IMPLEMENTASI ALGORITMA RIJNDAEL 128 PADA APLIKASI CHATTING BERBASIS HTML5 WEBSOCKET

Eko Sularsono[1]
ekosularsono@ti.ukdw.ac.id

Willy Sudiarto Raharjo[2]
willysr@ti.ukdw.ac.id

Yuan Lukito[3]
yuanlukito@ti.ukdw.ac.id

Abstract

In the past, web-based chat application didn't consider security as part of must-have requirement, thus many insecure examples were broken in short time after it was released. Data sniffing is one common attack that could be used to attack insecure applications because the data was transferred using an insecure medium, which is HTTP. We propose a new web-based chat application that is built based on HTML5 WebSocket technology using Socket.IO library to improve confidentiality of the messages sent between two or multiple parties. We combine it with NodeJS and Express to facilitate real-time discussion between client and server and vice versa. We also use Rijndael (known as AES - Advanced Encryption Standard) to make sure that the message stays confidential and only known by sender and receiver. To satisfy the integrity property, we apply SHA-3 hash function. By combining SSL/TLS, AES, and SHA-3 hash function, we have added multiple layer of security inside this application and no additional effort needed by the user. Based on conducted experiments, we can conclude that this application could satisfy security requirements (confidentiality and integrity), either on the client or server side.

Kata Kunci: kriptografi, Rijndael, enkripsi, dekripsi websocket, chatting

1. Pendahuluan

Komunikasi merupakan salah satu kebutuhan vital bagi manusia untuk bertahan hidup. Telah banyak inovasi dan terobosan untuk menciptakan layanan komunikasi yang bisa diandalkan. Salah satu contoh terobosan dan inovasi yang telah tercipta adalah aplikasi *chatting* berbasis web. Aplikasi *chatting* berbasis web saat ini menjadi media komunikasi alternatif yang digemari oleh banyak orang. Seiring dengan perkembangannya, aplikasi *chatting* berbasis web dituntut untuk mampu mendistribusikan pesan instan dalam waktu yang sangat singkat. Aplikasi *chatting* berbasis web juga memerlukan sistem keamanan yang baik untuk mengamankan pesan instan bersifat rahasia dan eksklusif. Masalah penyadapan pesan instan pada aplikasi yang berjalan di jaringan Internet juga menjadi isu krusial yang harus ditemukan solusinya.

Sebagai usaha untuk mengatasi permasalahan diatas, pada penelitian ini akan dibuat aplikasi *chatting* berbasis HTML5 WebSocket untuk meningkatkan fleksibilitas, keandalan serta memenuhi kebutuhan komunikasi *realtime* jarak jauh. Fitur *transport* dengan protokol WebSocket tersebut akan diterapkan dengan menggunakan *library* Socket.IO. Sebagai solusi untuk menghindari penyadapan data, distribusi pesan instan yang memiliki informasi berharga pada aplikasi tersebut akan diamankan dengan menggunakan suatu teknik penyandian yang biasa disebut dengan kriptografi. Teknik kriptografi Rijndael 128 nantinya akan diterapkan untuk mengamankan distribusi pesan instan pada aplikasi *chatting* berbasis HTML5 WebSocket yang akan dibuat.

[1] *Program Studi Teknik Informatika, Fakultas Teknologi Informasi,Universitas Kristen Duta Wacana*
[2] *Program Studi Teknik Informatika, Fakultas Teknologi Informasi,Universitas Kristen Duta Wacana*
[3] *Program Studi Teknik Informatika, Fakultas Teknologi Informasi,Universitas Kristen Duta Wacana*

2. Landasan Teori

2.1. WebSocket

WebSocket pada dasarnya menggunakan koneksi dua arah dalam satu socket (Wang, Salim & Moskovits, 2013). WebSocket merupakan teknologi terbaru yang ada pada HTML5 untuk mendukung koneksi dua arah antara klien dan server secara lebih cepat dan ringan dibandingkan dengan metode HTTP tradisional. Dalam WebSocket permintaan HTTP menjadi permintaan untuk membuka koneksi WebSocket dan kemudian menggunakan kembali koneksi tersebut dari klien ke server dan server ke klien.

2.2. Node.js

Node.js adalah sebuah platform yang dibangun di atas Chrome JavaScript Runtime untuk membangun aplikasi yang memiliki performa baik dan cepat. Node.js menggunakan model event-driven, model non-blocking I/O yang membuatnya ringan dan efisien sehingga cocok untuk aplikasi real-time dengan data intensif yang berjalan pada perangkat yang tersebar (Kalemi & Tola, 2013). Kelebihan utama Node.js adalah pada fitur non-blocking I/O yang dimilikinya, sehingga Node.js dapat melayani banyak permintaan dari klien sekaligus tanpa harus menunggu permintaan sebelumnya selesai diproses.

2.3. Express

Express merupakan sebuah *web application framework* berbasis JavaScript untuk node. Framework Express.js merupakan framework yang matang, populer, kuat, teruji dan tidak diragukan lagi untuk layanan web Node.js (Mardanov, 2014). *Framework* ini merupakan pilihan yang tepat untuk mengembangkan halaman web dinamis karena berbasis pada konsep OOP *(Object Oriented Programming)*. Express yang didesain fleksibel, ringan dan minimal sangat memudahkan pengembang halaman web dalam membangun sebuah halaman web modern.

2.4. Socket.IO

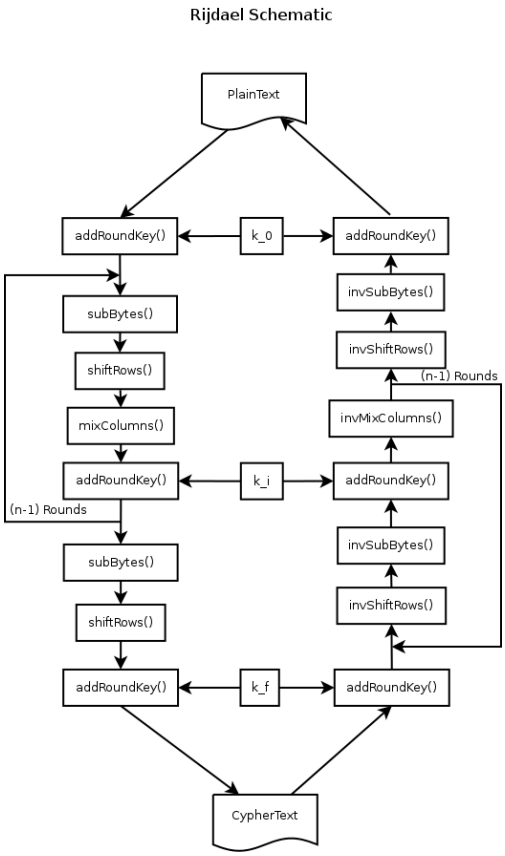
Socket.IO adalah abstraksi dari WebSocket yang memungkinkan komunikasi real-time antara browser dan server (Gelens, Bourget, & Anderson, 2014). Library JavaScript yang dikembangkan oleh LearnBoost ini disediakan untuk mendukung pengembangan aplikasi *realtime* bagi halaman web. Karakter *even-driven* yang ada pada Socket.IO juga dimiliki oleh Node.js, sehingga sangat cocok digunakan berdampingan dengan Node.js.

2.5. Kriptografi

Kriptografi berasal dari bahasa Yunani, *Crypto* dan *graphia*. *Crypto* berarti rahasia dan *graphia* berarti tulisan. Menurut terminologinya, kriptografi adalah ilmu dan seni untuk menjaga keamanan ketika pesan dikirim dari suatu tempat ke tempat lain (Arius, 2008). Inti dari kriptografi adalah untuk menjaga plainteks (atau kunci, atau keduanya) tersembunyi dari penyadap atau bisa juga disebut lawan, penyerang, pencegat, penyusup, lawan, atau secara sederhana disebut musuh (Schneier, 1996). Seiring berjalannya waktu dimana perkembangan teknologi semakin pesat, algoritma kriptografi dibuat menjadi lebih rumit dan kompleks. Kriptografi modern memecahkan masalah ini dengan menggunakan sebuah kunci yang digunakan saat enkripsi dan dekripsi.

2.6. Garis Besar Algoritma Rijndael 128

Rijndael diciptakan oleh dua orang kriptografer asal Belgia, Joan Daemen dan Vincent Rijmen. Kedua kriptografer tersebut mengirimkan proposal Rijndael kepada Institute of Standards and Technology (NIST) untuk proses seleksi standar kriptografi terbaru yang dinamakan Advanced Encryption Standard (AES). NIST menetapkan Rijndael sebagai pemenang AES. Pada Mei 2002, Rijndael dijadikan sebagai standar algoritma kriptografi oleh pemerintah federal Amerika Serikat. Secara garis besar, proses enkripsi dan dekripsi Rijndael dapat diilustrasikan pada Gambar 1.



Gambar 1. Diagram proses enkripsi dan dekripsi algoritma Rijndael 128 (http://2.bp.blogspot.com/-1eCSsEYJTJQ/UB_FA2ZITPI/AAAAAAAAJGk/7ofhuLxHlOA/s1600/rijdaemDiagram.png)

Rijndael adalah suatu blok cipher yang terdiri dari sebuah variabel blok dan sebuah variabel kunci dengan panjang blok dan panjang kunci dapat secara sepesifik ditentukan 128, 192 atau 256 bit (Daeman & Rijmen, 2003). Algoritma Rijndael 128 secara spesifik memiliki panjang *state* dan panjang kunci sebesar 128 bit. *State* merupakan blok plainteks yang akan dienkripsi atau didekripsi. 128 bit data pada *state* dan kunci tersebut dipecah menjadi 16 bagian. Setiap bagian berisi masing-masing 1 byte data (8 bit data). Tiap byte data pada *state* dan kunci tersebut masing-masing dimasukkan kedalam 16 byte matriks berukuran 4x4 dalam notasi heksadesimal, seperti diilustrasikan pada Gambar 2.

41	4b	20	41
59	49	43	20
4f	54	4f	59
20	41	42	41

53	4e	20	49
45	47	53	50
4d	41	4b	53
41	54	52	49

Gambar 2. Hasil pemetaan 128 bit data *state* (kiri) dan kunci (kanan) dalam notasi heksadesimal pada 16 byte matriks.

State dan kunci dengan notasi heksadesimal yang telah dimasukkan ke dalam matriks berukuran 4x4 selanjutnya akan masuk ke dalam dua proses yang berbeda. *State* akan masuk ke dalam proses enkripsi atau dekripsi, sedangkan kunci akan masuk ke dalam proses penjadwalan kunci (key schedule).

2.7. Proses Enkripsi Algoritma Rijndael 128

Transformasi perputaran algoritma Rijndael terdiri dari empat jenis transformasi yang berbeda (Daemen & Rijmen, 2003). Empat jenis transformasi seperti yang dapat dilihat pada Gambar 1. adalah SubBytes, ShiftRows, MixColumns, dan AddRoundKey untuk proses enkripsi; serta invSubBytes, invShiftRows, invMixColumns, dan AddRoundKey untuk proses dekripsi. Sebelum memasuki putaran pertama dalam proses enkripsi, dilakukan operasi bitwise XOR (Exclusive OR) pada *state* awal dengan kunci *(cipher key)*. Tahap ini biasanya disebut dengan initial round. Penjelasan lebih mendalam mengenai transformasi SubBytes, ShiftRows, Mixcolumns, dan AddRoundKey pada proses enkripsi akan dijelaskan di bawah ini:

a. SubBytes

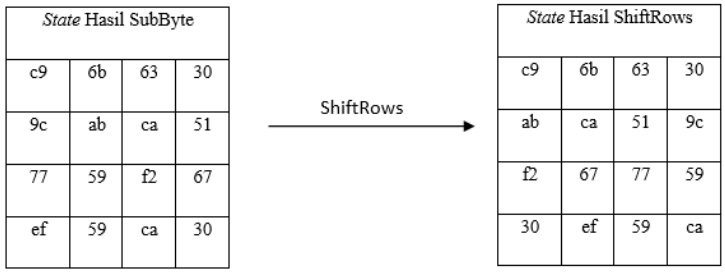
SubByte adalah substitusi byte dengan menggunakan tabel substitusi (S-Box) pada Tabel 1. Masing-masing byte pada *state* setelah melalui initial round akan disubtitusikan dengan menggunakan S-Box.

Tabel 1.
Tabel data yang dipakai sebagai dasar pengujian

x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

b. ShiftRows

ShiftRows adalah pergeseran byte menurut aturan tertentu pada setiap baris array *state*. Masing-masing byte pada tiap baris array *state* setelah operasi SubByte akan mengalami pergeseran menurut aturan yang berlaku. Baris pertama *state* setelah operasi SubByte tidak mengalami pergeseran. Baris kedua *state* setelah operasi SubByte mengalami pergeseran ke kiri sebanyak satu kali. Baris ketiga *state* setelah operasi SubByte mengalami pergeseran ke kiri sebanyak dua kali, sedangkan baris keempat *state* setelah operasi SubByte mengalami pergeseran ke kiri sebanyak tiga kali. Ilustrasi proses ShiftRows terhadap *state* hasil SubByte dapat dilihat pada Gambar 3.



Gambar 3. Ilustrasi tranformasi ShiftRows

c. MixColomns

MixColumns adalah operasi untuk mengacak data pada *state* hasil dari transformasi ShiftRows. Masing-masing kolom pada *state* (4 byte data) dikalikan dengan menggunakan perkalian matriks pada Gambar 4. Perkalian matriks tersebut dapat dijabarkan seperti formula perkalian yang ada pada Gambar 5. Perkalian untuk formula tersebut berdasarkan pada Rijndael's Galois field GF(2^8).

$$\begin{pmatrix} s'0,1 \\ s'1,1 \\ s'2,1 \\ s'3,1 \end{pmatrix} = \begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix} \begin{pmatrix} s0,1 \\ s1,1 \\ s2,1 \\ s3,1 \end{pmatrix}$$

Transform Matrix of Mix Columns

Gambar 4. Perkalian matriks transformasi MixColumns
(http://http.developer.nvidia.com/GPUGems3/elementLinks/36fig08.jpg)

$$s'_{0,c} = (\{02\} \bullet s_{0,c}) \oplus (\{03\} \bullet s_{1,c}) \oplus s_{2,c} \oplus s_{3,c}$$
$$s'_{1,c} = s_{0,c} \oplus (\{02\} \bullet s_{1,c}) \oplus (\{03\} \bullet s_{2,c}) \oplus s_{3,c}$$
$$s'_{2,c} = s_{0,c} \oplus s_{1,c} \oplus (\{02\} \bullet s_{2,c}) \oplus (\{03\} \bullet s_{3,c})$$
$$s'_{3,c} = (\{03\} \bullet s_{0,c}) \oplus s_{1,c} \oplus s_{2,c} \oplus (\{02\} \bullet s_{3,c})$$

Gambar 5. Formula dari perkalian transfromasi MixColumns pada Gambar 4.
(http://students.ceid.upatras.gr/~mprokala/techarticles/cryptography/AES/images/image048.png)

d. AddRoundKey

Proses AddRoundKey yaitu melakukan bitwise XOR antara *state* sekarang (*state* hasil MixColumns) dengan round key. Round key adalah kunci yang dibangkitkan melalui proses penjadwalan kunci atau yang biasa disebut Rijndael key schedule. Pada algoritma Rijndael 128 terdapat 10 Round key yang harus dibangkitkan. Penjadwalan kunci Rijndael 128 secara lebih detail akan dijelaskan pada bagian 2.9.

2.8. Proses Dekripsi Algoritma Rijndael 128

Proses dekripsi pada algoritma rijndael menggunakan transformasi yang berlawanan dari proses enkripsi. Transformasi yang berlawanan tersebut digunakan untuk menghasilkan inverse cipher sehingga cipherteks dapat dikembalikan menjadi plainteks. Penjelasan lebih mendalam mengenai transformasi InvShiftRows, InvSubBytes, dan InvMixColumns akan dijelaskan di bawah ini:

a. InvSubBytes

InvSubByte adalah substitusi byte dengan menggunakan tabel substitusi Inverse S-Box pada Tabel 2. Tabel subtitusi Inverse S-Box merupakan kebalikan dari tabel S-Box. Sebagai contoh dilakukan operasi Inv-SubByte terhadap masing-masing byte pada *state* hasil transformasi InvShiftRows. Masing-masing byte pada *state* yang akan didekripsi disubtitusikan dengan menggunakan tabel Inverse S-Box.

Tabel 2.
Tabel data yang dipakai sebagai dasar pengujian

		y														
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
	1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
	2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
	3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
	4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
	5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
	6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
	7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
	8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
	9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
	a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
	b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
	c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
	d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
	e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
	f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

b. InvShiftRows

InvShiftRows adalah pergeseran byte menurut aturan tertentu pada setiap baris array *state*. Masing-masing byte pada tiap baris array *state* awal akan mengalami pergeseran menurut aturan yang berlaku. Baris pertama *state* awal tidak mengalami pergeseran. Baris kedua *state* awal mengalami pergeseran ke kiri sebanyak tiga kali. Baris ketiga *state* awal mengalami pergeseran ke kiri sebanyak dua kali, sedangkan baris keempat *state* awal mengalami pergeseran ke kiri sebanyak satu kali.

c. InvMixColumns

Transformasi MixColumns memiliki invers yang disebut dengan transformasi InvMixColumns. Transformasi InvMixColumns digunakan dalam proses dekripsi cipherteks. Operasi perhitungan pada transformasi InvMixColumns sama dengan transformasi MixColumns, namun kolom matriks yang digunakan dalam perkalian pada transformasi InvMixColumns diubah menjadi matriks pada gambar 6.

$$\begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 0e & 09 & 0b \\ 0b & 0d & 0e & 09 \end{bmatrix}$$

Gambar 6. Matriks untuk perkalian transfromasi InvMixColumns

2.9. Penjadwalan Kunci Rijndael 128

Penjadwalan kunci atau yang biasa disebut Rijndael key schedule digunakan untuk membangkitkan round key. Pada algoritma Rijndael 128 dibutuhkan sepuluh round key. Masing-masing round key tersebut akan digunakan dalam proses AddRoundKey pada setiap putaran enkripsi atau dekripsi. Dalam penjadwalan kunci, untuk membangkitkan round key dibutuhkan tabel Rcon seperti yang dipaparkan pada Tabel 3.

Tabel 3.
Tabel data yang dipakai sebagai dasar pengujian

01	02	04	08	10	20	40	80	1b	36
00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00

Ada beberapa operasi yang digunakan untuk membangkitkan kunci menjadi sepuluh round key. Operasi-operasi yang digunakan antara lain adalah operasi pergeseran baris untuk mencari RotWord, operasi SubByte dan operasi bitwise XOR. Di bawah ini akan dijelaskan proses pembangkitan round key lebih mendalam:

a. Langkah pertama yaitu mencari RotWord. Masing-masing nilai yang ada pada kolom kunci paling akhir akan bergeser ke atas sebanyak satu kali. Nilai yang ada pada baris pertama bergeser menjadi di baris keempat, nilai baris kedua bergeser menjadi di baris pertama, nilai baris ketiga bergeser menjadi baris kedua dan nilai baris keempat bergeser menjadi baris ketiga.
b. Langkah kedua adalah melakukan operasi SubByte terhadap RotWord dengan tabel S-Box pada Tabel 1.
c. Melakukan operasi bitwise XOR kolom pertama kunci dengan hasil SubByte dari RotWord dan kolom pertama pada tabel Rcon, untuk mencari nilai kolom pertama round key 1.
d. Melakukan operasi bitwise XOR kolom kedua kunci dengan kolom pertama round key 1, untuk mencari nilai kolom kedua round key 1.
e. Melakukan operasi bitwise XOR kolom ketiga kunci dengan kolom kedua round key 1, untuk mencari nilai kolom ketiga round key 1.
f. Melakukan operasi bitwise XOR kolom keempat kunci dengan kolom ketiga round key 1, untuk mencari nilai kolom keempat round key 1.

g. Didapatkan round key 1. Round key 1 yang didapat tersebut selanjutnya digunakan sebagai dasar pembangkitan round key 2. Round key 2 yang didapatkan nantinya juga digunakan untuk membangkitkan round key 3, demikian seterusnya hingga kesepuluh round key dapat dibangkitkan.

3. Hasil dan Pembahasan

3.1. Implementasi Penggunaan Protokol WebSocket

Pada penelitian ini, fitur transport dengan protokol WebSocket yang ada pada Socket.IO digunakan untuk menyediakan komunikasi *realtime* antara client dan server pada aplikasi *chatting* yang dibangun. Socket.IO digunakan berdampingan dengan Node.js dan *web application framework* Express. Implementasi penggunaan protokol WebSocket tersebut dapat dilihat pada listing program pada Gambar 7. Fitur socket.join(room) yang ada pada Socket.IO juga digunakan untuk memastikan pesan yang dikiirim dan diterima pada aplikasi chatting hanya dapat dilihat oleh pengguna aplikasi di *room* yang sama.

```
var app = require('express')();
server = require('http').createServer(app);
var io = require('socket.io').listen(server);
io.set('transports', ['websocket']);
server.listen(process.env.PORT || 80);
```

Gambar 7. Implementasi Penggunaan Socket.IO

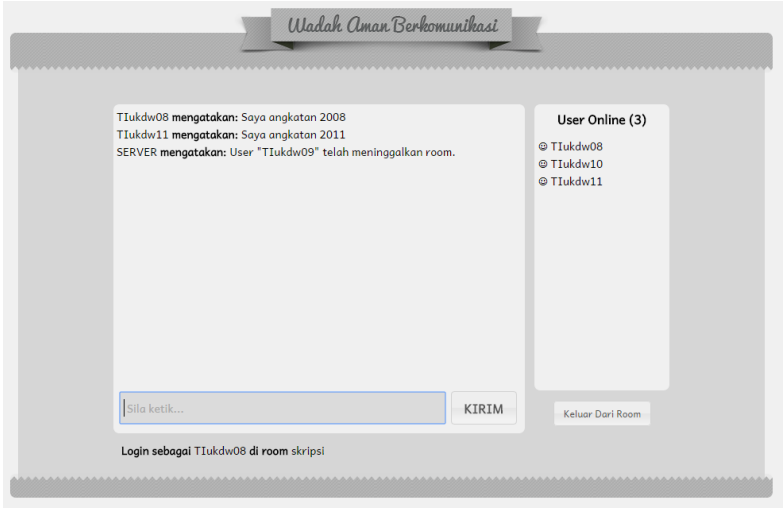
3.2. Implementasi Enkripsi Dan Dekripsi Pesan Dengan Algoritma Rijndael

Enkripsi dan dekripsi terhadap pesan yang dikirim atau diterima seluruhnya dilakukan pada *client-side* (browser). Server hanya berfungsi mengatur dan mengarahkan pesan kepada penerima yang berhak. Proses enkripsi dan dekripsi berlangsung otomatis setelah pesan dikirim atau diterima oleh pengguna aplikasi. Proses enkripsi dan dekripsi pesan menggunakan algoritma Rijndael pada library CryptoJS versi 3.1.2. yang diperoleh dari https://code.google.com/p/crypto-js/downloads/list. Sebelum pesan dikirim menuju server, plainteks pesan merngalami proses *hashing* menggunakan algoritma SHA3 yang ada pada library CryptoJS versi 3.1.2. Hash dari pesan asli dan cipherteks hasil enkripsi selanjutnya dikirimkan ke server. Saat pesan diterima dalam bentuk cipherteks dan hash dari pesan asli, browser *(client-side)* secara otomatis melakukan dekripsi cipherteks. Pesan asli yang didapatkan dari proses dekripsi selanjutnya mengalami proses hashing menggunakan algoritma SHA3. Hasil dari hashing tersebut selanjutnya dibandingkan dengan hash pesan asli yang diterima bersama dengan cipherteks. Perbandingan ini dilakukan untuk mengetahui pesan yang diterima telah mengalami perubahan atau tidak selama diperjalanan.

3.3. Implementasi Antarmuka Sistem

Gambar 8. dan Gambar 9. merupakan capture dari aplikasi chatting berbasis HTML5 WebSocket yang telah dibuat. Gambar 8. merupakan tampilan dari form “Buat Atau Masuk Room”. Form ini muncul pertama kali ketika pengguna mengunjungi URL aplikasi yang telah terhubung pada jaringan Internet. Gambar 9. merupakan tampilan dari form “Chatting”. Form “Chatting” digunakan oleh pengguna untuk melakukan percakapan dengan pengguna lain yang sedang online.

Gambar 8. . Implementasi antarmuka form "Buat Atau Masuk Room"

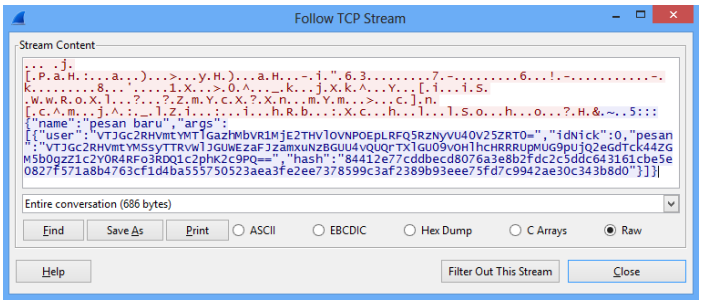


Gambar 9. Implementasi antarmuka form "Chatting"

3.4. Pengujian Sistem Terhadap Penyadapan Man-In-The-Middle Attack (MITMA)

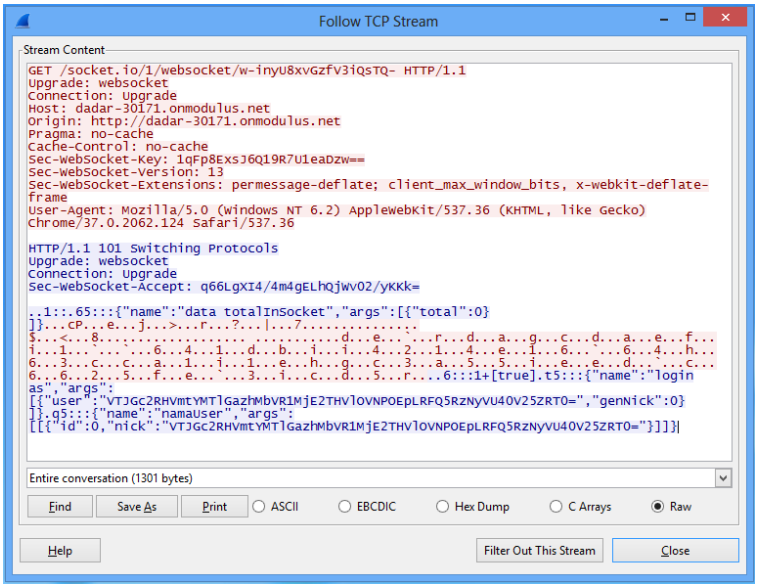
Pengujian terhadap MITMA (Man-in-the-middle attack) dilakukan untuk melihat apakah paket data pesan terjaga keamanannya saat dalam perjalanan. Pengujian terhadap aplikasi chatting berbasis HTML5 WebSocket yang telah dibuat dilakukan dengan menggunakan program Wireshark versi 1.10.7. Pengujian terhadap MITMA dibagi menjadi dua bagian. Bagian pertama yaitu menguji keamanan data dalam perjalanan saat sistem tidak menggunakan protokol keamanan SSL (Secure Socket Layer) dengan koneksi TLS 1.2. Bagian kedua yaitu menguji keamanan data dalam perjalanan saat sistem menggunakan protokol keamanan SSL. Kedua bagian pengujian tersebut akan menggunakan "TIukdw08" sebagai data nickname, "skripsi" sebagai data *room* dan "p455w0Rd" sebagai data password *room* untuk menganalisa keamanan data saat pengguna membuat atau bergabung ke dalam *room*. Analisis keamanan data pesan yang dikirim menggunakan data "Pesan ini adalah pesan rahasia untuk TIukdw14", sedangkan analisis keamanan pesan yang diterima menggunakan data "Kepada TIukdw08, pesan rahasia diterima". Hasil pengujian terhadap penyadapan Man-In-The-Middle Attack secara lebih mendetail dipaparkan pada bagian dibawah ini:

a. Keamanan Paket Data Tanpa Protokol SSL.
   - Analisis keamanan pengiriman pesan tanpa menggunakan protokol SSL.



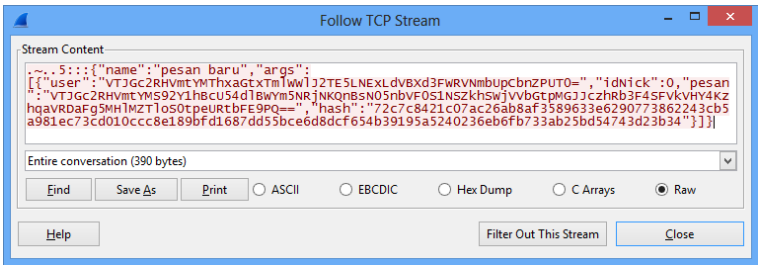
Gambar 10. Penyadapan data saat pengguna mengirimkan pesan tanpa menggunakan protokol SSL

- Analisis keamanan data saat membuat atau bergabung ke dalam *room* tanpa menggunakan protokol SSL.



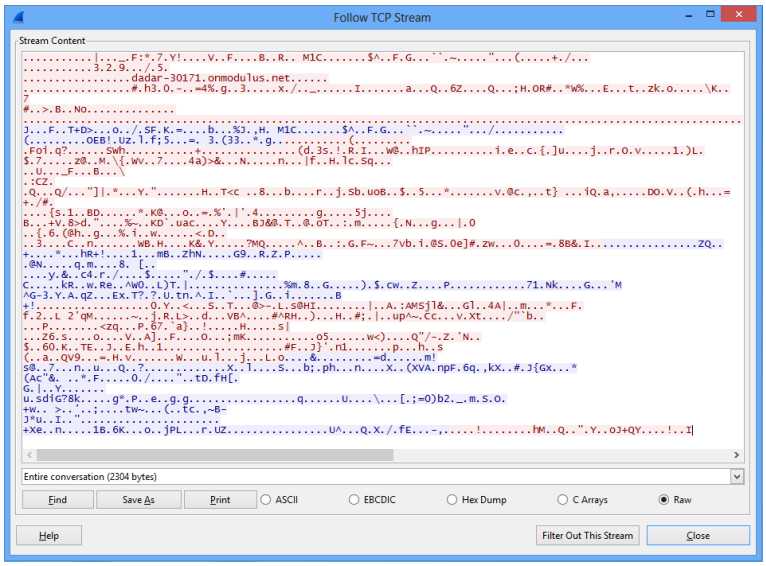
Gambar 11. Penyadapan data saat pengguna membuat atau bergabung ke dalam *room* tanpa menggunakan protokol SSL

- Analisis keamanan penerimaan pesan tanpa menggunakan protokol SSL.



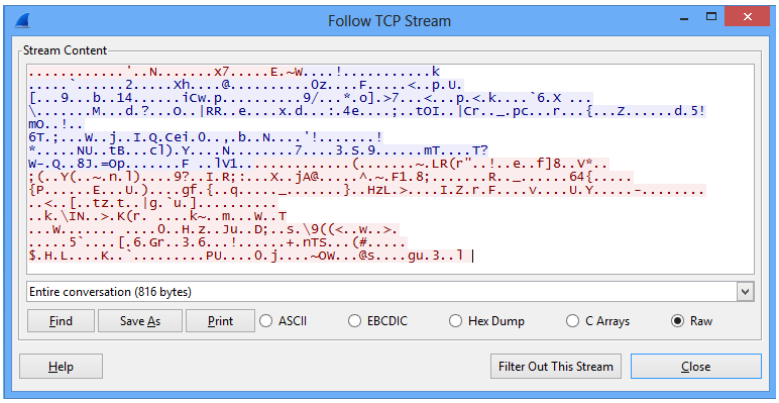
Gambar 12. Penyadapan data saat pengguna menerima pesan tanpa menggunakan protokol SSL

b. Keamanan Paket Data Dengan Protokol SSL.
   - Analisis keamanan data saat membuat atau bergabung ke dalam *room* dengan menggunakan protokol SSL.



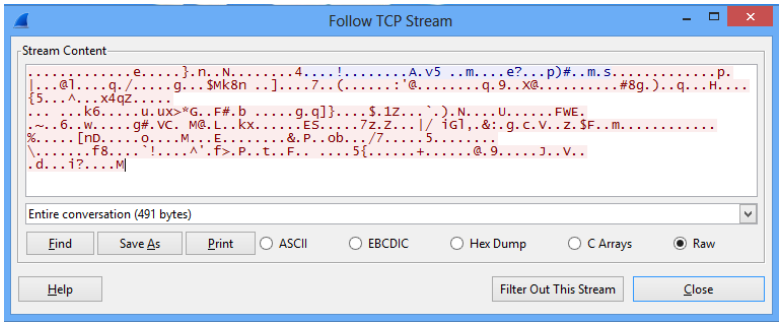
Gambar 13. Penyadapan data saat pengguna membuat atau bergabung ke dalam *room* dengan menggunakan protokol SSL

- Analisis keamanan pengiriman pesan dengan menggunakan protokol SSL.



Gambar 14. Penyadapan data saat pengguna mengirim pesan dengan menggunakan protokol SSL

- Analisis keamanan penerimaan pesan dengan menggunakan protokol SSL.



Gambar 15. Penyadapan data saat pengguna menerima pesan dengan menggunakan protokol SSL

Pengujian terhadap penyadapan Man-In-The-Middle Attack dengan program Wireshark memperlihatkan seluruh paket data yang masuk atau keluar dari klien dan server. Pada

pengujian pertama dapat dilihat bahwa paket data yang masuk dan keluar tanpa menggunakan protokol keamanan SSL masih sulit diketahui maknanya. Proses enkripsi dan *hashing* yang dilakukan sebelum paket data dikirimkan membuat paket data sulit dibaca. Pengujian kedua memperlihatkan bahwa penggunaan protokol keamanan SSL memberi keamanan berlapis terhadap paket data di dalam perjalanan. Paket data yang dikirim menggunakan protokol keamanan SSL menjadi lebih sulit dibaca dibandingkan dengan tidak menggunakan protokol keamanan SSL.

3.5. Analisis Pengujian Keamanan Variabel Yang Tersimpan Pada Server

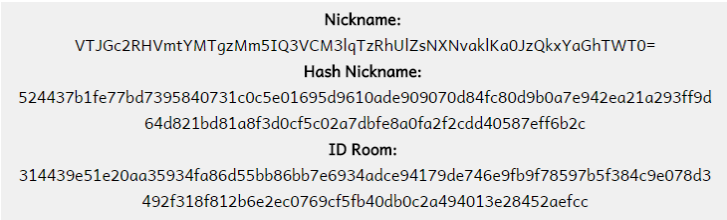
Pengujian keamanan di sisi server dilakukan untuk melihat apakah data (variabel) yang tersimpan di dalam socket (server) dapat dibaca maknanya atau tidak. Server hanya menyimpan cipherteks *nickname*, hash dari *nickname* dan ID *room*. Cipherteks *nickname* digunakan untuk menampilkan list semua user yang sedang online pada *room* tertentu. Hash dari *nickname* digunakan pada pengecekan kesamaan *nickname* untuk membangkitkan *nickname* baru jika *nickname* tersebut telah terdaftar pada *room* yang sama. ID *room* digunakan pada fitur socket.join(room) sebagai dasar pengelompokan user ke dalam *room* yang sama. ID *room* tersebut dibangkitkan dengan menggunakan hash dari nama *room* dan hash dari password *room*. Semua data yang ada di server secara khusus data asli *nickname*, *room* dan password *room* diharapkan tidak dapat dibaca di server. Data asli *nickname*, *room* dan password *room* yang dipakai sebagai dasar pengujian dapat dilihat pada Tabel 4. di bawah ini.

Tabel 4.
Tabel data yang dipakai sebagai dasar pengujian

Data Uji	Nickname	Nama Room	Password Room
1	TIukdw08	skripsi	p455w0Rd
2	TIukdw09	skripsi	p455w0Rd
3	TIukdw10	skripsi	p455w0Rd
4	TIukdw11	skripsi	p455w0Rd
5	processor	koMpUtEr	skripsiTI
6	memory	koMpUtEr	skripsiTI
7	hardisk	koMpUtEr	skripsiTI
8	keyboard	koMpUtEr	skripsiTI
9	nodejs	javascript	RahasiaA
10	socketio	javascript	RahasiaA

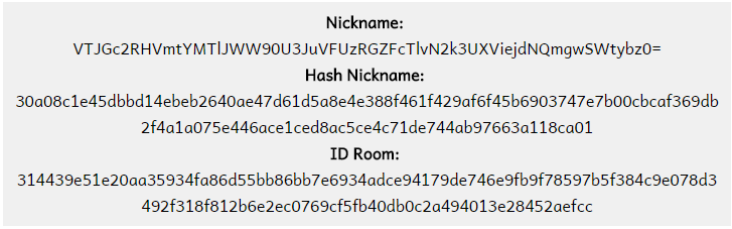
Pada pengujian keamanan di sisi server ini, akan ditampilkan nilai dari semua variabel yang tersimpan di dalam socket (server). Semua nilai dari variabel yang ditampilkan tersebut akan menjadi acuan untuk menarik kesimpulan apakah aplikasi chatting berbasis HTML5 WebSocket yang dibuat aman atau tidak. Hasil pengujian dari sepuluh data uji pada Tabel 1. secara akan dipaparkan di bawah ini.

a. Pengujian Pertama



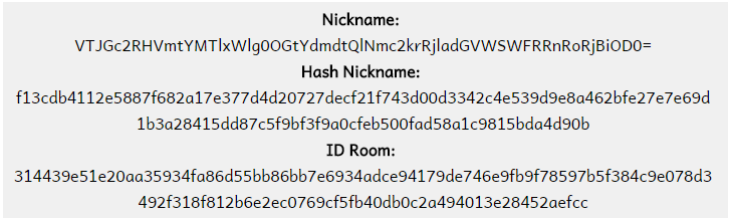
Gambar 16. Semua variabel di dalam socket pada pengujian pertama

b. Pengujian Kedua



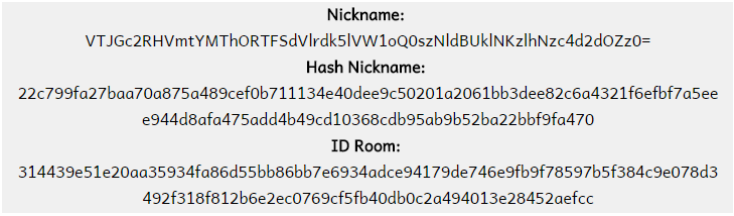
Gambar 17. Semua variabel di dalam socket pada pengujian kedua

c. Pengujian Ketiga



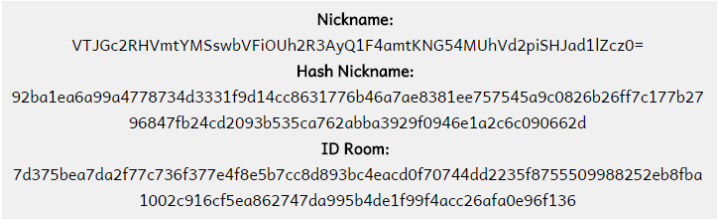
Gambar 18. Semua variabel di dalam socket pada pengujian ketiga

d. Pengujian Keempat



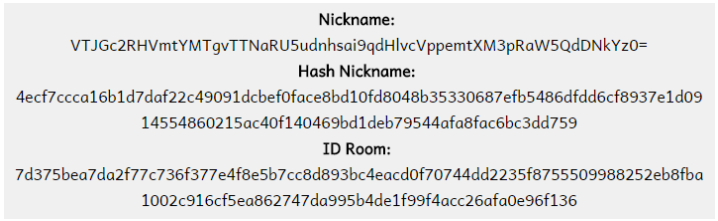
Gambar 19. Semua variabel di dalam socket pada pengujian keempat

e. Pengujian Kelima



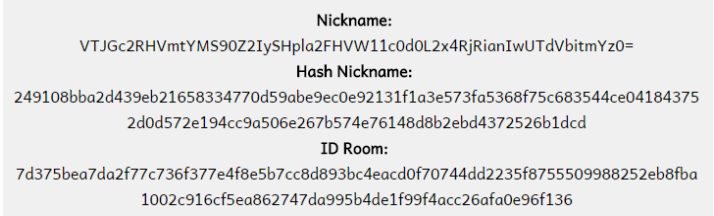
Gambar 20. Semua variabel di dalam socket pada pengujian kelima

f. Pengujian Keenam



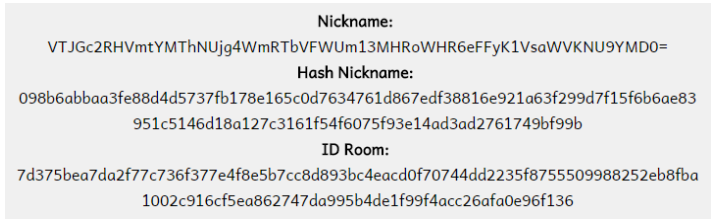
Gambar 21. Semua variabel di dalam socket pada pengujian keenam

g. Pengujian Ketujuh



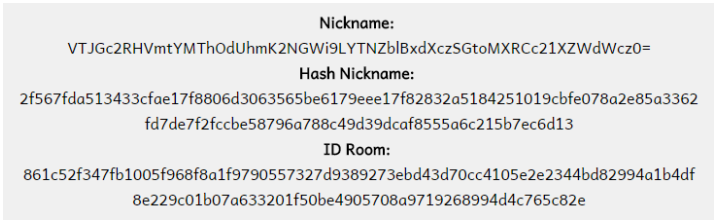
Gambar 22. Semua variabel di dalam socket pada pengujian ketujuh

h. Pengujian Kedelapan



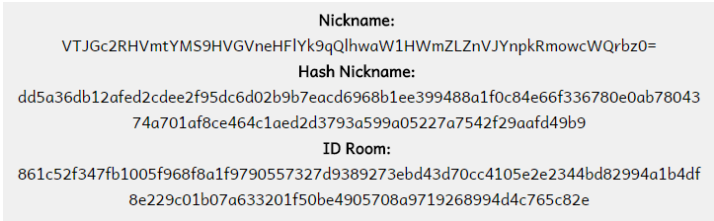
Gambar 23. Semua variabel di dalam socket pada pengujian kedelapan

i. Pengujian Kesembilan



Gambar 24. Semua variabel di dalam socket pada pengujian kesembilan

j. Pengujian Kesepuluh



Gambar 25. Semua variabel di dalam socket pada pengujian kesepuluh

Data yang tersimpan di dalam socket (server) sebelumnya mengalami proses enkripsi dan *hashing*. Data *nickname* dan nama *room* yang telah tersimpan di dalam socket telah mengalami proses enkripsi dengan menggunakan algoritma Rijndael. Data password yang telah tersimpan di dalam socket juga telah mengalami proses *hashing* dengan menggunakan algoritma SHA3. Dari sepuluh pengujian diatas, dapat dilihat bahwa data asli dari semua data yang tersimpan di dalam socket tidak dapat dibaca dengan mudah.

4. Kesimpulan

Setelah dilakukan pengujian dan analisis terhadap aplikasi yang telah dibuat, dapat disimpulkan bahwa:

1. Penggunaan platform Node.js memberi kemudahan dalam membangun aplikasi chatting dengan performa baik dan cepat. Hal ini disebabkan karena Node.js memiliki fitur non-blocking I/O yang sangat dibutuhkan pada aplikasi realtime.
2. Socket.IO menyediakan fitur transport dengan protokol WebSocket dan dapat digunakan berdampingan dengan Node.js. Hal ini membuat komunikasi realtime antara client dan server pada aplikasi chatting yang dibuat dapat diakomidir dengan baik.
3. Penggunaan fitur socket.join (room) pada Socket.IO memastikan pesan yang dikiirim dan diterima pada aplikasi chatting hanya dapat dilihat oleh pengguna aplikasi di room yang sama.
4. Algoritma Rijndael dapat diimplementasikan pada aplikasi chatting berbasis HTML5 WebSocket yang telah dibangun. Semua proses enkripsi dan dekripsi dengan algoritma Rijndael dilakukan di sisi client. Hal ini membuat distribusi data pengguna dan pesan instan pada aplikasi tersebut lebih terjaga keamanannya.
5. Data nickname dan nama room yang disimpan pada server sebelumnya telah mengalami proses enkripsi dengan algoritma Rijndael. Data password room yang tersimpan pada server sebelumnya telah mengalami proses hashing menggunakan algoritma SHA-3. Hal ini membuat keamanan data yang tersimpan di dalam server lebih terjaga keamanannya.

Daftar Pustaka

Arius, D. (2008). Pengantar Ilmu KRIPTOGRAFI, Teori, Analisis dan Implementasi. Andi Offset, Yogyakarta.

Daemen, J., & Rijmen, V. (2003). AES proposal: Rijndael. Diakses pada tanggal 26 Juli 2013 dari http://csrc.nist.gov/archive/aes/rijndael/Rijndael-ammended.pdf

Gelens, J., Bourget, A., & Anderson, J. (2014). gevent-socketio Documentation. (Release 0.3.1). Diakses pada tanggal 16 Mei 2014 dari https://media.readthedocs.org/pdf/gevent-socketio/latest/gevent-socketio.pdf

Kalemi, E., & Tola, K. (2013). Updating web content in real time using Node. js. Diakses pada tanggal 19 Februari 2014 dari http://dspace.epoka.edu.al/mobile/bitstream/handle/1/844/paper_12.pdf?sequence=1

Mardanov, Azat. (2014). Express.js Guide - The Comprehensive Book on Express.js. Diakses pada tanggal 16 Mei 2014 dari http://samples.leanpub.com/express-sample.pdf

Scheneier, B. (1996). Applied Cryptography Second Edition: protocols, algorithms, and source code in C. John Wiley and Sons. Diakses pada tanggal 26 Februari 2014 dari http://www.cse.iitk.ac.in/users/anuag/crypto.pdf

Wang, V., Salim, F., & Moskovits, P. (2013). The Definitive Guide to HTML5 WebSocket. Apress. Diakses pada tanggal 24 Juli 2014 dari http://it-ebooks.info/go.php?id=2026-1395905654-f114ccf36d0bdde0cbe89d7c0bf4e440