Terakreditasi SINTA Peringkat 2
Surat Keputusan Dirjen Penguatan Riset dan Pengembangan Ristek Dikti No. 10/E/KPT/2019
masa berlaku mulai Vol. 1 No. 1 tahun 2017 s.d Vol. 5 No. 3 tahun 2021

Terbit online pada laman web jurnal: **http://jurnal.iaii.or.id**

JURNAL RESTI
(Rekayasa Sistem dan Teknologi Informasi)
Vol. 4 No. 6 (2020) 1149 – 1156 ISSN Media Elektronik: 2580-0760

Implementasi Enkripsi dan Otentikasi Transmisi Data ZeroMQ Menggunakan Advanced Encryption Standard

I Made Sukarsa[1], I Made Rama Pradana[2], Putu Wira Buana[3]
[1,2,3]Program Studi Teknologi Informasi, Fakultas Teknik, Universitas Udayana
[1]sukarsa@unud.ac.id*, [2]ramapradana67@gmail.com, [3]wbhuana@gmail.com

Abstract

Communication via sockets is used to transmit information between applications or between processes over network or locally. ZeroMQ is a library for sending messages using sockets that are quite well known. Talking about sending data, message security is an important part that needs to be taken into account, especially when sending data over a network. ZeroMQ sends messages openly without securing the messages sent. This is evidenced by research which states that ZeroMQ does not have a security layer for sending messages over the network and direct observation of message packets using the wireshark application. Therefore, this study creates a method of securing and authenticating message delivery using AES (Advanced Encryption Standard) CBC (Cipher Block Chaining) mode combined with an authentication method. The AES CBC mode was chosen because it is faster than other methods and has strong encryption. This encryption and authentication are used so that the sender and recipient of the message are both valid senders and recipients so that no message changes during message delivery and messages can only be opened by the message recipient and the sender of the message. Based on the tests conducted, it shows an average increase in delivery time of 1.5 seconds when encryption and authentication are added.

Keywords: authentication, encryption, parallel pipeline, socket, zeromq

Abstrak

Komunikasi melalui soket digunakan untuk mengirimkan informasi antar aplikasi atau antar proses. ZeroMQ adalah salah satu pustaka untuk mengirim pesan menggunakan soket yang cukup terkenal. Berbicara mengenai pengiriman data, keamanan pesan merupakan bagian penting yang perlu diperhitungkan terutama apabila melakukan pengiriman data melalui jaringan. ZeroMQ mengirim pesan secara terbuka tanpa mengamankan pesan yang dikirim. Hal tersebut dibuktikan dengan penelitian yang menyebutkan bahwa ZeroMQ tidak memiliki lapisan keamanan untuk pengiriman pesan melalui jaringan dan observasi langsung pada paket pesan ZeroMQ menggunakan aplikasi Wireshark. Oleh karena itu, dibuatlah penelitian ini untuk membuat metode pengamanan dan otentikasi pengiriman pesan menggunakan enkripsi AES (*Advanced Encryption Standard*) Mode CBC (*Cipher Block Chaining*) yang dikombinasikan dengan metode otentikasi sederhana. AES Mode CBC dipilih karena metode ini lebih cepat daripada metode lain dan memiliki enkripsi yang kuat. Enkripsi dan otentikasi ini digunakan agar pengirim dan penerima pesan merupakan pengirim dan penerima yang valid sehingga tidak ada perubahan pesan selama pengiriman pesan dan pesan hanya dapat dibuka oleh penerima pesan dan pengirim pesan. Pengujian dilakukan untuk mengukur pengaruh enkripsi dan otentikasi terhadap performa pengiriman pesan. Berdasarkan pengujian yang dilakukan menunjukkan rata-rata peningkatan waktu pengiriman yaitu 1,5 detik ketika ditambahkan enkripsi dan otentikasi.

Kata kunci: enkripsi, parallel pipeline, otentikasi, soket, zeromq

1. Pendahuluan

Komunikasi jaringan kini sudah tidak asing lagi. Berbagai jenis aplikasi mulai menggunakannya untuk menjalin komunikasi atau mengirim pesan antar aplikasi yang dimiliki. Teknologi ini menggunakan soket untuk mengirim pesan [1]. Soket dapat diartikan sebagai titik akhir dalam saluran komunikasi di komputer. Secara umum, setiap komputer memiliki soket yang diikat ke nomor *port* tertentu. Soket memungkinkan suatu proses untuk berkomunikasi dengan proses lain pada mesin yang sama atau bahkan berbeda. Soket juga memungkinkan komunikasi antar proses yang memiliki bahasa pemrograman berbeda. Soket membuat saluran komunikasi yang dapat digunakan untuk bertukar data antara program atau aplikasi secara lokal atau lintas jaringan.

Diterima Redaksi : 19-10-2020 | Selesai Revisi : 18-11-2020 | Diterbitkan Online : 20-12-2020

Saat ini, banyak pustaka-pustaka yang membantu dalam bertukar pesan atau informasi menggunakan soket. Pustaka ini dapat memaksimalkan potensi soket sehingga dapat digunakan untuk berbagai macam kebutuhan [2][3]. Salah satunya adalah ZeroMQ. ZeroMQ (atau disebut juga ØMQ, 0MQ atau ZMQ) adalah pustaka perpesanan yang bersifat *asynchronous* dan memiliki performa tinggi. ZeroMQ mendukung pola pesan umum antara lain *publish-subscribe*, *request-repry*, dan salah satu pola pengiriman terbaiknya yaitu *parallel pipeline*. *Parallel pipeline* adalah salah satu pola pengiriman pesan yang handal milik ZeroMQ. Pola pengiriman ini memiliki konsep dimana ZeroMQ memproses pesan secara *parallel* menggunakan beberapa *worker* sehingga pesan dapat diselesaikan dengan cepat dan terdistribusi. ZeroMQ menggunakan berbagai jenis media pengiriman seperti TCP, dalam proses, antar proses, *multicast* dan *WebSocket*.

ZeroMQ dapat digunakan untuk bertukar pesan atau informasi melalui jaringan. Sayangnya, secara umum, ZeroMQ tidak melindungi pesan yang dikirim melalui jaringan. Penelitian yang dilakukan oleh Rajcan menunjukkan bahwa ZeroMQ tidak memiliki lapisan keamanan dalam mengirim pesan melalui jaringan sehingga perlu menambahkan sistem keamanan untuk memastikan pesan yang dikirim aman sampai tujuannya [4]. Fakta ini juga dibuktikan dengan teridentifikasinya pesan tanpa pengamanan yang dikirim oleh ZeroMQ menggunakan aplikasi Wireshark. Kekurangan tersebut tentunya menimbulkan adanya kerentanan pada pesan yang dikirim terutama terhadap jenis serangan *Man in the Middle Attack*.

Advanced Encryption Standard atau disingkat AES adalah salah satu algoritma enkripsi yang memiliki kecepatan tinggi dan dirancang untuk keamanan data [5][6][7]. AES merupakan sistem penyandian yang bersifat non-feistel menggunakan komponen yang selalu memiliki *invers* dengan Panjang blok 128-bit. Kunci AES menggunakan proses yang berulang yang disebut dengan ronde.

Otentikasi adalah sebuah proses mengidentifikasi sebuah permintaan untuk memastikan apakah permintaan tersebut dapat dilanjutkan ke proses selanjutnya atau tidak [10]. Proses otentikasi selalu berada diawal sebuah aplikasi. Dalam penelitian ini, otentikasi diperlukan untuk memastikan pesan yang diterima berasal dari pengirim yang valid dan memastikan bahwa pesan yang diterima juga valid. Proses otentikasi dilakukan setelah proses dekripsi selesai. Pesan yang dianggap tidak valid akan diabaikan, sebaliknya pesan yang dianggap valid akan diproses sesuai dengan fungsionalitas yang ada.

Telah banyak penelitian yang dilakukan terkait keamanan data pada pengiriman data melalui jaringan. Purwinarko & Hardyanto telah mengembangkan mekanisme enkripsi dan otentikasi dalam aplikasi seluler [11]. Metode enkripsi yang digunakan adalah AES yang digabungkan dengan Blowfish. Mekanisme tersebut diimplementasikan dalam aplikasi seluler yang melakukan proses otentikasi ke *server* di internet. Sistem dikembangkan dengan menggunakan bahasa pemrograman Java dari Sun Microsystem. Nama pengguna dan kata sandi yang dimasukkan oleh pengguna aplikasi ini akan dienkripsi menggunakan metode AES. Kunci untuk mengenkripsi *username* dan *password* merupakan kunci asli yang telah dienkripsi menggunakan metode Blowfish. Hasil *ciphertext* dari proses enkripsi dikirim ke *server* untuk proses dekripsi selanjutnya.

Penelitian yang dilakukan Laksh membangun sebuah sistem pengamanan pada *Bilateral Teleportation System*. *Bilateral Teleportation System* merupakan sistem yang mengkolaborasikan teknologi robot dengan jaringan sensor yang digunakan untuk melakukan operasi jarak jauh. Salah satu hal yang dapat digunakan dengan sistem ini adalah operasi medis jarak jauh. Data penting dikirim ke sistem dan modifikasi kecil bisa berakibat fatal.

Ravikumara & Lakshmikanth mengembangkan mekanisme untuk mengirim dan mengenkripsi data pada sistem [12]. Algoritma Diffie-Hellman digunakan untuk mendistribusikan kunci ke tetangganya. Sistem ini menggunakan Protokol Enkripsi Hop-By-Hop dimana data akan dienkripsi selama ditransmisikan melalui jaringan.

[13] membuat metode baru yang dapat digunakan untuk mengenkripsi dan menyembunyikan data yang dikirim melalui jaringan. Metode tersebut adalah CryptoSteganography, metode yang menggabungkan kriptografi dengan steganografi. Metode ini akan mengenkripsi data dan kemudian menyembunyikannya dalam sebuah gambar dan mengirimkannya ke penerima pesan. Metode ini menggunakan AES untuk melakukan proses enkripsi dari sisi kriptografi dan dari sisi steganografi menggunakan LSB untuk menyembunyikan data terenkripsi. MSE dan PSNR juga digunakan untuk menjaga kualitas gambar.

Berdasarkan uraian di atas, dapat diketahui bahwa keamanan data pada pengiriman data melalui jaringan sangatlah penting untuk dilakukan terutama pada sistem-sistem yang mengandalkan integritas data. Penelitian yang telah dilakukan sebelumnya telah berhasil melakukan enkripsi dan dekripsi menggunakan berbagai jenis algoritma, namun penelitian tersebut tidak menjamin data yang dikirim diterima oleh penerima yang tepat. Melihat kekurangan tersebut penelitian ini menggabungkan metode enkripsi data dan otentikasi data menggunakan Advanced Encryption Standard (AES) dan metode otentikasi yang penulis kembangkan sendiri. ZeroMQ yang berkinerja tinggi dikombinasikan dengan teknik enkripsi data AES dan teknik otentikasi data dapat meningkatkan keamanan data yang dikirim

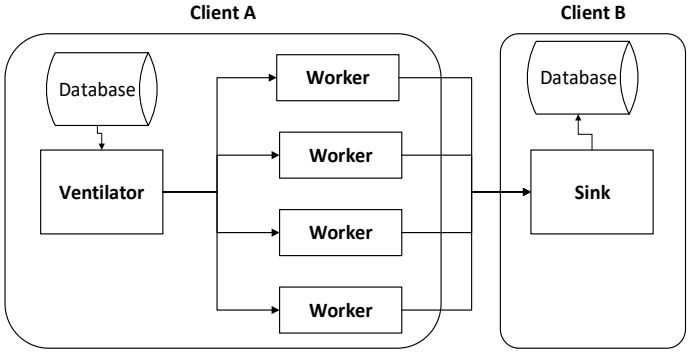
melalui jaringan. AES digunakan sebagai metode enkripsi untuk mengamankan data dan otentikasi digunakan untuk memastikan pesan yang dikirim berasal dari pengirim yang valid dan penerima yang valid. AES Mode CBC dipilih karena memiliki kelebihan yaitu setiap *cipher block* digunakan untuk mengenkripsi *cipher* berikutnya sehingga enkripsi lebih sulit dipecahkan dan memiliki performa yang tinggi dibandingkan dengan metode enkripsi lainnya [8][14].

Penelitian ini berfokus pada pembuatan skema enkripsi dan otentikasi untuk memastikan pesan yang dikirim melalui ZeroMQ *parallel pipeline* dapat diproteksi. Penelitian ini juga memastikan bahwa skema yang dibuat tidak terlalu mempengaruhi kinerja ZeroMQ *parallel pipeline* dengan melakukan pengujian.

2. Metode Penelitian

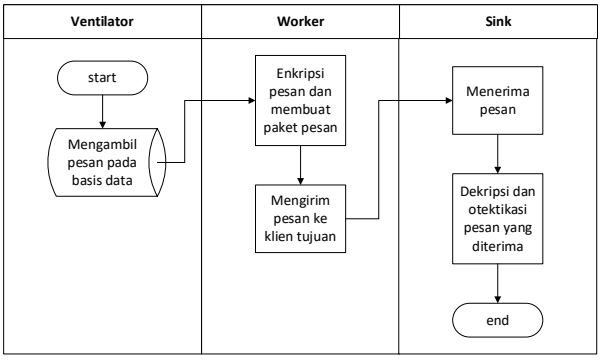
Penelitian ini bertujuan untuk menghasilkan lapisan keamanan baru pada sistem pengiriman pesan ZeroMQ *parallel pipeline* yang cepat dan ringan.

ZeroMQ *parallel pipeline* terdiri dari tiga bagian yaitu *Ventilator*, *Worker* dan *Sink*. *Ventilator* bertugas menyelesaikan tugas. *Worker* berfungsi untuk memproses tugas yang dibuat oleh *Ventilator*. *Sink* bertanggung jawab untuk mengumpulkan hasil pemrosesan dari *Worker* [15]. Gambaran umum sistem pada penelitian ini ditunjukkan pada Gambar 1.



Gambar 1. Gambaran Umum Sistem

Pengiriman pesan dari *Client A* ke *Client B* seperti pada Gambar 1 menunjukkan bahwa pada sisi pengirim terdapat dua bagian yaitu *Ventilator* dan *Worker*. *Ventilator* akan mengambil daftar pesan yang akan dikirim ke tujuannya di *database*. Pesan yang disimpan dalam *database* menyimpan setidaknya beberapa kolom.



Gambar 2. Tugas Masing-Masing Bagian pada ZeroMQ Parallel Pipeline

Gambar 2 menunjukkan tugas dari masing-masing bagian dari ZeroMQ *parallel pipeline*. *Ventilator* bertugas untuk mengambil pesan yang akan dikirim ke basis data, kemudian pesan yang akan dikirim tersebut akan diberikan kepada *Worker* untuk diproses. *Worker* akan melakukan proses enkripsi pesan lalu membuat paket pesan yang diap dikirim ke tujuannya. *Sink* akan menerima paket pesan yang dikirim oleh *Worker* dan memulai proses dekripsi dan otentikasi.

ZeroMQ dapat mengirim berbagai jenis pesan seperti JSON, *Buffer* dan *String*, namun pada penelitian ini jenis data pesan yang digunakan adalah *String* sehingga data apapun yang akan dikirim harus diubah terlebih dahulu kedalam bentuk *String*. Data yang akan dikirim tersebut disimpan dalam basis data pada tabel pesan.

Penelitian ini menggunakan data *dummy* atau buatan yang berbentuk kalimat sebanyak sepuluh jenis kalimat. Kalimat tersebut akan secara acak dipilih untuk dienkripsi dan dikirim ke tujuan untuk pengujian. Tabel 1 menunjukkan data kalimat yang digunakan.

Tabel 1.Daftar Kalimat sebagai Data Penelitian

No	Kalimat
1	Berikut adalah kalimat rahasia
2	Kode rahasia 12345
3	Pesan rahasia jangan dibuka
4	Kode kunci pintu utama adalah 1234
5	Pesan rahasia untuk kepala divisi marketing
6	Berikut adalah pesan untuk kepala
7	Email terkirim ke tujuan
8	Server dihidupkan
9	Chmod 755
10	Exec sudo apt-get install node

Kalimat pada Tabel 1 digunakan sebagai data pesan yang akan dikirim, Kalimat tersebut bersama tujuan penerimanya akan dimasukkan ke tabel pesan.

Tabel 2.Kolom-Kolom Tabel Pesan

Nama Kolom	Keterangan
msg_id	*Primary key*
client_id	ID penerima pesan
data	Pesan yang akan dikirim
is_sent	Tanda status pengiriman pesan

Semua pesan yang akan dikirim oleh *Ventilator* disimpan dalam tabel yang ditunjukkan pada Tabel 2. *Ventilator* akan lebih mudah mengambil data yang belum terkirim dan akan lebih mudah untuk menghasilkan pesan. Sistem ini juga membutuhkan tabel yang menyimpan daftar klien yang dapat diajak berkomunikasi. Tabel tersebut berfungsi untuk menyimpan data klien yang terkait dengan proses otentikasi dan enkripsi pesan. Kolom tabel klien ditunjukkan pada Tabel 3.

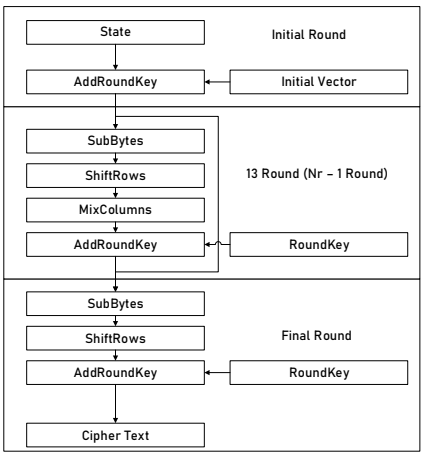
Tabel 3. Kolom-Kolom Tabel Klien

Nama Kolom	Keterangan
client_id	*Primary key*
client_key	16 bytes *secret key* untuk proses enkripsi dan dekripsi pesan
client_iv	16 bytes *initial vector* proses enkripsi dan dekripsi pesan
client_ip	Alamat IP klien

Tabel klien terkait dengan tabel pesan. *Ventilator* membaca pesan tabel dan pesan klien yang tidak terkirim berdasarkan kolom *is_sent*. Setiap pesan akan dikirim ke *Worker* untuk proses enkripsi selanjutnya dan dikirimkan ke tujuan. *Worker* menunggu pesan dari *Ventilator* dengan mendengarkannya pada nomor *port* tertentu sesuai dengan konfigurasi yang dilakukan. Jumlah *Worker* yang berjalan bisa berbeda-beda tergantung seberapa cepat pengiriman harus dilakukan. Semakin banyak *Worker* yang berjalan maka akan semakin cepat proses yang dilakukan [8].

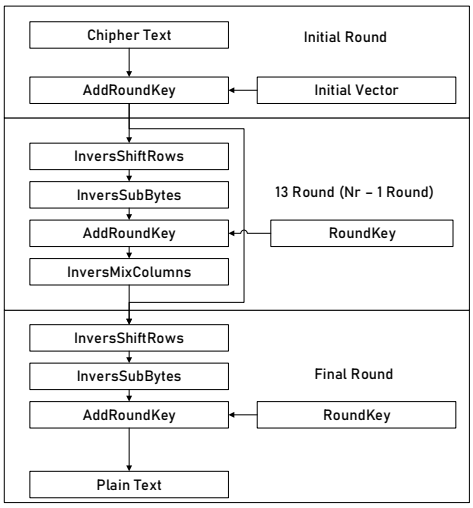
Proses enkripsi dilakukan oleh *Worker* karena dalam konsep *parallel pipeline* tugas *Ventilator* hanya membagi tugas [8]. Metode enkripsi yang digunakan adalah AES atau *Advanced Encryption Standard* yang menggunakan mode operasi CBC atau *Cipher Block Chaining*. Enkripsi ini menggunakan kunci enkripsi 16byte dan 16byte *initial vector*. Kunci enkripsi 16byte digunakan untuk menghasilkan enkripsi 256bit sehingga keamanan data lebih terjamin.

Proses enkripsi algoritma AES terdiri dari 4 jenis transformasi bytes yaitu *SubBytes*, *ShiftRows*, *MixColumns*, dan *AddRoundKey* [8][9]. proses enkripsi diawali dengan input yang telah ditambahkan ke dalam *state* akan mengalami transformasi byte *AddRoundKey*. *State* akan mengalami transformasi *SubBytes*, *ShiftRows*, *MixColumns*, dan *AddRoundKey* secara berulang-ulang sebanyak *Nr*. Proses ini dalam algoritma AES disebut sebagai *round function*. *Round* yang terakhir agak berbeda dengan *round-round* sebelumnya dimana pada *round* terakhir, *state* tidak mengalami transformasi *MixColumns*.



Gambar 3. Proses Enkripsi AES

Gambar 3 adalah alur proses enkripsi pada AES Mode CBC. Setiap proses enkripsi dibagi menjadi beberapa *round* tergantung panjang kunci yang digunakan yaitu 10 *round* untuk 128 bit, 12 *round* untuk 192 bit dan 14 *round* untuk 256 bit kunci. Proses pada *round* pertama adalah proses *AddRoundKey* dengan menambahkan *Initial Vector* sebagai kuncinya. *Round* selanjutnya adalah proses *SubBytes*, *ShiftRows*, *MixColumns* dan *AddRoundKeys*. Proses pada *round* terakhir antara lain *SubBytes*, *ShiftRows* dan *AddRoundKey* sehingga menghasilkan *Cipher Text*. Proses dekripsi dari data yang telah di enkripsi menggunakan AES Mode CBC ditunjukkan pada Gambar 4.



Gambar 4. Proses Dekripsi AES

Proses dekripsi pada Enkripsi AES adalah kebalikan dari proses enkripsi. Jumlah *round* yang diperlukan juga sama seperti jumlah *round* pada proses enrkripsi. Proses pada *round* pertama adalah proses *AddRoundKey*. Proses *round* selanjutnya adalah *InversShiftRows*,

InversSubBytes, *AddRoundKey* dan *InversMixColumns*. Proses pada *round* terakhir adalah *InversShiftRows*, *InversSubBytes*, *AddRoundKey* dan akan menghasilkan *plain text*.

AES Mode CBC menggunakan kunci enkripsi simetris yang artinya kunci tersebut digunakan dalam proses enkripsi dan dekripsi [12]. Tantangannya adalah mengamankan kunci tersebut sehingga data yang dienkripsi tidak dapat didekripsi dengan mudah. Kunci enkripsi dan *initial vector* setiap *host* tidak didistribusikan selama komunikasi, tetapi kunci enkripsi dan *initial vector* disimpan di setiap *host* dalam bentuk tabel di *database*. Kunci enkripsi dan vektor awal dari *host* disimpan di *hardcode* pada sistem sedangkan kunci enkripsi dan vektor awal dari *host* lain yang bertindak sebagai penerima pesan disimpan dalam *database*.

Setiap blok teks biasa dalam mode operasi CBC dilakukan proses XOR dengan hasil *ciphertext* dari blok sebelumnya. Setiap *ciphertext* dari setiap blok bergantung pada blok *ciphertext* sebelumnya. *Initial vector* digunakan untuk membuat blok XOR pertama sehingga unik. Kesalahan pengkodean dalam satu blok teks biasa akan menyebar ke blok teks tersandi berikutnya. Mode ini dapat menghasilkan *ciphertext* yang berbeda sehingga kriptanalisis tidak dapat dilakukan dengan mudah.

Pesan yang dikirim *Worker* menggunakan format JSON dengan dua atribut yaitu *sender_id* dan *data*. Atribut *sender_id* adalah kode unik yang dimiliki setiap *host* untuk mengidentifikasi pengirim pesan. Atribut *data* adalah isi pesan yang akan dikirim. Isi atribut data ini dienkripsi menggunakan AES Mode CBC.

Worker yang sudah mendapatkan tugas dari *Ventilator* kemudian membuat paket pesan berupa JSON. Format pesan yang dibuat oleh Worker ditunjukkan pada Gambar 5.

```
{
  "sender_id": 12,
  "data": {
    "sender_id": 12,
    "data": "Hello this is secret message",
    "timestamp": "2019-10-12 12:13:00"
  }
}
```

Gambar 5. Paket Pesan Sebelum Terenkripsi

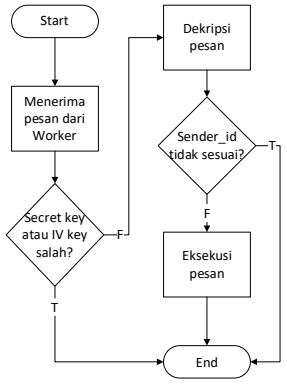
Gambar 5 menunjukkan format pesan yang belum dienkripsi oleh *Worker*. Atribut *sender_id* ditambahkan di atribut *data*. Penambahan ini dibuat untuk tujuan otentikasi pesan yang akan dijelaskan lebih lanjut pada bagian selanjutnya. Objek pada atribut *data* yaitu *sender_id*, *data* dan *timestamp* akan diubah menjadi text kemudian dilakukan proses enkripsi untuk menghasilkan paket yang siap dikirim seperti pada Gambar 6.


```
{
  "sender_id": 12,
  "data": "Rz2sYECCh7JfUy/x+qjAPto0RBvJw8jU
    /eOSeC6IVzLdN9QJkM1Y8MsyJIwy5oTOswiH1jZxWiJu9O55kaGwS6EUUHxAAE2CW3
    taC86hZtxZZdfklbFcgxch0c5MaRWs"
}
```


Gambar 6. Paket Pesan Setelah Terenkripsi

Objek yang ada di atribut *data* dienkripsi menjadi *string*. *Initial vector* atau kunci enkripsi seperti yang disebutkan sebelumnya tidak didistribusikan dalam pesan yang dikirim. *Worker* mengenkripsi atribut objek *data* menggunakan kunci enkripsi dan *initial vector* milik penerima pesan yang disimpan di tabel klien, sehingga ketika pesan sampai ke penerima, penerima dapat melakukan proses dekripsi menggunakan kunci enkripsi dan *initial vector* miliknya. Metode ini juga dapat memastikan bahwa tidak ada yang dapat mendekripsi pesan kecuali penerima dan pengirim karena memerlukan dua parameter untuk mendekripsi kunci enkripsi dan initial vector yang hanya dimiliki oleh kedua *host* tersebut.

Otentikasi diperlukan untuk memastikan pesan yang diterima berasal dari pengirim yang valid dan memastikan bahwa pesan yang diterima juga valid. Setiap pesan yang dikirim akan membawa identitas klien pengirim pesan. Identitas klien pengirim pesan tersebut disimpan ada dua tempat berbeda yaitu didalam data yang terenkripsi dan diatribut lain berdampingan dengan data yang terenkripsi. Proses otentikasi akan membandingkan identitas pengirim yang terdapat pada data yang terenkripsi dan pada atribut diluar data yang terenkripsi tersebut, apabila terjadi ketidakcocokan maka pesan tersebut dianggap tidak valid. Proses otentikasi menggunakan *sender_id* dalam paket pesan, kunci enkripsi dan *initial vector* dari pesan penerima. Alur proses otentikasi ditunjukkan pada Gambar 7.



Gambar 7. Paket Pesan Setelah Terenkripsi

Pertama, sink mendekripsi pesan masuk. Proses dekripsi dilakukan menggunakan kunci enkripsi penerima dan kunci *initial vector*. Kunci Enkripsi atau *initial vector* yang salah akan menyebabkan proses dekripsi gagal sehingga dapat dikategorikan pesan tersebut tidak valid. Pesan yang valid akan berisi *sender_id*, *data*, dan *timestamp*. Isi *sender_id* akan dibandingkan dengan *sender_id* yang berada di luar atribut *data*. Proses ini

dilakukan untuk memastikan tidak ada perubahan data selama proses pengiriman dan memvalidasi pengirim pesan. Pesan yang sudah dianggap valid kemudian bisa diproses sesuai dengan fungsionalitas yang ada.

3. Hasil dan Pembahasan

Tahap terakhir dalam penelitian ini adalah pengujian. Pengujian yang dilakukan adalah dengan membandingkan kinerja pengiriman pesan tanpa enkripsi dan otentikasi dengan kinerja pengiriman pesan dengan enkripsi dan otentikasi. Selama pengiriman pesan, perhitungan kehandalan sistem juga dilakukan dengan membandingkan jumlah pesan yang dikirim dengan pesan yang diterima oleh *host* tujuan.


```
{
  "sender_id": 10,
  "data": {
    "sender_id": 10,
    "data": "Kode kunci pintu utama adalah 1234",
    "timestamp": "2019-10-12 12:13:00"
  }
}
```

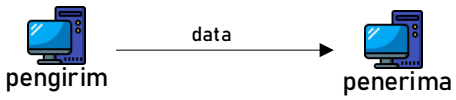

Gambar 8. Sampel Data Pengunjian Sebelum Terenkripsi

Gambar 8 adalah contoh salah satu dari sepuluh pesan yang diacak dan dikirim ke penerima yang telah ditentukan. Data yang akan dikirim adalah “Kode kunci pintu utama adalah 1234”. Data tersebut akan dienkripsi bersama dengan dua properti lain yaitu sender_id dan timestamp dengan menggunakan secret key dan initial vector dari penerima pesan.


```
{
  "sender_id": 10,
  "data": "sBDwhBjP5vpQ6nBpXVCNjr51Xpj8OPkoMiiFKmMunklyO1vB3x0Ogd51RmhxjGC0Otk4mcDmJmdYiZF62
    46k++KbAnPW5dp2NeLwemzfRso46WWO7AML05NqQZhR9DradbNOQOBKYKiYx6ncGd7F1+p
    +hTgGwQLlwiKnl4VmQ74="
}
```


Gambar 9. Sampel Data Pengujian Setelah Dienkripsi

Hasil dari enkripsi pesan pada Gambar 8 ditunjukkan pada Gambar 9. Hasil enkripsi tersebut siap dikirim ke penerima dengan skema yang ditunjukkan pada Gambar 10.



Gambar 10. Gambaran Pengujian Sistem

Pengiriman pesan melalui jaringan membutuhkan waktu yang cepat sehingga membutuhkan proses enkripsi dan otentikasi yang cepat pula. Gambar 10 menunjukkan visualisasi pengujian yang dilakukan dimana *host* pengirim akan mengirim data dengan format sesuai dengan Gambar 9 dengan data uji yang dipilih secara acak kepada *host* penerima. Pengujian dilakukan dengan mengirimkan data sebanyak 100, 300, 500, 700, 900 dan 1100 baris menggunakan dua buah *Worker*. Data yang dikirim dibuat dalam bentuk *string* acak yang dibuat menggunakan sebuah fungsi dengan panjang data yaitu 100 karakter. Format data yang dikirim sama seperti pada Gambar 9.

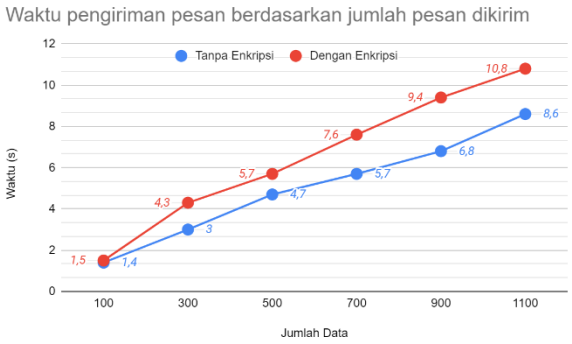
$$\Delta t = ts - tr \quad (1)$$

Perhitungan waktu pengiriman pesan menggunakan rumus yang ditunjukkan pada Rumus 1. Δt adalah waktu pengiriman pesan yang didapat dari pengurangan atau selisih antara ts atau waktu pesan dikirim dan tr atau waktu pesan diterima. Hasil pengujian ditunjukkan pada Tabel 4.

Tabel 4. Hasil Pengujian Performa Sistem

Jumlah Pesan Dikirim	Waktu Pengiriman (s)		Peningkatan Waktu (s)
	Tanpa Enkripsi	Dengan Enkripsi	
100	1,4	1,5	0,1
300	3	4,3	1,3
500	4,7	5,7	1
700	5,7	7,6	1,9
900	6,8	9,4	2,6
1100	8,6	10,8	2,2
	Rata-rata peningkatan waktu		1,5

Tabel 4 adalah hasil pengujian performa sistem. Tabel tersebut menunjukkan bahwa, ketika dilakukan pengiriman data sebanyak 100 data, sistem membutuhkan waktu selama 1,4 detik pada mode tanpa pengamanan dan 1,5 detik pada mode pengamanan begitu juga ketika dilakukan pengiriman data sebanyak 300 data, waktu yang dibutuhkan pada mode tanpa pengamanan adalah 3 detik dan 4,3 detik menggunakan mode pengamanan dan seterusnya. Rata-rata peningkatan waktu yaitu 1,5 detik.



Gambar 11. Visualisasi Performa Pengiriman Pesan Berdasarkan Jumah Pesan dan Waktunya

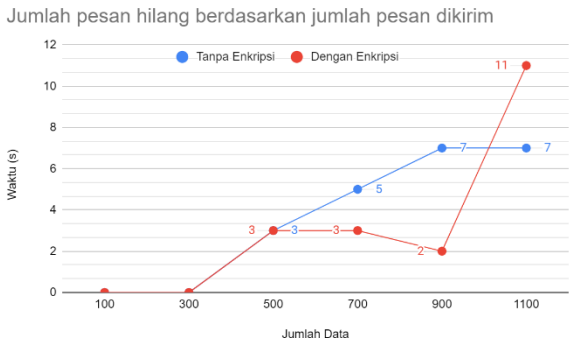
Gambar 11 adalah grafik hasil pengujian yang dilakukan. Hasil pengujian menunjukkan bahwa penambahan enkripsi dan otentikasi pada pengiriman data memiliki pengaruh yang relatif kecil pada waktu pengirimannya. Kondisi jaringan dan adanya masalah sindrom sambungan lambat pada *Worker* menyebabkan *Worker* membutuhkan waktu terlalu lama untuk mengambil proses baru dan mengirimnya [15]. Kedua faktor tersebut mempengaruhi waktu pengiriman pesan sehingga waktu pengiriman pesan menjadi tidak konsisten.

Pengujian lain juga dilakukan untuk mengukur seberapa handal pola pengiriman dengan *parallel pipeline* ini. Pengujian kehandalan dilakukan dengan menghitung jumlah pesan yang sampai ke *host* penerima. Pengujian ini juga dilakukan dengan mengirim 100, 300, 500, 700, 900 dan 1100 data dari *host* pengirim dan menghitung pesan yang diterima oleh *host* penerima. Hasil kahandalan sistem ditunjukkan pada Tabel 5.

Tabel 5. Hasil Pengujian Kehandalan Sistem

	Jumlah Pesan Dikirim					
	100	300	500	700	900	1100
Jumlah Pesan diterima (Tanpa Enkripsi)	100	300	497	695	893	1093
Jumlah Pesan hilang (Tanpa Enkripsi)	0	0	3	5	7	7
Jumlah Pesan Diterima (Dengan Enkripsi)	100	300	497	697	898	1089
Jumlah Pesan hilang (Dengan Enkripsi)	0	0	3	3	2	11

Hasil pengujian yang ditunjukkan pada Tabel 5 menunjukkan bahwa tidak terjadi kehilangan data pada pengiriman data sebanyak 100 dan 300 data. Kehilangan data baru terjadi ketika pengiriman 500 data dan seterusnya. Kasus tersebut terjadi pada pengiriman pesan tanpa enkripsi dan otentikasi dan pengiriman pesan dengan enkripsi dan otentikasi. Hal ini terjadi karena faktor kegagalan jaringan dan secara teknis ZeroMQ dengan pola pengiriman pesan *parallel pipeline* tidak dapat menangani hal tersebut. Faktor eksternal juga dapat menyebabkan masalah tersebut antara lain terjadi karena kegagalan jaringan pada saat pengiriman pesan atau terkait masalah sumber daya. Masalah ini dapat diatasi dengan menggunakan pola komunikasi *Request-Reply* [16]. Visualisasi kehilangan data lebih jelas ditunjukkan pada Gambar 12.



Gambar 12. Visualisasi Hasil Uji Reliabilitas Sistem

Gambar 12 dengan jelas menunjukkan bahwa antara model komunikasi yang menggunakan enkripsi dan tidak menggunakan enkripsi memiliki jumlah kehilangan data yang sangat mirip dan tidak menentu yang artinya enkripsi dan otentikasi yang ditambahkan tidak mempengaruhi kehandalan pengiriman pesan. Hanya saja semakin banyak data yang terkirim maka semakin tinggi pula tingkat kehilangan data yang akan terjadi.

4. Kesimpulan

Enkripsi dan otentikasi transmisi pesan di ZeroMQ menggunakan AES Mode CBC telah berhasil diselesaikan. Enkripsi dan otentikasi ini dapat digunakan sebagai metode yang efisien untuk mengamankan pesan karena enkripsi dan otentikasi secara signifikan tidak mempengaruhi waktu pengiriman pesan tergantung pada jumlah pesan yang dikirim, tetapi masalah kualitas jaringan dan masalah *Worker* dapat mempengaruhi waktu pengiriman pesan. Hasil pengujian menunjukkan rata-rata peningkatan waktu pengiriman yaitu 1,5 detik dengan peningkatan waktu terkecil yaitu 0,1 detik pada pengiriman 100 data dan peningkatan waktu tertinggi yaitu 2,6 detik pada pengiriman 900 data ketika ditambahkan enkripsi dan otentikasi. Selain itu ditemukan bahwa terjadi kehilangan data mulai pada pengiriman sebanyak 500 data yang disebabkan karena *network error* dan ketidakmampuan ZeroMQ *parallel pipeline* untuk menangani permasalahan tersebut.

Daftar Rujukan

[1] R. L. Maata, R. Cordova, B. Sudramurthy, and A. Halibas, "Design and Implementation of Client-Server Based Application Using Socket Programming in a Distributed Computing Environment," *2017 IEEE Int. Conf. Comput. Intell. Comput. Res. ICCIC 2017*, no. December, 2018.

[2] N. Ivaki, N. Laranjeiro, and F. Araujo, "A survey on reliable distributed communication," *J. Syst. Softw.*, vol. 137, no. October, pp. 713–732, 2018.

[3] L. Magnoni, "Modern messaging for distributed sytems," *J. Phys. Conf. Ser.*, vol. 608, no. 1, 2015.

[4] M. Rajcan, "Securing Data in Motion in ZeroMQ and Data at Rest on Windows Platform," 2017.

[5] A. Muhammad Abdullah, "Advanced Encryption Standard (AES) Algorithm to Encrypt and Decrypt Data," *Cryptogr. Netw. Secur.*, no. June, 2017.

[6] S. Patil and R. Patil, "Faster Transfer of AES Encrypted Data over Network," vol. 5, no. 6, pp. 7674–7676, 2014.

[7] S. Parmar and K. . Dave, "Implementation of Data Encryption and Decryption Algorithm for Information Security," *Int. J. Adv. Sci. Eng. Technol.*, vol. 1, no. 2, 2013.

[8] P. Mahajan and A. Sachdeva, "A Study of Encryption Algorithms AES, DES and RSA for Security," *Glob. J. Comput. Sci. Technol. Network, Web Secur.*, vol. 13, no. 15, 2013.

[9] Vaidehi and Rabi, "Design and Analysis of AES-CBC Mode for High Security Applications," *Second Int. Conf. Curr. Trends Eng. Technol.*, 2014.

[10] N. A. Lal, S. Prasad, and M. Farik, "A Review Of Authentication Methods," vol. 5, no. 11, pp. 246–249, 2016.

[11] A. Purwinarko and W. Hardyanto, "A Hybrid Security Algorithm AES and Blowfish for Authentication in Mobile Applications," *Sci. J. Informatics*, vol. 5, no. 1, p. 80, 2018.

[12] Ravikumara and Lakshmikanth, "Implementation of Hop-By-Hop Encryption Protocol for Transmission of Motion Control Data over Public Network Using Sors," *Int. J. Innov. Res. Adv. Eng.*, vol. 2, no. 4, pp. 221–224, 2015.

[13] A. F. Osuolale, "Secure Data Transfer Over the Internet Using Image CryptoSteganography," *Int. J. Sci. Eng. Res.*, vol. 8, no. 12, pp. 1115–1121, 2017.

[14] M. R. Joshi and R. A. Karkade, "Network Security with

Cryptography," *Int. J. Comput. Sci. Mob. Comput.*, vol. 41, no. 1, pp. 201–204, 2015.
[15] ZeroMQ, "ZGuide." [Online]. Available: http://zguide.zeromq.org/. [Accessed: 20-Aug-2020].
[16] D. Lombu, S. D. Tarihoran, and I. Gulo, "Kombinasi Mode Cipher Block Chaining Dengan Algoritma Triangle Chain Cipher Pada Penyandian Login Website," *J-SAKTI (Jurnal Sains Komput. dan Inform.*, vol. 2, no. 1, p. 1, 2018.