 0011 1100 0110 0110 11 1001 1011 000; K[6]; K[7])
= E-Function (1 0010 0011 1100 0110 0110 11 1001 1011 000; 001 1100 1001 0010 1101 1101 0001 1110 0; 100 0000 0100 1010 1001 1101 1001 1001 0)
Out1 = 1 0010 0011 1100 0110 0110 11 1001 1011 000
Out2 = 001 1100 1001 0010 1101 1101 0001 1110 0;
Out3 = 100 0000 0100 1010 1001 1101 1001 1001 0
D[0] = D[0] <13
= 1 0010 0011 1100 0110 0110 11 1001 1011 000<<<13
= 0110 11 1001 1011 000 1 0010 0011 1100 0110
D[2] =D[2] +out2
= 1 1110 0010 1010 1111 0110 1101 0010 0100 +001 1100 1001 0010 1101 1101 0001 1110 0
= 10 0001 1011 1101 0101 0010 0111 0110 0000
If I < 8 then
D[1] =D[1] +out1
= 100 1110 0110 1011 1100 0000 1110 1010 0111+1 0010 0011 1100 0110 0110 11 1001 1011 000
= 111 0010 1110 0100 1000 1110 1011 0111 1111
D[3] =D[3] ⊕out3
= 1 0100 1000 1111 1011 1110 1111 1001 1111⊕ 100 0000 0100 1010 1001 1101 1001 1001 0
= 1 1100 1000 0110 1110 1101 0100 1010 1101
(D[3];D[2];D[1];D[0])← (D[0];D[3];D[2];D[1])
Setelah cipher akhir didapatkan dalam bentuk biner, kemudian konversikan ke karakter, yang mana hasilnya adalah jðè¢

BULLETIN OF INFORMATION TECHNOLOGY (BIT)
Volume 1, No. 1, Maret 2020, pp 37 - 51
ISSN 2722-0524 (media online)

Proses Dekripsi

Proses dekripsi kebalikan dari proses enkripsi yanmjuga terdiri dari tiga proses yaitu yaitu *fordware mixing*, *cryptoghraphic core* , dan *backwards mixing*.

a. Forward mixing

For i =0 to 3 do
D[i] =D[i]+K[36 +i]
D[0] = D[0] + K[36]
= 11 0101 0011 1011 0111 0001 1110 0101 0101 + 001 1100 1001 0010 1101 1101 0001 1110 0
= 11 1000 1100 1101 1100 1101 1000 1001 0001
D[1] = D[1] + K[37]
= 1110 0110 1101 1101 0110 0101 1101 0101 1000+ 001 1100 1001 0010 1101 1101 0001 1110 0
= 1110 1010 0110 1111 1100 0001 0111 1001 0100
D[2] = D[2] + K[38]
= 11 0000 0100 1001 1000 1111 0110 1010 0111+ 001 1100 1001 0010 1101 1101 0001 1110 0
= 11 0011 1101 1011 1110 1011 0000 1110 0011
D[3] = D[3] + K[39]
= 110 1010 1111 0000 1110 1000 1010 0010+ 001 1100 1001 0010 1101 1101 0001 1110 0
= 1010 0100 0001 0110 1010 0010 1101 1110

b. *Cryptoghraphic core*

For i = 15 down to 0 do
For I = 15
(D[3];D[2];D[1];D[0]) (D[2];D[1];D[0];D[3])
D[0] = D[0] >13
= 1 0010 0010 0010 1000 1111 0001 1011 0111 1001> 13
= 1 1011 0111 10011 0010 0010 0010 1000 1111 000
(out1; out2; out3) =E-function(D[0];K[2i+4];K[2i+ 5])
= E-Function(D[0]; K[2.15 + 4]; K[2.15+5])
= E-Function(D[0]; K[34]; K[35])
= E-Function(1 1011 0111 10011 0010 0010 0010 1000 1111 000; 0110 0011 0100 1000 1110 1110 1 1001 0100; 100 1111 0011 0101 0101 1010 0101 0000 0)
Out1 = 1 1011 0111 10011 0010 0010 0010 1000 1111 000
Out2 = 0110 0011 0100 1000 1110 1110 1 1001 0100
Out3 = 100 1111 0011 0101 0101 1010 0101 0000 0
D[2] =D[2] - out2
= 11 0100 0001 1100 1001 0010 1010 1111 0011-0110 0011 0100 1000 1110 1110 1 1001 0100
= 10 0111 1011 0011 0111 0100 1101 0101 1111
D[3] =D[3] - out1
= 1 0010 1011 1100 0110 0000 0010 1000 1010- 1 1011 0111 10011 0010 0010 0010 1000 1111 000
= 1111 1111 1111 1111 1111 1111 1110 0101 1011 0010 0011 0100 1110 1110 0001 0010
D[1] =D[1] ⊕ out3
= 1110 1010 1111 0010 1100 1101 0011 1110 1101 ⊕100 1111 0011 0101 0101 1010 0101 0000 0
= 1111 1111 1111 1111 1111 1111 1110 1011 1000 0011 0111 0010 1000 1001 0101 1111

c. Backwards mixing

For i = 7 down to 0 do
(D[3];D[2];D[1];D[0])← (D[2];D[1];D[0];D[3])
D[0] =D[0] <24
= 1 1011 0111 10011 0010 0010 0010 1000 1111 000 <24
= 0 1000 1111 0001 1011 0111 10011 0010 0010 001
D[3] =D[3] ⊕ S1[high byte of D[0]]
= 1111 1111 1111 1111 1111 1111 1110 0101 1011 0010 0011 0100 1110 1110 0001 0010 ⊕ S1[0 1000 1111 0001 1011 0111 10011 0010 0010 001]
= 1111 1111 1111 1111 1111 1111 1110 0101 1011 0010 0011 0100 1110 1110 0001 0010⊕ 0x0d72ee46,
= 1111 1111 1111 1111 1111 1111 1110 0101 1011 0010 0011 0100 1110 1110 0001 0010⊕ 1101 0111 0010 1110 1110 0100 0110
= 1111 1111 1111 1111 1111 1111 1110 0101 1011 1111 0100 0110 0000 0000 0101 0100
D[2] =D[2] - S0[3rd byte ofD[0]]

=10 0111 1011 0011 0111 0100 1101 0101 1111– S0[0 1000 1111 0001 1011 0111 10011 0010 0010 001]
= 10 0111 1011 0011 0111 0100 1101 0101 1111–0x28c8ffe0,
= 10 0111 1011 0011 0111 0100 1101 0101 1111 –10 1000 1100 1000 1111 1111 1110 0000
= 10 0101 0010 0110 1110 0100 1101 0111 1111

D[1] =D[1] - S1[2nd byte ofD[0]]
= 1111 1111 1111 1111 1111 1111 1110 1011 1000 0011 0111 0010 1000 1001 0101 1111- S1[0 1000 1111 0001 1011 0111 10011 0010 0010 001]
= 1111 1111 1111 1111 1111 1111 1110 1011 1000 0011 0111 0010 1000 1001 0101 1111- 0x28c8ffe0
= 1111 1111 1111 1111 1111 1111 1110 1011 1000 0011 0111 0010 1000 1001 0101 1111- 10 1000 1100 1000 1111 1111 1110 0000
= 1111 1111 1111 1111 1111 1111 1110 1011 0101 1010 1010 1001 1000 1001 0111 1111

D[1] =D[1] ⊕ S0[low byte of D[0]]
= 1111 1111 1111 1111 1111 1111 1110 1011 0101 1010 1010 1001 1000 1001 0111 1111⊕ S0[0 1000 1111 0001 1011 0111 10011 0010 0010 001]
= 1111 1111 1111 1111 1111 1111 1110 1011 0101 1010 1010 1001 1000 1001 0111 1111⊕ 0x28c8ffe0
= 1111 1111 1111 1111 1111 1111 1110 1011 0101 1010 1010 1001 1000 1001 0111 1111 ⊕ 10 1000 1100 1000 1111 1111 1110 0000
= 1111 1111 1111 1111 1111 1111 1110 1011 0111 0010 0110 0001 0111 0110 1001 1111

For i = 0 to 3 do
= 01000001, 01000011, 01000101, 01010010

D[i] = D[i] + K[i]

D[0] = D[0] + K[0]
= 111 1110 1100 0001 0100 1011 1010 1000 + 011 1111 0110 0000 1010 0101 1011 0011 1
= 01000001

D[1] = D[1] + K[1]
= 1111 1001 1000 1110 1000 1011 0000 0001 +111 1100 1100 0111 0100 0101 0101 1111 0
= 01000011

D[2] = D[2] + K[2]
= 1010 0101 1101 1111 1100 1001 1110 0001 + 101 0010 1110 1111 1110 0100 1100 1110 0
= 01000101

D[3] = D[3] + K[3]
= 10 1011 1000 0100 0101 1101 1101 1111 + 001 0101 1100 0010 0010 1110 1100 0110 1
= 01010010

Berdasarkan proses dekripsi di atas, maka didapatkan plainteks akhir yaitu D[0], D[1], D[[2], D[3] dalam bentuk binary 01000001;01000011;01000101;01010010 yang kemudian dikonversikan ke bentuk karakter yaitu ACER.

3. IMPLEMENTASI

Bab ini akan melakukan pembahasan tentang pengujian dan analisa hasil dari program aplikasi yang telah dibuat. Tujuan dari pengujian ini adalah untuk mengetahui apakah aplikasi yang telah dibuat sesuai dengan peracanganya. Selain itu juga, untuk mengetahui detail dari jalanya aplikasi serta kesalahan yang ada untuk jadikan pengembangan dan perbaikan lanjut. Proses pengujian ini dibutuhkan beberapa peralatan-peralatan berupa perangkat keras dan perangkat lunak.

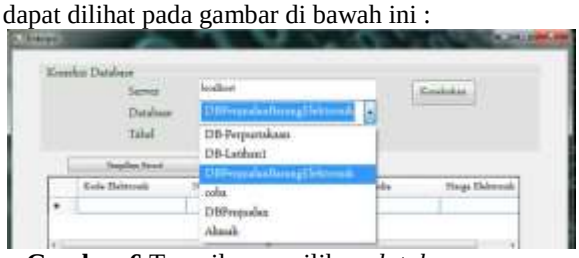
3.1. Tampilan Program Aplikasi

Tampilan *output* ini didapat dari proses pengujian sistem. Pengujian sistem aplikasi bertujuan untuk mengetahui apakah aplikasi berjalan dengan sesuai fungsinya, yaitu mengenkripsi dan dekripsi citra digital serta pengiriman pesan melalui jaringan. Pengujian memiliki 2 tahap yaitu proses enkripsi citra digital awal dan proses dekripsi citra hasil enkripsi.

1. Proses Enkripsi
   Proses enkripsi memiliki beberapa tahap yaitu, proses mengkoneksikan server, pemilihan *database*, pemilihan tabel, menampilkan *record*, memasukan kunci dan proses enkripsi.
   a. Mengkoneksikan *server*
      Adapun proses mengkoneksikan *server database* dilakukan dengan menginputkan *server* dan memilih *buton* koneksikan seperti pada gambar di bawah ini :

Gambar 5 Tampilan mengkoneksikan *server*

b. Pemilihan *database*
Setelah mengkoneksikan *server*, pilih *database* yang akan disandikan pada kotak pilihan *database*. Adapun tampilan pemilihan *database* dapat dilihat pada gambar di bawah ini :



Gambar 6 Tampilan pemilihan *database*

c. Pemilihan tabel
Setelah pemilihan *database*, pilih tabel yang akan ditampilkan pada kotak pilihan tabel. Adapun tampilan pemilihan tabel dapat dilihat pada gambar di bawah ini :

Gambar 7 Tampilan pemilihan tabel

d. Menampilkan *record*
Setelah penampilan tabel, tampilkan *record* dengan memilih tombol tampilkan *record*. Adapun tampilan menampilkan *record* dapat dilihat pada gambar di bawah.

Gambar 8 Tampilan menampilkan *record*

e. Memasukan kunci
Setelah *record* dari *database* yang dipilih ditampilkan, inputkan kunci yang digunakan proses enkripsi. Adapun tampilan memasukan kunci dapat dilihat pada gambar di bawah.

BULLETIN OF INFORMATION TECHNOLOGY (BIT)
Volume 1, No. 1, Maret 2020, pp 36 - 49
ISSN 2722-0524 (media online)

Gambar.9 Tampilan memasukan kunci

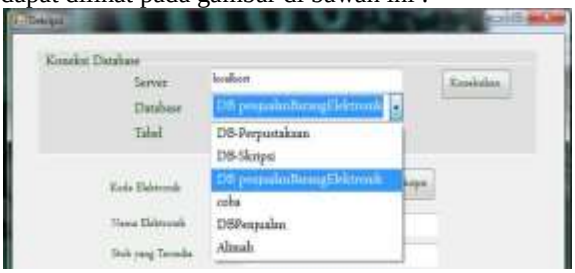
f. Proses enkripsi
Setelah memasukan kunci, lakukan proses enkripsi pada *record* yang ditampilkan dengan kunci yang diinputkan. Adapaun tampilan proses enkripsi dapat dilihat pada gambar di bawah.

Gambar 10 Tampilan proses enkripsi

2. Proses Dekripsi
Proses dekripsi memiliki beberapa tahap yaitu, proses mengkoneksikan *server*, pemilihan *database*, pemilihan tabel, dekripsi per *record* dan proses dekripsi keseluruhan *record*.
a. Mengkoneksikan *server*
Sama seperti proses enkripsi, adapun proses mengkoneksikan *server database* dilakukan dengan menginputkan *server* dan memilih *buton* koneksikan seperti pada gambar di bawah ini :

Gambar 11 Tampilan mengkoneksikan *server*

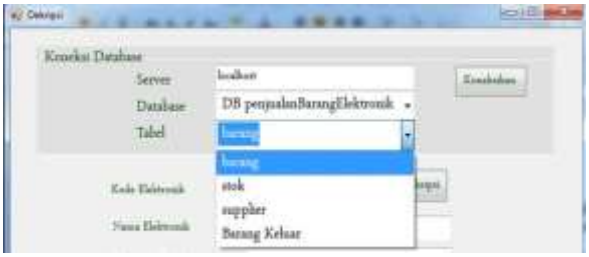
b. Pemilihan *database*
Setelah mengkoneksikan server, pilih *database* yang akan disandikan pada kotak pilihan *database*. Adapun tampilan pemilihan *database* dapat dilihat pada gambar di bawah ini :



Gambar 12 Tampilan pemilihan *database*

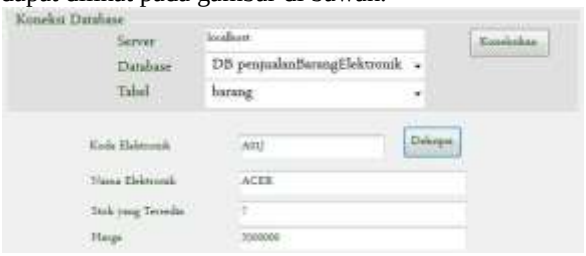
c. Pemilihan tabel
Setelah pemilihan *database*, pilih tabel yang akan ditampilkan pada kotak pilihan tabel. Adapun tampilan pemilihan tabel dapat dilihat pada gambar di bawah ini :

BULLETIN OF INFORMATION TECHNOLOGY (BIT)
Volume 1, No. 1, Maret 2020, pp 37 - 51
ISSN 2722-0524 (media online)



Gambar 13 Tampilan pemilihan tabel

d. Dekripsi per *record*
Setelah terkoneksi dengan *database* dan tabel yang dipilih, inputkan *primary key* dari *record database*. Adapun tampilan dekripsi per *record* dapat dilihat pada gambar di bawah.



Gambar 14 Tampilan pemilihan tabel

e. Proses dekripsi keseluruhan *record*
Proses ini mendekripsikan keseluruhan *record database* yang telah dikoneksikan. Adapun tampilan dekripsi keseluruhan *record* dapat dilihat pada gambar di bawah.

Gambar 15 Tampilan pemilihan dekripsi keseluruhan *record*

3. KESIMPULAN

Berdasarkan hasil penerapan algoritma Mars pada penyandian *record database* yang telah dilakukan, penulis dapat menarik kesimpulan sebagai berikut:

1. Penyandian *record database* diawali dengan penentuan nama *database* dan pemilihan tabel yang memiliki *record* yang akan disandikan, dimana masing-masing *record* pada tabel yang dipilih disandikan berdasarkan algoritma Mars.
2. Penerapan algoritma mars pada *record database* dilakukan dengan mengambil *record* dari *database* yang dipilih dan kemudian dimasukan ke dalam proses enkripsi dan dekripsi algoritma mars.
3. Perancangan aplikasi yang akan dibangun dibuat menngunakan beberapa bagan alir seperti *flowchart* dan UML, kemudian rancangan dibangun menggunakan bahasa pemograman Visual Basic yang mana algoritma mars diterapkan pada salah satu tombol yang berfungsi untk mengenkripsi *record* pada aplikasi yang dibangun.

DAFTAR PUSTAKA

[1] Y. Yanti, Munawir, Zulfan, and Erdiwansyah, "Implementasi Sistem Keamanan Database Menggunakan Metode Triangle Chain," *Serambi Eng.*, vol. II, no. 4, 2017.
[2] J. S. Komputer, N. R. Yanti, and D. A. Ritonga, "Implementasi Algoritma Data Encryption Standard Pada Penyandian Record Database," no. 1, 2018.
[3] T. Zebua, "TRIANGLE CHAIN PADA PENYANDIAN RECORD DATABASE ...," *Pelita Inform. Budi Darma*, vol. 3, no. April, 2013.
[4] M. Natsir, "Pengembangan Prototype Sistem Kriptografi Untuk Enkripsi Dan Dekripsi Data Office Menggunakan

Metode Blowfish Dengan Bahasa Pemrograman Java," vol. 6, 2017.
[5] D. Sartika, "PENGEMBANGAN PERANGKAT LUNAK PENYEMBUNYIAN PESAN," *J. Ilm. Inform. Glob.*, vol. 7, no. 1, pp. 1–6, 2016.
[6] Suprianto, "Pengantar Ilmu Kriptografi Teori Analisis Dan Implementasi," *J. Comput. Bisnis*, vol. 1, no. 2, pp. 105–118, 2007.
[7] M. Rinaldi, *Kriptografi*. Bandung: Informatika Bandung, 2006.
[8] A. Dony, *Pengantar Ilmu Kriptografi Teori Analisa dan Implementasi*. Yogyakarta: ANDI, 2008.
[9] D. Sartika, "Pengembangan Perangkat Lunak Penyembunyian Pesan Terenkripsi Menggunakan Algoritma Mars Pada Citra Digital Dengan Metode Adaptif," vol. 7, no. 1, pp. 1–6, 2016.
[10] T. Zebua, "TRIANGLE CHAIN PADA PENYANDIAN RECORD DATABASE ...," *Pelita Inform. Budi Darma*, vol. 111, no. April, pp. 37–49, 2013.