SNESTIK

Seminar Nasional Teknik Elektro, Sistem Informasi, dan Teknik Informatika

https://ejurnal.itats.ac.id/snestik dan https://snestik.itats.ac.id

Informasi Pelaksanaan :
SNESTIK I - Surabaya, 26 Juni 2021
Ruang Seminar Gedung A, Kampus Institut Teknologi Adhi Tama Surabaya

Informasi Artikel:
DOI : 10.31284/p.snestik.2021.1769

Prosiding ISSN 2775-5126

Fakultas Teknik Elektro dan Teknologi Informasi-Institut Teknologi Adhi Tama Surabaya
Gedung A-ITATS, Jl. Arief Rachman Hakim 100 Surabaya 60117 Telp. (031) 5945043
Email : snestik@itats.ac.id

Implementasi Kriptografi Enkripsi Pesan Dengan Metode Modifikasi (Initialization Vector) Algoritma RC4

Firdaus Alam Hudi[1], Siti Agustini[2], dan Muchamad Kurniawan[3]

Institut Teknologi Adhi Tama Surabaya[1,2,3]

e-mail: alanfirdaus999@gmail.com

ABSTRACT

Digital information such as text messages in chatting applications which are private and confidential is vulnerable to wiretapping by irresponsible people. These messages have confidentiality which must be protected. One of the ways to protect those messages is using cryptography. Cryptography technique is able to protect the message by randomizing a message and making into different message. Cryptography has several methods for scrambling data. Rivest Code 4 (RC4) algorithm would be used as a from of cryptography. Rivest Code 4 (RC4) algorithm is symmetric cryptography that using same key for encryption and decryption. Rivest Code 4 (RC4) algorithm has the weakness so that someone could attack it easily. This research suggested modification of RC4 by adding block to the initial vector. Modification of RC4 algorithm would gain a level of protection and it would not use as one of the techniques to protect the messages in chatting application and the final result, the text in database would be encrypted.

Keywords: *Cryptography*, RC4, *initialization vector*.

ABSTRAK

Informasi digital seperti pesan teks pada aplikasi *chatting* yang bersifat pribadi dan rahasia yang rentang terhadap penyadapan oleh pihak lain yang tidak bertanggung jawab, tentunya pesan tersebut memiliki kerahasiaan yang harus tetap dijaga. Salah satu cara agar pesan tersebut dapat dijaga yaitu dengan melakukan teknik pengamanan menggunakan kriptografi. Teknik kriptografi dapat mengamankan pesan dengan cara mengacak sebuah pesan tersebut sehingga pesan yang dihasilkan akan berbeda dengan pesan asli. Kriptografi sendiri memiliki berbagai metode untuk mengacak suatu data. Algoritma *Rivest Code 4 (RC4)* akan digunakan sebagai bentuk tujuan dari kriptografi, algoritma *Rivest Code 4 (RC4)* adalah algoritma simetris yang dimana enkripsi dan dekripsi menggunakan satu buah kunci yang sama, oleh karena

itu algoritma RC4 ini mempunyai kelemahan dari beberapa teknik serangan dengan mudah. Maka dari itu pada penilitian ini algoritma RC4 akan dilakukan modifikasi dengan menambahkan blok pada *initial vector* yang digunakan untuk enkripsi dan dekripsi. Dengan modifikasi algoritma RC4 ini akan menambahkan tingkat kemanan dan tidak terlalu mempengaruhi kecepatan proses enkripsi dan dekripsi pada algoritma tersebut. Aplikasi ini dapat dijadikan sebagai salah satu teknik pengamanan pesan pada aplikasi *chatting* yang hasil akhirnya pesan teks pada databasenya akan terenkripsi.

Kata kunci: Kriptografi, RC4, *initialization vector*.

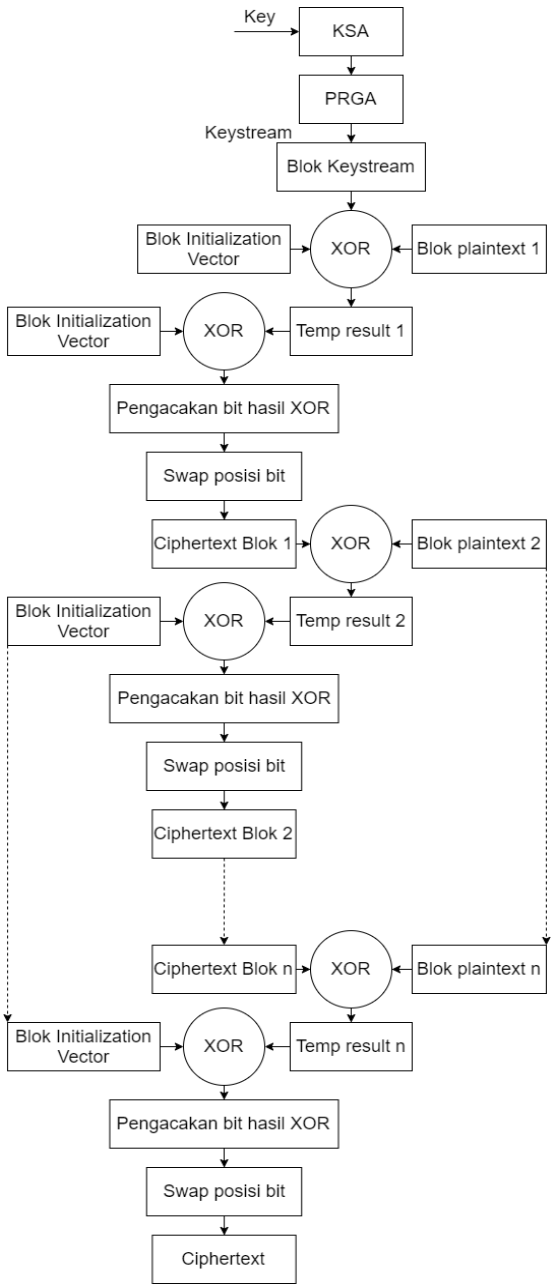
PENDAHULUAN

Semakin berkembangnya teknologi maka semakin dimudahkannya orang-orang untuk saling berkomunikasi dan bertukar informasi. Teknologi informasi saat ini sangat erat hubungannya dengan media komunikasi yang digunakan untuk menyampaikan informasi dari suatu tempat ke tempat yang lainnya. Pada perkembangan teknologi tersebut menyebabkan pihak-pihak tertentu untuk memanfaatkan celah pada kemanan untuk dilakukan penyerangan dan menyalahgunakan data-data pribadi seperti penyadapan informasi atau menggunakan informasi tersebut untuk kepentingan tertentu. Yang dimaksud disini ialah terdapat seseorang yang ingin mengetahui informasi yang disampaikan oleh orang-orang yang informasi itu nantinya akan disalahgunakan.

Salah satu cara untuk meningkatkan keamanan informasi adalah dengan teknik kriptografi. RC4 adalah salah satu metode dari teknik kriptografi dan merupakan kriptografi simetris [1]. Pada penelitian sebelumnya [2], [3], penulis melakukan uji coba terhadap pesan hanya menggunakan algoritma RC4 dan mendapatkan hasil yang relative lebih rentan terhadap serangan *ciphertext-only attack*, dikarenakan tidak terdapat kemanan yang berlapis pada setiap XOR nya, namun dalam pengujian tersebut di implementasikan ke dalam sebuah website yang jika dibandingkan dengan aplikasi android maka aplikasi android lebih sering digunakan oleh banyak orang. Dalam aspek kemanan ini penyandian sutau pesan sangat diperlukan system kemanan untuk melindungi data yang ditransmisikan melalui suatu jaringan komunikasi. Dalam penyandian suatu pesan ini akan dilakukan dengan menggunakan teknik kriptografi. Algoritma RC4 yang akan digunakan untuk melakukan penyandian suatu pesan pada sebuah aplikasi *chatting*. Algoritma RC4 *(Rivest Cipher 4)* mempunyai kelemahan algoritma ini mudah diserang dengan teknik *know-plaintext attack* [4],[5] dan *ciphertext-only attack* [4]. Untuk menambah keamanan pada algoritma RC4 ini akan dilakukan modifikasi, modifikasi pada penelitian ini adalah menambahkan sebuah nilai *initialization vector* (IV) pada setiap operasi XOR antara biner *plaintext* dan biner kunci beserta akan dilakukan pemindahan bit tertentu pada saat selesai setiap iterasi.

METODE

Gambar 1 adalah alur proses modifikasi RC4. Proses modifikasi algoritma RC4 ini dilakukan supaya dapat menambah tingkat kemanan pada algoritma RC4, dengan cara menambah sebuah nilai *initialization vector* yang kemudian akan di-XOR-kan dengan masing-masing *plaintext* untuk enkripsi, dan akan di-XOR-kan dengan *ciphertext* untuk dekripsi. Proses ini akan dilakukan pada setiap semua iterasi untuk mengoptimalkan ketahanan dari penyerangan. Proses yang pertama nilai *initialization vector* akan di XOR dengan kunci akan menghasilkan rangkaian bit yang telah teracak, hasil dari rangkaian bit yang telah teracak tersebut akan dilakukan proses XOR dengan *plaintext*, selanjutnya hasil proses XOR dengan plaintext akan dilakukan proses swap bit untuk enkripsi. Pada proses dekripsi dilakukan proses swap bit terlebih dahulu kemudian hasilnya akan dilakukan operasi XOR dengan kunci dan menghasilkan rangkaian bit, selanjutnya rangkaian bit tersebut yang hasil setelah dilakukan XOR dengan kunci kemudian akan dilakukan proses XOR dengan panjang nilai *initialization vector* dan akan menghasilkan pesan yang kembali seperti semula.



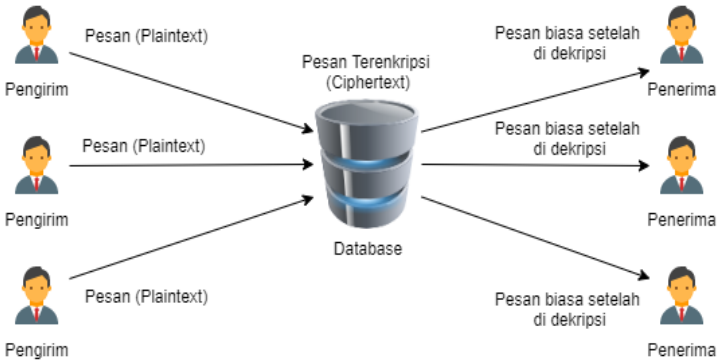
Gambar 1. Proses Modifikasi Algoritma RC4

Skema Proses Aplikasi

Gambar 2 merupakan proses aplikasi sehingga sampai menghasilkan pesan yang terenkripsi, sebagai gambaran umum alur proses rancangan program. Sebagai gambaran umum, tujuan proses enkripsi ini adalah mengubah *plaintext* kedalam bentuk kode-kode yang tidak bisa dibaca atau lebih sering dikenal sebagai *ciphertext*, bentuk kode-kode tersebut akan mengamankan informasi dari pesan yang dikirimkan oleh pengguna. Berikut penjelasannya :

1. Pesan dikirimkan oleh pengguna adalah pesan teks biasa *(plaintext)*.

2. Pada saat pesan tersebut didalam database pesan tersebut akan terenkripsi sehingga menjadi kode-kode atau *ciphertext* yang sulit untuk dibaca.
3. Tetapi pada saat pesan tersebut sampai ke penerima maka pesan tersebut akan di dekripsi sehingga pesan tersebut kembali menjadi pesan biasa.

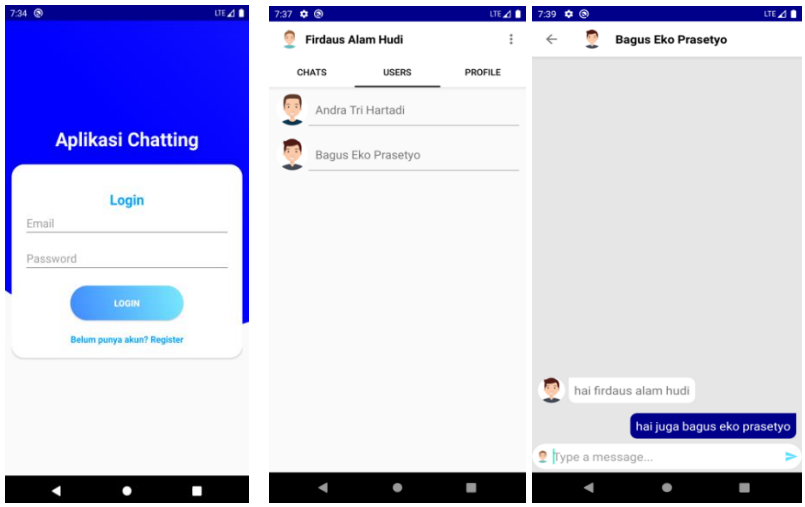


Gambar 2. Skema Rancangan Program

HASIL DAN PEMBAHASAN

Tampilan Aplikasi Chatting

Tampilan halaman login akan tampil pada saat aplikasi pertama kali dijalankan seperti Gambar 3 (a). Pengguna harus mengisi e-mail dan juga password yang telah terdafatar untuk masuk ke halaman menu utama, jika e-mail atau password tidak cocok maka terdapat *warning*. Jika pengguna belum mempunyai akun, maka pengguna harus mendaftar dengan cara klik register dibagian bawah. Halaman register akan tampil dan mengarahkan pada tampilan register, pada halaman register ini terdapat form username, e-mail, dan juga password yang digunakan untuk mendafatarkan diri kedalam aplikasi.



(a) (b) (c)

Gambar 3. a) Halaman Login, b) Halaman User, c) Halaman Pesan

Halaman *users* seperti Gambar 3(b) akan menampilkan pengguna yang telah terdaftar pada aplikasi, jadi jika pengguna ingin melakukan *chatting* tapi pengguna belum pernah

melakukan *chatting* dengan pengguna tersebut sebelumnya, pengguna dapat mencari pengguna lain pada menu ini. Tampilan halaman pesan Gambar 3(c), pengguna saling melakukan komunikasi *chat* antar sesama pengguna, saling bertukar informasi. pada halaman ini semua pesan akan di enkripsi tetapi pengguna tetap dapat melihat pesan yang dikirimkan ke pengguna lainnya, pesan yang terenkripsi berada pada databasenya.

Tampilan Database Firebase

Semua pesan disini di enkripsi, jadi seseorang yang mempunyai hak akses kedalam database nya juga tidak akan dapat membaca pesan tersebut, karena pesan tersebut di enkripsi secara *end-to-end*. Pada gambar Gambar 5 dapat dilihat bahwa pengguna yang mengirimkan pesan dan pesan tersebut akan di enkripsi.

Gambar 4. Tampilan Database Firebase

Pengujian *Avalanche Effect*

Pengujian dilakukan dengan menggunakan *Avalanche Effect*, perbandingan ini dilakukan pada Algoritma RC4 sebelum dimodifikasi, dan Algoritma RC4 setelah dimodifikasi. Pada pengujian ini menggunakan mod yang sesuai dengan panjang kunci, misalnya panjang kunci 4 maka menggunakan mod 4 pada program nya. Hasil *Avalanche Effect* pada Algoritma RC4 sebelum dimodifikasi. Dapat dilihat pada Tabel 1 dan Tabel 2 bahwa Algoritma RC4 setelah dimodifikasi mendapat hasil *Avalanche Effect* lebih tinggi daripada Algoritma RC4 sebelum dimodifikasi, maka dari itu Algoritma RC4 setelah dimodifikasi akan menambah tingkat kemanan daripada Algoritma RC4 yang sebelum dimodifikasi.

Tabel 1. *Avalanche Effect* Algoritma RC4

Plaintext	**Perpindahan bit**	***Avalanche Effect***
Menggunakan Kunci 60, 80, 70, 90, dan mod 4 pada program		
HALO	4	12.5 %
ANDROID	8	14.28 %
APLIKASI CHATTING	11	8.08 %
Menggunakan Kunci 60, 80, 70, 90, 65, 95, 75, 85, dan mod 8 pada program		
HALO	4	12.5 %
ANDROID	7	12.5 %
APLIKASI CHATTING	14	10.29 %
Menggunakan kunci 50, 20, 65, 35, 70, 25, 70, 65, 85, 40, 50, 80, 75, 85, 90, dan mod 15		
HALO	8	25 %
ANDROID	15	26.78 %
APLIKASI CHATTING	38	27.94 %

Tabel 2. *Avalanche Effect* Algoritma Modifikasi RC4

Plaintext	**Perpindahan bit**	***Avalanche Effect***
Menggunakan Kunci 60, 80, 70, 90, dan mod 4 pada program		
HALO	16	50 %
ANDROID	31	50.35 %
APLIKASI CHATTING	68	50 %
Menggunakan Kunci 60, 80, 70, 90, 65, 95, 75, 85, dan mod 8 pada program		
HALO	12	37.5 %
ANDROID	26	46.42 %
APLIKASI CHATTING	56	41.17 %
Menggunakan kunci 50, 20, 65, 35, 70, 25, 70, 65, 85, 40, 50, 80, 75, 85, 90, dan mod 15		
HALO	12	37.5 %
ANDROID	29	51.78 %
APLIKASI CHATTING	61	44.85 %

Hasil persentase pada pengujian *Avalanche Effect* diatas pada awalnya menggunakan 4 kunci dengan mod 4 hasil yang didapatkan Algoritma RC4 setelah dimodifikasi lebih banyak perpindahan bit dan juga hasil persentase nya lebih tinggi dibandingkan dengan algoritma RC4 sebelum dimodifikasi, kemudian untuk membuktikannya lagi dilakukan pengujian kedua dengan menggunakan kunci dan mod lebih banyak yaitu panjang kuncinya 8 dengan mod 8, dan hasilnya dapat dilihat kalau Algoritma RC4 setelah dimodifikasi masih lebih tinggi, kemudian pengujian ketiga menggunakan panjang kunci 15 dan mod 15 juga hasilnya tetap sama yaitu perpindahan bit dari Algoritma RC4 setelah dimodifikai lebih banyak dan persentase nya juga lebih tinggi dibandingkan dengan Algoritma RC4 sebelum dimodifikasi. Begitu juga dari hasil pengujian yang dilakukan untuk menguji kecepatan proses enkripsi dan dekripsi Algoritma RC4 sebelum di modifikasi yaitu :

- Menggunakan *key* 60, 80, 70, 90 dan menggunakan *plaintext* 4 karakter dengan 5 kali pengiriman pesan adalah 0.027 – 0.04 detik, selanjutnya menggunakan *plaintext* 8 karakter dengan 5 kali pengiriman pesan adalah 0.031 – 0.046 detik, dan menggunakan *plaintext* 17 karakter dengan 5 kali pengiriman pesan adalah 0.02 – 0.034 detik.
- Selanjutnya menggunakan *key* 60, 80, 70, 90, 65, 95, 75, 85 dan menggunakan *plaintext* 4 karakter dengan 5 kali pengiriman pesan adalah 0.023 – 0.064 detik, menggunakan *plaintext* 8 karakter dengan 5 kali pengiriman pesan adalah 0.029 – 0.045 detik, dan menggunakan *plaintext* 17 karakter dengan 5 kali pengiriman pesan adalah 0.017 – 0.036 detik.
- Selanjutnya menggunakan *key* 50, 20, 65, 35, 70, 25, 70, 65, 85, 40, 50, 80, 75, 85, 90 dan menggunakan *plaintext* 4 karakter dengan 5 kali pengiriman pesan adalah 0.018 – 0.039 detik, menggunakan *plaintext* 8 karakter dengan 5 kali pengiriman pesan adalah 0.029 – 0.035 detik, menggunakan *plaintext* 8 karakter dengan 5 kali pengiriman pesan adalah 0.02 – 0.033 detik.

Dan hasil kecepatan proses enkripsi dan dekripsi Algoritma RC4 setelah dimodifikasi yaitu :

- Menggunakan *key* 60, 80, 70, 90 dan menggunakan *plaintext* 4 karakter dengan 5 kali pengiriman pesan adalah 0.023 – 0.04 detik, selanjutnya menggunakan *plaintext* 8 karakter dengan 5 kali pengiriman pesan adalah 0.03 – 0.044 detik, dan menggunakan *plaintext* 17 karakter dengan 5 kali pengiriman pesan adalah 0.021 – 0.039 detik.
- Selanjutnya menggunakan *key* 60, 80, 70, 90, 65, 95, 75, 85 dan menggunakan *plaintext* 4 karakter dengan 5 kali pengiriman pesan adalah 0.016 – 0.025 detik, menggunakan *plaintext* 8 karakter dengan 5 kali pengiriman pesan adalah 0.025 – 0.05 detik, dan menggunakan *plaintext* 17 karakter dengan 5 kali pengiriman pesan adalah 0.021 – 0.039 detik.

- Selanjutnya menggunakan *key* 50, 20, 65, 35, 70, 25, 70, 65, 85, 40, 50, 80, 75, 85, 90 dan menggunakan *plaintext* 4 karakter dengan 5 kali pengiriman pesan adalah 0.023 – 0.032 detik, menggunakan *plaintext* 8 karakter dengan 5 kali pengiriman pesan adalah 0.029 – 0.035 detik, menggunakan *plaintext* 8 karakter dengan 5 kali pengiriman pesan adalah 0.021 – 0.029 detik.

Hasil kecepatan enkripsi dan dekripsi tersebut dilakukan pada saat mengirimkan pesan dalam aplikasi *chatting*, jadi hasil perbandingan tersebut tidak terlalu signifikan dalam segi kecepatan yang digunakan pada saat mengirimkan pesan dan pesan tersebut telah di enkripsi pada database dan langsung di dekripsi pada aplikasi.

Dalam aspek kemanan algoritma RC4 setelah dimodifikasi lebih aman daripada Algoritma RC4 sebelum dimodifikasi, karena dapat dilihat dari hasil tabel *Avalanche Effect* diatas, perpindahan bit dari Algoritma RC4 setelah dimodifikasi lebih banyak daripada Algoritma RC4 sebelum dimodifikasi, meskipun Algoritma RC4 setelah dimodifikasi lebih banyak melakukan perpindahan bit tapi pada saat digunakan untuk mengirimkan pesan yang otomatis di enkripsi dan juga di dekripsi pada aplikasi hasil yang diperoleh dari kecepatannya jika dibandingkan dengan algoritma RC4 yang sebelum dimodifikasi tidak terlalu beda jauh. Pengujian kecepatan di uji dalam aplikasi menggunakan lama waktu selama proses pengiriman pesan.

KESIMPULAN

Berdasarkan penelitian yang telah dilakukan, dapat disimpulkann bahwa dengan kriptografi menggunakan metode *Rivest Code 4* (RC4) yang telah dimodifikasi dapat meningkatkan keamanan sebuah teks. Selain itu, hasil penelitian menunjukkan bahwa pada pengujian *Avalanche Effect* Algoritma RC4 sebelum modifikasi mendapat hasil *Avalanche Effect* lebih kecil daripada Algoritma RC4 yang setelah dimodifikasi, dan dari hasil tersebut dapat diketahui bahwa algoritma RC4 setelah dimodifikasi memperoleh perubahan bit yang lebih banyak dan memperoleh hasil *Avalanche Effect* yang lebih tinggi daripada Algoritma RC4 sebelum modifikasi, karena jika perubahan bit semakin banyak maka akan mengakibatkan akan semakin sulitnya Algoritma yang digunakan untuk dipecahkan. Dari segi kecepatan antara Algoritma RC4 setelah dimodifikasi mendapatkan hasil nilai yang hanya beda tipis dengan Algoritma RC4 setelah dimodifikasi, oleh karena itu dari hasil modifikasi Algoritma RC4 ini tidak terlalu mempengaruhi kecepatan pada saat digunakan untuk mengirimkan pesan yang terenkripsi pada databasenya dan langsung di dekripsi pada aplikasi nya.

DAFTAR PUSTAKA

[1] S. Agustini and M. Kurniawan, "Peningkatan Keamanan Teks Menggunakan Kriptografi Dan Steganografi," *SCAN - J. Teknol. Inf. dan Komun.*, vol. 14, no. 3, pp. 33–38, 2019, doi: 10.33005/scan.v14i3.1685.

[2] P. Jindal and B. Singh, "RC4 encryption - A literature survey," *Procedia Comput. Sci.*, vol. 46, no. Icict 2014, pp. 697–705, 2015, doi: 10.1016/j.procs.2015.02.129.

[3] S. R. Fluhrer, I. Mantin, and A. Shamir, "Weaknesses in the Key Scheduling Algorithm of RC4," in *Revised Papers from the 8th Annual International Workshop on Selected Areas in Cryptography*, 2001, pp. 1–24.

[4] A. Klein, "Attacks on the RC4 Stream Cipher," *Des. Codes Cryptogr.*, vol. 48, no. 3, pp. 269–286, Sep. 2008, doi: 10.1007/s10623-008-9206-6.

[5] S. Maitra and G. Paul, "New Form of Permutation Bias and Secret Key Leakage in Keystream Bytes of RC4," 2008, vol. 5086, pp. 253–269, doi: 10.1007/978-3-540-71039-4_16.

- Halaman Ini Sengaja Dikosongkan -