PENGAMANAN TEKS MENGGUNAKAN ALGORITMA TRANSPOSISI DAN MODIFIKASI SANDI MORSE

(*Text Security Using Transposition Algorithm and Morse Code Modification*)

Rizki Gangsar Septiono [1)], Kiswara Agung Santoso [2)], Ahmad Kamsyakawuni [3)]

[1,2,3)] Jurusan Matematika, Fakultas MIPA, Universitas Jember
Jl. Kalimantan 37 Jember 68121, Jember
e-mail: gangsaralahyo29@gmail.com, {kiswara, kamsyakawuni}.fmipa@unej.ac.id

Abstract. Humans as social beings communicate to exchange information. Information has many forms including text data, sound, images, and so on. Not all information is freely accessible. Information security is needed to prevent information from being misused by irresponsible parties. Cryptography studies how to convert information into a form that is unknown except for the sender and recipient of the information. Cryptography is divided into two, classical cryptography and modern cryptography. One form of cryptography that can be found around us is the use of Morse code in scouting. Classical cryptography and modern cryptography can be combined to increase the security of the algorithms used in hiding information. In this article, the transposition algorithm is applied in binary digits. Keys in the form of ASCII characters are converted into binary using modified Morse code and operated with plaintext. The results showed that the combination of transposition algorithm and Morse code modification adds complexity so that the proposed algorithm is difficult for cryptanalysts to solve.

Keywords: Cryptography, Morse Code, Transposition Algorithm.

1. Pendahuluan

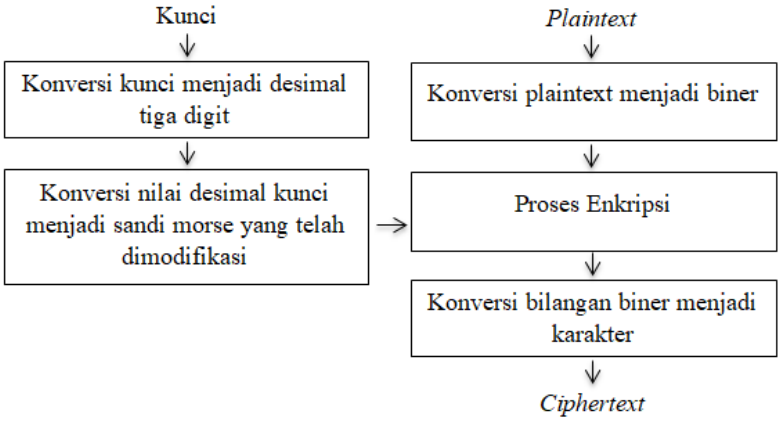
Perkembangan teknologi yang semakin pesat membuat arus informasi semakin dinamis. Penyimpanan informasi saat ini didominasi oleh penyimpanan digital menggunakan komputer berupa data. Keamanan dan kerahasian menjadi aspek penting dalam pertukaran informasi [2]. Tidak semua data bersifat publik sehingga perlu adanya sistem pengamanan yang menjamin bahwa data yang ada tidak diretas atau diubah isinya oleh pihak tertentu, yaitu kriptografi. Kriptografi merupakan studi terkait metode komunikasi yang aman antara dua pihak yang bertujuan untuk menghindari pihak ketiga memahami pesan-pesan yang dikirimkan agar tidak disalah gunakan oleh pihak tidak bertanggung jawab [11]. Kriptografi sudah dikenal jauh pada masa Mesir Kuno, namun mulai berkembang sejak masa Kekaisaran Romawi dibawah pemerintahan Kaisar Julius Caesar yang sekarang kita kenal sebagai Caesar Cipher [10]. Kriptografi dibagi menjadi dua yaitu kriptografi klasik dan kriptografi modern. Kriptografi klasik merupakan algoritma berbasis pengacakan karakter dengan aturan tertentu [1]. Kriptografi modern beroperasi dalam mode bit yang dinyatakan dalam rangkaian string ataupun bit biner 0 dan 1 [13]. Bit biner digunakan pada komputer yang diwakili dengan dua keadaan (*two-state element*) yaitu saat tidak ada arus (keadaan *off*) atau saat ada arus (keadaan *on*) [6].

Kunci berperan penting untuk meningkatkan keamanan algoritma kriptografi [3]. Kunci algoritma kriptografi dibagi menjadi dua, yaitu kunci simetris dan kunci asimetris. Kunci simetris berarti kunci yang digunakan pada proses enkripsi dan dekripsi sama. Sedangkan pada kunci asimetris, terdapat kunci publik yang digunakan pada proses enkripsi dan kunci privat yang digunakan pada proses dekripsi [7]. Nasution [9] menggabungkan antara vigenere cipher dan transposisi cipher untuk mengamankan data teks. Data teks yang dienkripsi dapat dikembalikan seperti semula melalui proses dekripsi, namun data yang terbaca hanya karakter A sampai Z. Karakter diluar A sampai Z akan otomatis dihapus. Standar internasional karakter huruf dan simbol yang digunakan adalah *American Standard Code for Information Interchange* (ASCII). Penelitian yang dilakukan oleh Wiradarma dkk [15] menggabungkan vigenere cipher dan metode Electronic Code Book (ECB) yang merupakan algoritma kriptografi klasik dan modern. Kesimpulan yang didapat adalah penggabungan kedua algoritma tersebut dapat memperkuat tingkat keamanan karena kompleksitasnya lebih rumit. Dalam bit biner terdapat operasi biner yang sering digunakan yaitu Exclusive Or (XOR). XOR menggunakan logika negasi biimplikasi yang dilakukan pada setiap digit biner informasi yang telah dikonversi kedalam biner dengan digit biner lain [8]. Tamba [14] dalam penelitiannya mengambil kesimpulan bahwa algoritma transposisi cipher akan lebih kuat jika dilakukan proses XOR dengan kunci.

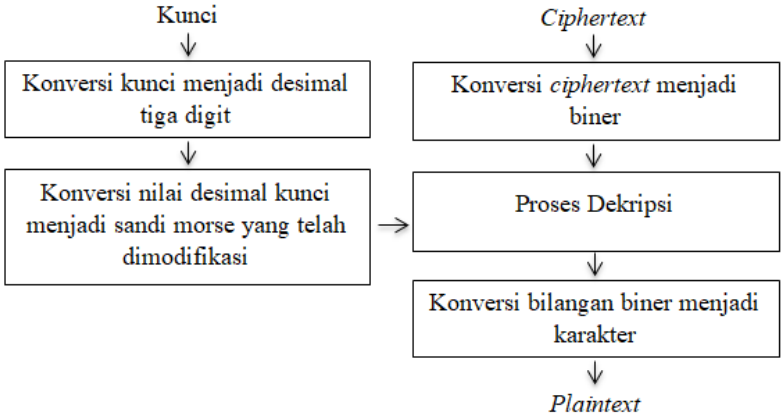
Kriptografi dalam bentuk yang sederhana dapat kita jumpai dalam kehidupan sehari-hari, salah satunya pada materi sandi yang ada pada pramuka. Pramuka mempelajari sandi sebagai sarana pertukaran informasi. Salah satu sandi yang dipelajari dalam pramuka adalah sandi morse [16]. Sandi morse merupakan sistem representasi huruf, angka, tanda baca, dan sinyal menggunakan kode titik dan garis. Sandi morse didasarkan pada kode morse yang dibuat oleh Samuel F. B. Morse dan Alfred Vail pada tahun 1835 [4]. Damayanti [5] menggunakan sandi morse untuk memperkuat tingkat keamanan algoritma Caesar Cipher. Kesimpulan yang didapatkan dari penelitian tersebut adalah modifikasi Caesar Cipher ke dalam bentuk sandi morse sulit dipecahkan oleh pihak yang tidak berkepentingan.

2. Metodologi

Data penelitian yang digunakan adalah pesan teks sebagai plaintext dan kunci. Karakter yang digunakan adalah karakter selain kode ASCII Control Character (Character code 0-31). Proses pengkodean dibagi menjadi enkripsi dan dekripsi. Enkripsi merupakan proses penyandian plaintext menjadi ciphertext, sedangkan Dekripsi merupakan proses mengembalikan ciphertext menjadi plaintext [12]. Skema enkripsi dapat dilihat pada Gambar 1 dan skema deskripsi dapat dilihat pada Gambar 2.



Gambar 1. Skema enkripsi



Gambar 2. Skema dekripsi

Tabel 1. Modifikasi sandi morse

Karakter	Sandi Morse	Konversi menjadi biner
1	●▬▬▬▬	01111
2	●●▬▬▬	00111
3	●●●▬▬	00011
4	●●●●▬	00001
5	●●●●●	00000
6	▬●●●●	10000
7	▬▬●●●	11000
8	▬▬▬●●	11100
9	▬▬▬▬●	11110
0	▬▬▬▬▬	11111

Modifikasi sandi morse pada penelitian ini digunakan untuk memberikan variasi konversi

nilai desimal karakter menjadi biner. Sandi morse yang digunakan adalah sandi morse berupa angka yang memiliki jumlah tetap untuk total titik dan strip yang digunakan. Modifikasi sandi morse dapat dilihat pada Tabel 1.

3. Hasil dan Pembahasan / Results

Pesan yang akan dikodekan adalah kalimat "Aku" dengan kunci "Oke".

3.1 Tahap Enkripsi

a. Konversi *plaintext* menjadi biner
Plaintext yang berupa karakter dikonversi menjadi bilangan biner 8 digit.
Plaintext = Aku
Plaintext (biner) = 010000010110101101110101

b. Konversi kunci menjadi desimal tiga digit
Kunci dibatasi sebanyak 8 karakter. Apabila kunci yang dimasukkan kurang dari 8, maka akan dilakukan perulangan pada karakter yang ada pada kunci sehingga menghasilkan kunci baru dengan panjang 8 karakter. Jika kunci yang dimasukkan lebih dari 8 karakter, maka program tidak dapat berjalan. Kunci dikonversi menjadi bilangan desimal tiga digit. Untuk bilangan desimal satu digit dan dua digit masing-masing ditambah dua digit nol dan satu digit nol sebelum digit desimal.

Kunci = Oke
Kunci Baru = OkeOkeOk
Kunci (desimal) = 079 107 101 079 107 101 079 107

c. Konversi nilai desimal kunci menjadi sandi morse yang telah dikonversi menjadi biner
Kunci yang berbentuk desimal tiga digit diubah menjadi sandi morse yang telah dikonversi menjadi biner. Kunci biner (misalkan Kb) berukuran 1×120 dipartisi menjadi 15 bagian dimana setiap bagian berukuran 1×8.

Kunci (desimal) = 079 107 101 079 107 101 079 107
Kunci (sandi morse) = 111111100011110011111111111000011111111011111
111111000111100111111111110000111111111011111
11111000111100111111111111000

$K(1) = Kb(1), \dots, Kb(8) = 1\ 1\ 1\ 1\ 1\ 1\ 1\ 0$
$K(2) = Kb(9), \dots, Kb(16) = 0\ 0\ 1\ 1\ 1\ 1\ 0\ 0$
$K(3) = Kb(17), \dots, Kb(24) = 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1$
$K(4) = Kb(25), \dots, Kb(32) = 1\ 1\ 1\ 0\ 0\ 0\ 0\ 1$
$K(5) = Kb(33), \dots, Kb(40) = 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1$
$K(6) = Kb(41), \dots, Kb(48) = 0\ 1\ 1\ 1\ 1\ 1\ 1\ 1$
$K(7) = Kb(49), \dots, Kb(56) = 1\ 1\ 1\ 1\ 0\ 0\ 0\ 1$
$K(8) = Kb(57), \dots, Kb(64) = 1\ 1\ 1\ 0\ 0\ 1\ 1\ 1$

$$
\begin{array}{rcrcl}
K(9) & = & Kb(65),\ldots,Kb(72) & = & 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1 \\
K(10) & = & Kb(73),\ldots,Kb(80) & = & 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1 \\
K(11) & = & Kb(81),\ldots,Kb(88) & = & 1\ 1\ 1\ 1\ 1\ 0\ 1\ 1 \\
K(12) & = & Kb(89),\ldots,Kb(96) & = & 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1 \\
K(13) & = & Kb(97),\ldots,Kb(104) & = & 1\ 0\ 0\ 0\ 1\ 1\ 1\ 1 \\
K(14) & = & Kb(105),\ldots,Kb(112) & = & 0\ 0\ 1\ 1\ 1\ 1\ 1\ 1 \\
K(15) & = & Kb(113),\ldots,Kb(120) & = & 1\ 1\ 1\ 1\ 1\ 0\ 0\ 0
\end{array}
$$

d. Proses enkripsi

1) Misalkan panjang pesan adalah n. *Plaintext* biner disusun menjadi matriks (misalkan matriks P) dengan ukuran $n \times 8$.
2) Ambil kunci ke-i. Simbol (i) merupakan "perulangan ke-" yang banyak perulangannya telah ditentukan (15 kali perulangan).
3) Transposisi dilakukan pada *plaintext* dan kunci. Transposisi yang dilakukan dapat dilihat pada Tabel 2. Hasil transposisi *plaintext* dimisalkan sebagai Pt dan hasil transposisi kunci dimisalkan sebagai Kt. Simbol (j) pada tabel merupakan "perulangan ke-" dengan banyak perulangan sesuai dengan panjang *plaintext*.

Tabel 2. Tabel transposisi enkripsi

Transposisi *plaintext*		**Transposisi kunci**	
$Pt(1)$	$P(j,1)$	$Kt(1)$	$K(i)(2)$
$Pt(2)$	$P(j,2)$	$Kt(2)$	$K(i)(5)$
$Pt(3)$	$P(j,5)$	$Kt(3)$	$K(i)(6)$
$Pt(4)$	$P(j,8)$	$Kt(4)$	$K(i)(3)$
$Pt(5)$	$P(j,6)$	$Kt(5)$	$K(i)(1)$
$Pt(6)$	$P(j,3)$	$Kt(6)$	$K(i)(4)$
$Pt(7)$	$P(j,4)$	$Kt(7)$	$K(i)(7)$
$Pt(8)$	$P(j,7)$	$Kt(8)$	$K(i)(8)$

4) Dilakukan operasi *Exclusive Or* (XOR) antara Pt dan Kt. Hasil XOR dimisalkan sebagai $C(j)$.
5) Dilakukan *Rotate Left Shift* (RLS) terhadap $C(j)$ sebanyak 3 kali untuk mendapatkan kunci baru ($K(i)$ baru) yang akan digunakan untuk karakter selanjutnya.
6) Langkah 3) sampai 5) diulangi sebanyak n kali sehingga akan dihasilkan C dengan ukuran $n \times 8$. C akan digunakan sebagai P yang baru.
7) Langkah 1) sampai 6) diulangi sebanyak 15 kali perulangan.

e. Konversi bilangan biner menjadi karakter

Sebelum dikonversi menjadi karakter, P diubah terlebih dahulu kedalam bilangan desimal. Setiap bilangan desimal yang didapat ditambah dengan 32 dikarenakan bilangan desimal 0 – 31 merupakan ASCII *Control Character* yang tidak dapat terbaca

saat dekripsi. Selanjutnya nilai desimal dikonversi menjadi karakter sehingga menghasilkan *ciphertext* dengan panjang yang sama dengan *plaintext*.

P	=	111011100000111111110011
P (desimal)	=	248 15 243
P (desimal + 32)	=	270 47 275
P (karakter)	=	Ď/ē

3.2 Tahap Dekripsi

a. Konversi *ciphertext* menjadi biner
Ciphertext terlebih dahulu diubah dalam desimal sebelum dikonversi menjadi bilangan biner. Saat mendapatkan bilangan desimal, setiap bilangan desimal dikurangi dengan 32 dikarenakan pada saat akhir enkripsi, bilangan desimal *ciphertext* sebelum dikonversi ke karakter ditambah dengan 32. Setelah itu, bilangan desimal dikonversi kedalam bilangan biner 8 digit.

Ciphertext	=	Ď/ē
Ciphertext (desimal)	=	270 47 275
Ciphertext (desimal - 32)	=	248 15 243
Ciphertext (biner)	=	111011100000111111110011

b. Konversi kunci menjadi desimal tiga digit
Proses konversi yang dilakukan sama seperti pada tahap enkripsi.

c. Konversi nilai desimal kunci menjadi sandi morse yang telah dikonversi menjadi biner
Kunci berbentuk desimal tiga digit diubah menjadi sandi morse yang telah dikonversi menjadi biner. Kunci biner (misalkan Kb) berukuran 1×120 dipartisi menjadi 15 bagian dimana setiap bagian berukuran 1×8.

Kunci (desimal) = 079 107 101 079 107 101 079 107
Kunci (sandi morse) = 111111100011110011111111111000011111111011111
111111000111100111111111110000111111111011111
11111000111100111111111111000

$Kb(1)$	=	$Kb(113), \dots, Kb(120)$	=	1 1 1 1 1 0 0 0
$Kb(2)$	=	$Kb(105), \dots, Kb(112)$	=	0 0 1 1 1 1 1 1
$Kb(3)$	=	$Kb(97), \dots, Kb(104)$	=	1 0 0 0 1 1 1 1
$Kb(4)$	=	$Kb(89), \dots, Kb(96)$	=	1 1 1 1 1 1 1 1
$Kb(5)$	=	$Kb(81), \dots, Kb(88)$	=	1 1 1 1 1 0 1 1
$Kb(6)$	=	$Kb(73), \dots, Kb(80)$	=	0 0 0 0 1 1 1 1
$Kb(7)$	=	$Kb(65), \dots, Kb(72)$	=	1 1 1 1 1 1 1 1
$Kb(8)$	=	$Kb(57), \dots, Kb(64)$	=	1 1 1 0 0 1 1 1
$Kb(9)$	=	$Kb(49), \dots, Kb(56)$	=	1 1 1 1 0 0 0 1
$Kb(10)$	=	$Kb(41), \dots, Kb(48)$	=	0 1 1 1 1 1 1 1
$Kb(11)$	=	$Kb(33), \dots, Kb(40)$	=	1 1 1 1 1 1 1 1
$Kb(12)$	=	$Kb(25), \dots, Kb(32)$	=	1 1 1 0 0 0 0 1
$Kb(13)$	=	$Kb(17), \dots, Kb(24)$	=	1 1 1 1 1 1 1 1

$Kb(14)$ = $Kb(9), \dots, Kb(16)$ = 0 0 1 1 1 1 0 0
$Kb(15)$ = $Kb(1), \dots, Kb(8)$ = 1 1 1 1 1 1 1 0

d. Proses dekripsi
   1) Misalkan panjang pesan adalah n. *Ciphertext* biner disusun menjadi matriks (misalkan matriks Cp) dengan ukuran $n \times 8$.
   2) Ambil kunci ke-i. Simbol (i) merupakan "perulangan ke-" yang banyak perulangannya telah ditentukan (15 kali perulangan).
   3) Lakukan *Rotate Left Shift* (RLS) sebanyak 3 kali pada setiap baris Cp lalu tempatkan hasil RLS setiap baris secara berutan kedalam $C(1,:), C(2,:), \dots, C(n-1,:)$. Kunci (misal K) dibentuk dengan anggota $[Kb(i); C(1,:); C(2,:); \dots; C(n-1,:)]$ dengan ukuran $n \times 8$.
   4) Kunci yang digunakan (misal K) dan *ciphertext* (misal C) ditransposisi sesuai dengan Tabel 3. Simbol (j) pada tabel merupakan "perulangan ke-" dengan banyak perulangan sesuai dengan panjang *ciphertext*.

Tabel 3. Tabel transposisi dekripsi

Transposisi *ciphertext*		**Transposisi kunci**	
$Ct(j,1)$	$C(j,1)$	$Kt(j,1)$	$K(j,2)$
$Ct(j,2)$	$C(j,2)$	$Kt(j,2)$	$K(j,5)$
$Ct(j,3)$	$C(j,6)$	$Kt(j,3)$	$K(j,4)$
$Ct(j,4)$	$C(j,7)$	$Kt(j,4)$	$K(j,7)$
$Ct(j,5)$	$C(j,3)$	$Kt(j,5)$	$K(j,6)$
$Ct(j,6)$	$C(j,5)$	$Kt(j,6)$	$K(j,1)$
$Ct(j,7)$	$C(j,8)$	$Kt(j,7)$	$K(j,8)$
$Ct(j,8)$	$C(j,4)$	$Kt(j,8)$	$K(j,3)$

   5) Dilakukan operasi *Exclusive Or* (XOR) antara Ct dan Kt. Hasil XOR dimisalkan sebagai P. P akan digunakan sebagai Cp yang baru.
   6) Langkah 1) sampai 5) diulangi sebanyak 15 kali perulangan.

e. Konversi bilangan biner menjadi karakter
   P yang berisi bilangan biner dikonversi menjadi karakter. Hasil konversi merupakan *plaintext* yang memiliki panjang yang sama dengan *ciphertext*.
   C = 010000010110101101110101
   P (karakter) = Aku

3.3 Analisis Hasil

Algoritma yang diajukan dibuat dengan menggabungkan beberapa algoritma kriptografi yang telah ada yaitu algoritma transposisi, Rotate Left Shift (RLS), dan Exclusif Or (XOR). Untuk membedakannya dengan algoritma lain, algoritma yang diajukan menggunakan sandi morse sebagai variasi dan penguat algoritma. Algoritma yang diajukan diaplikasikan menjadi program yang didesain dan dibuat menggunakan

perangkat lunak Matlab 2015a. Program belum dapat mengakomodasi ASCII Control Character sebagai pesan dikarenakan bagian tersebut tidak dapat disimbolkan untuk diproses. Pesan berhasil dienkripsi menjadi ciphertext dan juga Ciphertext berhasil didekripsi menjadi plaintext yang sama dengan pesan awal. Karakter pada *ciphertext* memiliki perbedaan cukup jauh dengan pesan awal. Lama program untuk memproses pesan bergantung pada panjang pesan. Hal ini disebabkan karena ada bagian pada program yang bergantung pada panjang pesan yang dimasukkan. Semakin panjang pesan, maka perulangan dilakukan sebanyak panjang pesan tersebut. Pesan hasil enkripsi tidak disimpan dalam file tertentu dikarenakan saat disimpan, karakter yang dimuat akan berbeda dengan hasil enkripsi pada program. Hasil tidak akan berubah jika pada hasil dilakukan copy paste. Hal ini disebabkan oleh adanya kemungkinan karakter pada hasil enkripsi diluar karakter ASCII sehingga tidak terbaca oleh sistem penyimpanan file. Algoritma yang diajukan tidak dibandingkan dengan algoritma pengaman teks yang lain serta tidak dinilai kekuatannya dikarenakan dalam studi literatur penulis belum menemukan literatur yang relevan terkait bagaimana cara mengetahui tingkat keamanan dari algoritma pengamanan teks.

4. Kesimpulan

Berdasarkan hasil dan pembahasan yang telah dipaparkan, sandi morse dapat digunakan dalam pengkodean teks. Sandi morse dikonversi menjadi biner yang selanjutnya digunakan sebagai kunci. Perulangan yang dilakukan membuat algoritma yang diajukan semakin kompleks sehingga dapat memperkuat keamanan algoritma. Pola transposisi telah ditentukan dan berbeda pada proses enkripsi dan dekripsi. Operasi XOR dilakukan pada plaintext dan kunci untuk memperoleh hasil digit biner yang lebih bervariasi sehingga ciphertext lebih sulit untuk diuraikan oleh pihak yang tidak bertanggung jawab.

Daftar Pustaka

[1] Amin, M., M., (2016), Implementasi Kriptografi Klasik Pada Komunikasi Berbasis Teks, *Jurnal Pseudocode*, **3(2)**, 129 – 136.

[2] Anwar, S., (2017), Implementasi Pengamanan Data Dan Informasi Dengan Metode Steganografi LSB Dan Algoritma Kriptografi AES, *Seminar Nasional Teknologi Informasi dan Multimedia 2017*, ISSN: 2302-3805, 37-42.

[3] Ariyus, D., Kurniasih, J., Profesi, D., E., (2019), Modifikasi Kunci Simetris Caesar Cipher dan Otp Menggunakan Algoritma Genetika Pada Steganografi, *CSRID Journal*, **11(1)**, 34 – 43.

[4] Bahtiar, R., S., (2018), *Buku Ajar Pengembangan Kepramukaan*, Penerbit UWKS Press, Surabaya.

[5] Darmayanti, I., Astrida, D. N., Arius, D., (2018), Penerapan Keamanan Pesan Teks Menggunakan Modifikasi Algoritma Caisar Chiper Kedalam Bentuk Sandi Morse, *Jurnal IT CIDA*, **4(1)**, 39 – 47.

[6] Gulo, F., (2016), Aplikasi Pembelajaran Konversi Bilangan Menggunakan Metode Computer Assisted Instruction (CAI), *Jurnal Riset Komputer (JURIKOM)*, **3(6)**, 34 –37.

[7] Hayaty, N., (2020), *Buku Ajar: Sistem Keamanan*, Teknik Informatika Universitas Maritim Raja Ali Haji, Tanjungpinang.

[8] Lubis, J. H., (2018), Implementasi Keamanan Data Dengan Metode Kriptografi XOR, *Jurnal Sistem Informasi Kaputama (JSIK)*, **2(2)**, 1 – 4.

[9] Nasution, A. B., (2019), Implementasi Pengaman Data Dengan Menggunakan Algoritma Caesar Cipher dan Transposisi Cipher, *Jurnal Teknologi Informasi*, **3(1)**, 1 – 6.

[10] Nurdin, A. P. N., (2017), Analisa dan Implementasi Kriptografi Pada Pesan Rahasia Menggunakan Algoritma Cipher Transposition, *Jurnal Elektronik Sistem Informasi dan Komputer (Jesik)*, **3(1)**, 1 – 10.

[11] Rubinstein, S. & Salzedo, (2018), *Cryptography*, Springer Undergraduate Mathematics Series, Switzerland.

[12] Sadikin, R., (2012), *Kriptografi untuk Keamanan Jaringan*, Penerbit ANDI, Yogyakarta.

[13] Suhardi, (2016), Aplikasi Kriptografi Data Sederhana Dengan Metode Exlusif-Or (XOR), *Jurnal Teknovasi*, **3(2)**, 23 – 31.

[14] Tamba, M., (2018), Implementasi Algoritma Transposisi Cipher Dalam Sistem Pengamanan Data Pada Jaringan LAN, *Jurnal Technology Informatics & Computer System (Times)*, **7(1)**, 1 – 7.

[15] Widarma, A., Siregar, H. Z., Irawan, M. D., (2019), Teknik Keamanan Data Menggunakan Vigenere Cipher dan Electronic Code Book (ECB), *Jurnal Sains Komputer & Informatika (J-SAKTI)*, **3(2)**, 393 – 400.

[16] Zubair, S., Solichin, A., (2017), Pengenalan Karakter Sandi Rumput Pramuka Menggunakan Jaringan Saraf Tiruan Dengan Metode Backpropagation, *Seminar Nasional Teknologi Informasi dan Multimedia 2017*, ISSN: 2302-3805, 1-6.