

Modifikasi Vigenere Cipher Dengan Pembangkit Kunci Blum Blum Shub

Firman Telaumbanua^{1*}, Taronisokhi Zebua¹

¹ Fakultas Ilmu Komputer dan Teknologi Informasi, Program Studi Teknik Informatika, Universitas Budi Darma, Medan, Indonesia
Email: ^{1*}firmantelaumbanua96@gmail.com, ²taronizeb@gmail.com

^{*}Email Penulis Korespondensi

Abstrak–Vigenere cipher merupakan algoritma kunci simetri yang menggunakan substitusi abjad dengan menggunakan huruf tersebut sebagai plaintext dan huruf kunci yang memiliki posisi sebanding. Seiring dengan perkembangan ilmu pengetahuan manusia, kelemahan dari vigenere cipher berhasil ditemukan.. Salah satu cara yang dapat dilakukan untuk mengatasi kelemahan vigenere cipher di atas adalah dengan melakukan pembangkitan kunci yang lebih acak. Penelitian ini menguraikan bagaimana prosedur yang dilakukan untuk memodifikasi pembangkitan kunci yang digunakan pada algoritma vigenere cipher. Proses pembangkitan kunci dilakukan berdasarkan pembangkit kunci blum blum shub, artinya kunci yang digunakan pada proses enkripsi dan dekripsi adalah kunci yang dibangkitkan berdasarkan pembangkit kunci blum blum shub, sehingga proses modifikasi yang dilakukan dalam pembangkitan kunci tersebut dapat meminimalkan tindakan pemecahan kunci yang dilakukan pihak lain serta algoritma ini dapat lebih optimal dalam mengamankan data. Hasil dari penelitian ini adalah merancang sebuah aplikasi pengamanan data berbasis android dengan menggunakan metode vigenere cipher yang telah dimodifikasi menggunakan pembangkit kunci blum blum shub. Aplikasi ini dapat digunakan untuk mengamankan data berupa teks, sehingga tidak dapat diambil oleh orang lain. Selain itu, aplikasi yang akan dirancang ini lebih mudah untuk digunakan dalam pengamanan data.

Kata Kunci: Keamanan, Kunci, Algoritma, Kriptografi, Blum Blum Shub

Abstract–Vigenere cipher is a symmetric key algorithm that uses alphabetic substitution by using these letters as plaintext and key letters that have comparable positions. Along with the development of human science, the weaknesses of the vigenere cipher were discovered. One way that can be done to overcome the weaknesses of the vigenere cipher above is to generate a more random key. This study describes how the procedure is carried out to modify the key generation used in the vigenere cipher algorithm. The key generation process is carried out based on the blum blum shub key generator, meaning that the key used in the encryption and decryption process is the key generated based on the blum blum shub key generator, so that the modification process carried out in generating the key can minimize key-breaking actions by other parties as well as This algorithm can be more optimal in securing data. The result of this research is to design an android-based data security application using the modified vigenere cipher method using the blum blum shub key generator. This application can be used to secure text data, so that it cannot be retrieved by other people. In addition, the application to be designed is easier to use in data security.

Keywords: Security, Keys, Algorithm, Cryptography, Blum Blum Shub

1. PENDAHULUAN

Penyalahgunaan informasi merupakan salah satu dampak negatif dari perkembangan teknologi yang berkembang pesat saat ini, oleh karena itu perlu dilakukan pengamanan informasi agar tidak dapat diakses oleh orang-orang yang tidak bertanggung jawab. Informasi-informasi tersebut dapat berupa data pribadi yang tidak dibuat untuk dipublikasikan, data perusahaan penting dan berbagai informasi lain yang bersifat rahasia. Salah satu cara yang dapat digunakan untuk mengamankan informasi tersebut adalah dengan menggunakan metode kriptografi. Kriptografi merupakan salah satu ilmu yang berperan penting dalam bidang pengamanan informasi. Kriptografi memiliki teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi misalnya kerahasiaan dan integritas data, serta otentikasi. Kuat lemahnya metode kriptografi tidak terletak dari hasil *enkripsi* atau *ciphertext*, melainkan terletak pada kunci yang digunakan, oleh sebab itu kunci merupakan jantung dari pertahanan data tersebut agar tidak dapat diakses atau dibobol oleh orang-orang yang tidak bertanggung jawab[1]. Salah satu yang termasuk dalam algoritma kunci simetri adalah *vigenere cipher*[1][2][3].

Vigenere cipher merupakan algoritma kunci simetri yang menggunakan substitusi abjad dengan menggunakan huruf tersebut sebagai *plaintext* dan huruf kunci yang memiliki posisi sebanding. Namun seiring dengan perkembangan ilmu pengetahuan manusia, kelemahan dari *vigenere cipher* berhasil ditemukan. Salah satu kelemahan dari *vigenere cipher* ini adalah kuncinya yang berulang sehingga dapat ditebak dengan tepat[4][5]. Salah satu cara yang dapat dilakukan untuk mengatasi kelemahan *vigenere cipher* di atas adalah dengan melakukan pembangkitan kunci yang lebih acak. Kunci yang dibangkitkan secara acak tahan terhadap serangan analisa frekuensi tinggi[4][6]. Salah satu algoritma yang dapat digunakan untuk membangkitkan kunci nilai-nilai adalah pembangkit kunci *blum blum shub*.

Pembangkit kunci *blum blum shub* merupakan salah satu pembangkit kunci yang cara kerjanya sederhana, namun paling bagus dan kompleks dalam mengamankan data. Pembangkit kunci ini aman dalam mengamankan data, karena pemecahan pembangkit kunci ini setara dengan memecahkan metode *quadratic residue problem* yang pada gilirannya akan memecahkan *number pseudocode* lengkap yang merupakan basis kriptografi dan menjadikan pembangkit kunci *blum blum shub* ini menjadi salah satu jenis pembangkit kunci yang paling disukai khususnya dalam proses pembangkitan kunci, karena kunci yang dihasilkan susah ditebak[7][8][9]. Kunci yang digunakan pada proses enkripsi dan dekripsi pada *vigenere cipher* setelah dimodifikasi adalah kunci yang dibangkitkan berdasarkan pembangkit kunci *blum blum shub*, sehingga proses modifikasi yang dilakukan dalam pembangkitan kunci tersebut dapat meminimalkan tindakan pemecahan kunci yang dilakukan pihak lain serta algoritma ini dapat lebih optimal dalam mengamankan data.

2. METODE PENELITIAN

2.1 Kriptografi

Kriptografi (*cryptography*) berasal dari bahasa Yunani yaitu “*cryptos*” artinya “*secret*” yang berarti rahasia, sedangkan “*graphein*” artinya “*writing*” yang berarti tulisan rahasia [10][2]. Kriptografi merupakan salah satu bidang ilmu yang mempelajari tentang menjaga keamanan pesan dalam proses pengiriman dengan menggunakan metode penyandian tertentu dengan tujuan agar informasi yang termuat dalam pesan tersebut tidak diambil dan disalahgunakan oleh orang lain [10][5].

2.2 Algoritma Vigenere Cipher

Vigenere cipher merupakan metode yang menggunakan table *vigenere* untuk mengenskripsi sebuah teks. Metode ini melakukan penyandian dengan tabel diagram dengan huruf alfabet yang terurut secara diagonal [5][4][11]. Dalam melakukan proses enkripsi, harus menggunakan rumus yang memang berlaku di dalam *vigenere cipher*. Adapun rumus matematis penggunaan *vigenere cipher* [5][4][12], yaitu:

- a. Algoritma enkripsi *vigenere cipher* :
 - $C_i = (P_i + K_i) \bmod 26$ (1)
 - atau
 - $C_i = (P_i + K_i) \bmod 256$ (2)
- b. Algoritma dekripsi *vigenere cipher* :
 - $P_i = (C_i - K_i) \bmod 26$ (3)
 - atau
 - $P_i = (C_i - K_i) \bmod 256$ (4)

2.3 Pembangkit Kunci Blum Blum Shub

Pembangkit Kunci *blum blum shub* merupakan pembangkit kunci bilangan acak yang sangat sederhana dan paling bagus secara kompleksitas teoritis [13][14][8]. Pembangkit kunci *blum blum shub* (BBS) dibuat pada tahun 1986 oleh Lenore Blum, Manuel Blum, Michael Shub yang dirancang dengan dasar teori bilangan [7]. BBS memiliki bentuk persamaan [13][14][7], yaitu :

$$X_{i+1} = X_i^2 \bmod m \text{ (5)}$$

3. HASIL DAN PEMBAHASAN

3.1 Analisa Masalah

Adapun contoh modifikasi *vigenere cipher* menggunakan pembangkit kunci *blum blum shub* seperti di bawah ini. Dimisalkan *plaintext* yang digunakan adalah **Virus Corona**

- a. Konversi setiap karakter *plaintext* ke dalam bilangan desimal
 Tiap huruf pada *plaintext* akan diubah menjadi sesuai urutannya pada tabel ASCII sehingga didapat seperti tabel di bawah ini :

Tabel 1. Konversi *Plaintext* Ke Desimal ASCII

| Huruf | Desimal ASCII |
|-------|---------------|
| V | 86 |
| i | 105 |
| r | 114 |
| u | 117 |
| s | 115 |
| Spasi | 32 |
| C | 67 |
| o | 111 |
| r | 114 |
| o | 111 |
| n | 110 |
| a | 97 |

- b. Proses pembangkitan kunci menggunakan pembangkit kunci *blum blum shub*
 1. Pilihlah dua buah bilangan prima rahasia, yaitu p dan q yang masing-masing kongruen dengan 3 mod 4 seperti berikut ini :
 $p \equiv 3 \bmod 4$ maka $p \bmod 4 = 3$, sehingga diambil $p = 59$
 $q \equiv 3 \bmod 4$ maka $q \bmod 4 = 3$, sehingga diambil $q = 71$
 2. Mencari nilai n dengan mengalikan p dan q
 $n = p \times q$

- $n = 59 \times 71$
 $n = 4189$
3. Pilih bilangan bulat lain yaitu s sebagai perangkat
 $2 \leq s < n$
 s dan n relatif prima
 Relatif prima merupakan dua buah bilangan yang memiliki faktor prima yaitu 1
 $n = 4189$ dengan FPB = 1, 59, 71
 $s = 21$ dengan FPB = 1, 3, 7
 didapat bahwa n dan s relatif prima dan kemudian hitung x_0 :
 $x_0 = s^2 \bmod n = 21^2 \bmod 4189 = 441 \bmod 4189 = 441$
4. Mencari LSB untuk membentuk karakter kunci
 Syarat untuk mendapatkan nilai LSB adalah jika hasil mod x_i bernilai ganjil, maka nilai LSB yang didapat adalah 1. Jika nilai mod x_i bernilai genap, maka nilai LSB yang didapat adalah 0. Hasil nilai mod x_1 adalah 441 dan merupakan bilangan ganjil, maka nilai LSB yang didapat adalah 1. Proses untuk mendapatkan hasil nilai mod x_2 berpatokan pada hasil x_1 atau hasil x_i sebelumnya. Demikian juga untuk mendapatkan nilai mod x_i selanjutnya selalu berpatokan pada nilai mod x_i sebelumnya.
 Proses pembentukan karakter kunci untuk karakter *plaintext* ke 1 :
 $x_1 = x_0^2 \bmod n = 441^2 \bmod 4189 = 441 \bmod 4189 = 441$; LSB = 1
 $x_2 = x_1^2 \bmod n = 441^2 \bmod 4189 = 194481 \bmod 4189 = 1787$; LSB=1
 $x_3 = x_2^2 \bmod n = 1787^2 \bmod 4189 = 3193369 \bmod 4189 = 1351$; LSB=1
 $x_4 = x_3^2 \bmod n = 1351^2 \bmod 4189 = 1825201 \bmod 4189 = 2986$; LSB=0
 $x_5 = x_4^2 \bmod n = 2986^2 \bmod 4189 = 8916196 \bmod 4189 = 2004$; LSB=0
 $x_6 = x_5^2 \bmod n = 2004^2 \bmod 4189 = 4016016 \bmod 4189 = 2954$; LSB=1
 $x_7 = x_6^2 \bmod n = 2954^2 \bmod 4189 = 8726116 \bmod 4189 = 429$; LSB=1
 Untuk membentuk 1 karakter diperlukan 8 bit bilangan biner. Jadi pesan di atas berjumlah 12 karakter, maka lakukan proses pencarian nilai x hingga 96. Berikut adalah kunci dari hasil pembangkitan kunci menggunakan pembangkit kunci *blum blum shub* :

Tabel 2. Kunci Baru

| Kelompok Biner Kunci | Hasil Gabungan LSB | Desimal | Karakter |
|--------------------------|--------------------|---------|----------|
| 1 ($x_1 - x_8$) | 11100110 | 230 | Æ |
| 2 ($x_9 - x_{16}$) | 10000100 | 132 | È |
| 3 ($x_{17} - x_{24}$) | 10111001 | 185 | ı |
| 4 ($x_{25} - x_{32}$) | 11101010 | 234 | Ê |
| 5 ($x_{33} - x_{40}$) | 01010010 | 82 | R |
| 6 ($x_{41} - x_{48}$) | 01000000 | 64 | @ |
| 7 ($x_{49} - x_{56}$) | 00111010 | 58 | : |
| 8 ($x_{57} - x_{64}$) | 11000111 | 199 | Ç |
| 9 ($x_{65} - x_{72}$) | 01101011 | 107 | K |
| 10 ($x_{73} - x_{80}$) | 11110111 | 247 | ÷ |
| 11 ($x_{81} - x_{88}$) | 01001110 | 78 | N |
| 12 ($x_{89} - x_{96}$) | 00101000 | 40 | (|

c. Proses Enkripsi

Proses enkripsi yang dilakukan berdasarkan modifikasi *vigenere cipher* yang menggunakan pembangkit kunci *blum blum shub* diselesaikan dalam bentuk perhitungan ASCII, yang dimana enkripsi dapat dinyatakan sebagai penjumlahan modulo 256 dari satu karakter *plaintext* dengan satu karakter kunci hasil pembangkitan BBS. Berikut adalah proses enkripsi yang menggunakan persamaan 2 yang ada pada bab sebelumnya

$$C_1 = (V + K_1) \bmod 256 = (86 + 230) \bmod 256 = 316 \bmod 256 = 60 = <$$

$$C_2 = (i + K_2) \bmod 256 = (105 + 132) \bmod 256 = 237 \bmod 256 = 237 = í$$

$$C_3 = (r + K_3) \bmod 256 = (114 + 185) \bmod 256 = 299 \bmod 256 = 43 = +$$

$$C_4 = (u + K_4) \bmod 256 = (117 + 234) \bmod 256 = 351 \bmod 256 = 95 = _$$

$$C_5 = (s + K_5) \bmod 256 = (115 + 82) \bmod 256 = 197 \bmod 256 = 197 = Å$$

$$C_6 = (\text{spasi} + K_6) \bmod 256 = (32 + 64) \bmod 256 = 96 \bmod 256 = 96 = ‘$$

$$C_7 = (C + K_7) \bmod 256 = (67 + 58) \bmod 256 = 125 \bmod 256 = 125 = }$$

$$C_8 = (o + K_8) \bmod 256 = (111 + 199) \bmod 256 = 310 \bmod 256 = 54 = 6$$

$$C_9 = (r + K_9) \bmod 256 = (114 + 107) \bmod 256 = 221 \bmod 256 = 221 = Ý$$

$$C_{10} = (o + K_{10}) \bmod 256 = (111 + 247) \bmod 256 = 358 \bmod 256 = 102 = f$$

$$C_{11} = (n + K_{11}) \bmod 256 = (110 + 78) \bmod 256 = 188 \bmod 256 = 188 = ¼$$

$$C_{12} = (a + K_{12}) \bmod 256 = (97 + 40) \bmod 256 = 137 \bmod 256 = 137 = \%o$$

Hasil enkripsi di atas akan menjadi *ciphertext*. *Ciphertext* yang didapat dari hasil perhitungan di atas adalah < í+ _Å}6Ýf¼%o

d. Proses dekripsi

Setelah *ciphertext* didapat dari proses enkripsi, maka proses dekripsi untuk mendapatkan *plaintext* hampir sama dengan proses enkripsi. Kunci yang digunakan sama dengan kunci yang dihasilkan dari proses pembangkit kunci *blum blum shub* yang digunakan pada proses enkripsi. Adapun proses dekripsi modifikasi *vigenere cipher* dengan pembangkit kunci *blum blum shub* yang menggunakan persamaan 4 yang telah dibahas pada bab sebelumnya, yaitu

$$\begin{aligned}
 P_1 &= (C - K_1) \bmod 256 = (60 - 230) \bmod 256 = -170 \bmod 256 = 86 = V \\
 P_2 &= (i - K_2) \bmod 256 = (237 - 132) \bmod 256 = 105 \bmod 256 = 105 = i \\
 P_3 &= (+ - K_1) \bmod 256 = (43 - 185) \bmod 256 = -142 \bmod 256 = 114 = r \\
 P_4 &= (_ - K_1) \bmod 256 = (95 - 234) \bmod 256 = -139 \bmod 256 = 117 = u \\
 P_5 &= (\grave{A} - K_1) \bmod 256 = (197 - 82) \bmod 256 = 115 \bmod 256 = 115 = s \\
 P_6 &= (' - K_1) \bmod 256 = (96 - 64) \bmod 256 = 32 \bmod 256 = 86 = \text{spasi} \\
 P_7 &= (} - K_1) \bmod 256 = (125 - 58) \bmod 256 = 67 \bmod 256 = 67 = C \\
 P_8 &= (6 - K_1) \bmod 256 = (54 - 199) \bmod 256 = -145 \bmod 256 = 111 = o \\
 P_9 &= (\acute{Y} - K_1) \bmod 256 = (221 - 107) \bmod 256 = 114 \bmod 256 = 114 = r \\
 P_{10} &= (f - K_1) \bmod 256 = (102 - 247) \bmod 256 = -145 \bmod 256 = 111 = o \\
 P_{11} &= (\frac{1}{4} - K_1) \bmod 256 = (188 - 78) \bmod 256 = 110 \bmod 256 = 110 = n \\
 P_{12} &= (\% - K_1) \bmod 256 = (137 - 40) \bmod 256 = 97 \bmod 256 = 97 = a
 \end{aligned}$$

Maka *plaintext* dari proses dekripsi di atas adalah **Virus Corona**.

3.2 Implementasi

Berikut adalah hasil program dari modifikasi *vignere cipher* dengan pembangkit kunci *blum blum shub* :



Gambar 1. Tampilan Program



Gambar 2. Halaman Modifikasi *Vigenere Cipher*

Tabel 3. Hasil Pengujian Program

| Vigenere Cipher | | Modifikasi Vigenere Cipher | | | | | |
|---------------------------|--------|----------------------------|-------|----|----|---------------------------------------|----------------------------|
| Plaintext | Kunci | Ciphertext | Nilai | | | Kunci | Ciphertext |
| | | | p | q | s | | |
| VIRUS CORON A | VAKSIN | QIBMAPJR YFI | 59 | 71 | 21 | ÆÈ¹ÊR@: ÇK÷N(| í+ }6Ýf ¼% .©;žol |
| TEKNIK INFORM ATIKA | PRODI | IVUQQZSB JWGDOXQ ZR | 61 | 67 | 19 | cD60ACK SOHE'CA N&''GâFF àç | ©†š''ÖCèâJ (|
| INDONE SIA | RAKYAT | ZNOMNXT IKKEKUEU Y | 61 | 67 | 19 | cD60ACK SOHE'CA | -²šÝtfÓÉÝ Fëž'uFqçA |

| Vigenere Cipher | | | Modifikasi Vigenere Cipher | | | | |
|---|----------|--------------------|----------------------------|----|----|-------------------------------------|----------------------|
| Plaintext | Kunci | Ciphertext | Nilai | | | Kunci | Ciphertext |
| | | | p | q | s | | |
| MERDE KA BELAJA R PROGR AM | APLIKASI | BTWITAJX RDAZKM | 11 1 | 79 | 37 | N&"GâFF à 7sw.sw7s; 7sw7; | yθãCANzÙ ©“ ,ãBş” |

4. KESIMPULAN

Hasil dari modifikasi *vigenere cipher* dengan pembangkit kunci *blum blum shub* menghasilkan kunci yang lebih optimal dibandingkan dengan kunci pada *vigenere cipher* yang belum dimodifikasi. Hal ini dibuktikan dari setiap pengujian yang dilakukan menghasilkan kunci yang lebih kuat dan optimal sehingga *plaintext* dan kuncinya sulit untuk ditebak.

REFERENCES

- [1] R. Nauri and N. Ratama, "Implementasi Algoritma Kriptografi AES (Advanced Encryption Standard) 128 Bit Untuk Pengamanan Dokumen Shipping," *Journal Of Artificial Intelligence And Innovative Applications*, pp. 37-44, 2 Mei 2020.
- [2] T. Rahmat Aditia, Y. Permasari and H. Erwin, "Kriptografi Advanced Encryption Standard (AES) Untuk Penyandian File Dokumen," *Jurnal Matematika UNISBA*, vol. 15, pp. 1-14, 1 Mei 2016.
- [3] A. A. Permana and D. Nurnaningsih, "Rancangan Aplikasi Pengamanan Data Dengan Algoritma Advanced Encryption Standard (AES)," *Jurnal Teknik Informatika*, vol. 11, pp. 177-186, 2018.
- [4] Y. A. Mulyadi, P. E. N and R. R. J.P, "implementasi Algoritma AES 128 dan SHA – 256 Dalam Pengkodean padaSebagian Frame Video CCTV MPEG-2," *urnal Teori dan Aplikasi Ilmu Komputer*, vol. 1, pp. 33-39, 2018.
- [5] R. R. T, Y. Permasari and H. Erwin, "Kriptografi Advanced Encryption Standard (AES) Kriptografi Advanced Encryption Standard (AES)," *Kriptografi Advanced Encryption Standard (AES)*, vol. 15, pp. 7-13, 2016.
- [6] M. Prabowo, I. Muhimmah and R. Kurniawan, "Pemodelan Pengiriman Data Citra Medis untuk," *Seminar Nasional Informatika Medis (SNIMed)*, vol. VIII, p. 76, 2017.
- [7] Ardiana, "Perancangan Sistem Informasi Radiologi Guna Mendukung Peningkatan Pelayanan Pada Pasien di Rumah Sakit Umum Daerah Al-Ihsan Pemprov Jabar," *Teras Kesehatan*, vol. I, pp. 2622-2396, 2019.
- [8] Ansori, "Pengertian Flowchart : Jenis, Simbol, dan Contohnya," 27 Maret 2020. [Online]. Available: <https://www.ansoriweb.com/2020/03/pengertian-flowchart.html>.
- [9] R. A. Nugraha and G. Pramukasari, "Sistem Informasi Akademik Sekolah Berbasis Web Di Sekolah Menengah Pertama Negeri 11 Tasikmalaya," *JUMIKA*, vol. IV, pp. 51-60, 2017.
- [10] Y. Yudhanto and A. H. Prasetyo, *Panduan Mudah Belajar Framework Laravel*, Jakarta: PT. Alex Media Komputindo, 2018, pp. 1-17.
- [11] A. Muahrdian, "Text Editor Visual Studio Code di Linux," 28 Juli 2017. [Online]. Available: <https://www.petanikode.com/text-editor-vscode/>.
- [12] B. Winarso, "Apa Itu Google Chrome dan Sepenggal Sejarahnya," 23 Maret 2016. [Online]. Available: <https://dailysocial.id/post/apa-itu-google-chrome/>.
- [13] I. M. Perkasa and B. E. Setiawan, "Pembangunan Web Service Data Masyarakat Menggunakan REST API dengan Access Token," *ULTIMA Computing*, vol. X, pp. 19-26, 2018.