Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer e-ISSN: 2548-964X
Vol. 2, No. 10, Oktober 2018, hlm. 4027-4033 http://j-ptiik.ub.ac.id

Analisis Mekanisme Keamanan Antara TLS/SSL Dan Crypto Pada Komunikasi IoT Middleware Dengan Subscriber Berbasis Protokol HTTP

Deny Hari Prasetya Dewa[1], Eko Sakti Pramukantoro[2], Dany Primanita Kartikasari[3]

Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya
Email: [1]denyhari.pras@gmail.com, [2]ekosakti@ub.ac.id, [3]dany.jalin@ub.ac.id

Abstrak

Terdapat celah keamanan pada pengiriman pesan pada sebuah *middleware* dimana pesan yang dikirimkan masih dalam bentuk *plain text*, yang menyebabkan kerahasiaan data menjadi tidak terjamin. Aktifitas *eavesdropping* akan sangat mudah dilakukan jika tidak segera diimplementasikan sebuah mekanisme keamanan. Berdasarkan permasalahan tersebut, diimplementasikan mekanisme *end-to-end security* menggunakan mekanisme keamanan TLS/SSL dan *Crypto*. Penggunaan mekanisme keamanan tidak hanya mengamankan jaringan, tetapi juga mempengaruhi kinerja sistem tersebut. Proses enkripsi dan dekripsi memperlambat kinerja *middleware*. Penelitian ini berfokus untuk mengetahui bagaimana dampak implementasi mekanisme keamanan TLS/SSL dan *Crypto*, khususnya AES-256 terhadap keamanan data dan kinerja *middleware*. Sebelum mekanisme keamanan diimplementasikan, data yang terdapat pada jaringan sangat mudah dibaca dan diketahui isinya. Setelah kedua mekanisme keamanan diimplementasikan, data tidak dapat diketahui lagi isinya karena telah dienkripsi. TLS/SSL memberikan keamanan yang lebih kuat dengan menggunakan pertukaran sertifikat sebagai sistem otentikasi. Kedua metode yang dipasang mampu mengamankan sistem tetapi keduanya mempengaruhi kinerja *middleware*. Kedua mekanisme keamanan tidak berpengaruh besar terhadap penggunaan CPU dan *memory*. AES-256 menghasilkan nilai yang lebih baik pada parameter *packet loss*, dengan nilai 1%. Sedangkan TLS/SSL lebih unggul di parameter *delay* dan *jitter*, dengan nilai 0.089008423 detik dan 0.003208877 detik.

Kata kunci: *Internet of Things*, *middleware*, *TLS/SSL*, *AES-256*, keamanan, *delay*.

Abstract

There is a security hole in message delivery on a middleware where messages are still in plain text, causing data confidentiality to be insecure. Eavesdropping activities will be very easy to do if a security mechanism is not implemented immediately. Based on the problem, implemented end-to-end security mechanism using TLS / SSL and Crypto. The use of security mechanisms not only secures the network, but also affects the performance of the system. The process of encryption and decryption slows down middleware performance. This study focuses on knowing how the impact of TLS / SSL and Crypto security mechanisms, especially AES-256 on data security and middleware performance. Before the security mechanism is implemented, the data contained on the network is very readable. After both security mechanisms are implemented, the data cannot be known again because the contents have been encrypted. TLS / SSL provides stronger security by using certificate exchanges as an authentication system. Both methods installed are capable of securing the system but both affect the performance of middleware. Both security mechanisms have no significant effect on CPU and memory usage. AES-256 produces better values on packet loss parameters, with a value of 1%. While TLS / SSL is superior in parameter delay and jitter, with value 0.089008423 sec and 0.003208877 sec.

Keywords: *Internet of Things*, *middleware*, *TLS/SSL*, *AES-256*, *security*, *delay*.

1. PENDAHULUAN

Pada penelitian sebelumnya telah dikembangkan sebuah *middleware* yang mampu mendukung interoperabilitas berbagai perangkat atau sensor (Anwari, 2017). Pada penelitian yang lain, telah dikembangkan *data storage framework* untuk *middleware* yang sama

(Pramukantoro, 2017). Pada penelitian-penelitian tersebut, *middleware* dapat melakukan komunikasi *end-to-end* dengan aplikasi menggunakan *subscriber* berbasis protokol HTTP. Dalam skema komunkasi tersebut, terdapat celah keamanan dimana paket data yang dikirimkan masih berupa *plain text*. Celah keamanan ini memberikan ancaman berupa *Eavesdropping*.

Rajra dan J Deepa (2015) menjelaskan bahwa *eavesdropping* adalah penangkapan komunikasi antara dua titik oleh pihak yang tidak berwenang. *Passive eavesdropping* adalah ketika seseorang hanya menangkap dan mendengarkan pesan di dalam komunikasi jaringan secara diam-diam. Sedangkan *active eavesdropping* adalah ketika penyerang tidak hanya mendengarkan pesan, tetapi juga menyusupkan pesan ke dalam sistem komunikasi. Hal ini dapat menyebabkan pesan menjadi kacau dan informasi yang sensitif dapat dicuri.

Dibutuhkan mekanisme keamanan untuk mengatasi celah keamanan pada komunikasi *middleware* tersebut, mekanisme keamanan tersebut adalah *end-to-end security*. Dalam end-to-end security terdapat dua metode yang dapat digunakan, yaitu TLS/SSL dan Crypto. TLS/SSL merupakan protokol yang paling banyak digunakan untuk memastikan autentikasi, integrasi dan *confidentiality* pada pertukaran informasi antara web server dan web client (Turner, 2014). Kriptografi adalah seni penulisan rahasia yang digunakan sejak zaman Romawi untuk menyembunyikan informasi rahasia atau menjaga keamanan pesan. Untuk menjaga kerahasiaan informasi, metode yang banyak digunakan adalah enkripsi / dekripsi (Maqsood F, 2017). Maqsood F (2017), dalam penelitiannya membandingkan kinerja dari setiap algoritma berdasarkan waktu enkripsi dan dekripsi, penelitian tersebut menghasilkan data yang menunjukkan bahwa waktu enkripsi dan dekripsi algoritma AES adalah yang paling cepat. Oleh karena itu, penulis menggunakan algoritma AES-256 untuk penelitian ini. Cara kerja AES adalah dengan mengulang langkah-langkah yang sama yang sudah ditetapkan berkali-kali, AES adalah semacam algoritma enkripsi untuk kunci rahasia dan beroperasi pada sejumlah *byte* yang tetap (Bhajaj, 2016).

Sebelum melakukan implementasi mekanisme keamanan ke dalam *middleware*, terdapat beberapa hal yang harus diperhatikan. Mudassar Ahmad (2012), dalam penelitiannya membuktikan bahwa penggunakaan mekanisme keamanan akan mengurangi kinerja jaringan. Kinerja adalah permasalahan yang utama pada implementasi mekanisme keamanan. Terdapat beberapa parameter jaringan yang harus diperhatikan untuk melihat kinerja jaringan, seperti tingkat pengiriman, *throughput*, *delay*, probabilitas tabrakan, efisiensi *bandwidth*, rasio *packet loss*, *bit error rate*, *delay* antrian, dan *jitter* (Islam N, 2016).

Oleh karena itu, dibutuhkan sebuah studi tentang penerapan *end-to-end security*, agar dapat mengetahui mekanisme keamanan apa yang dapat mengamankan *middleware* IoT dan mekanisme mana yang lebih efisen jika digunakan. Berdasarkan penjelasan di atas, penulis ingin menganalisis kinerja dan pengaruh dari mekanisme keamanan TLS/SSL dan AES-256 pada middleware IoT. Dengan analisis ini, diharapkan dapat mengetahui bagaimana kinerja mekanisme keamanan TLS/SSL dan AES-256 dalam mengamankan komunikasi dan bagaimana pengaruh mekanisme keamanan tersebut terhadap kinerja komunikasi *middleware*.

2. LANDASAN KEPUSTAKAAN

2.1. Kajian Pustaka

Terdapat sebuah penelitian sebelumnya tentang topik yang terkait dengan penelitian ini yang berjudul *Implementasi Advanced Encryption (AES) Pada Sistem Kunci Elektronik Kendaraan Berbasis Sistem Operasi Android dan Mikrokontroler Arduino* (Ramdhansya, Ariyanto, & Nuha, 2014). Pada penelitian tersebut telah diimplementasikan algoritma AES untuk mengenkripsi dan mendekripsi pesan antara mikrokontroler Arduino dengan sistem operasi *mobile* Android.

Terdapat penelitian lain yang berjudul *Analisis Perbandingan Mekanisme Secure Socket Layer (SSL) dan Transfer Layer Security (TLS) Pada Koneksi File Transfer Protocol (FTP) Server Ubuntu* yang menganalisa mekanisme yang terjadi saat mekanisme keamanan *SSL* dan *TLS* mengamankan server FTP (Tehupeiory & Chandra, 2016).

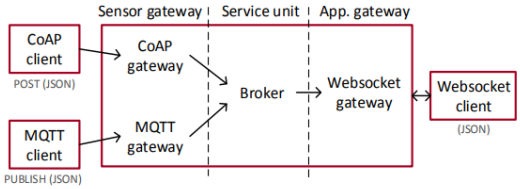
Penelitian terkait yang terakhir berjudul Topic *based IoT data storage framework for heterogeneous sensor data* (Pramukantoro, 2017). Penelitian tersebut membuat *framework* untuk penyimpanan data pada *middleware* yang sama dengan penelitian ini, menggunakan

MongoDB dan GridFS.

Berdasarkan penelitian di atas, penelitian ini dilakukan untuk menganalisa mekanisme keamanan *TLS/SSL* dan *AES-256* ketika diimplementasikan ke sistem *middleware*, dan untuk mengetahui bagaimana pengaruh keduanya terhadap kinerja komunikasi *middleware*.

2.2. MIDDLEWARE

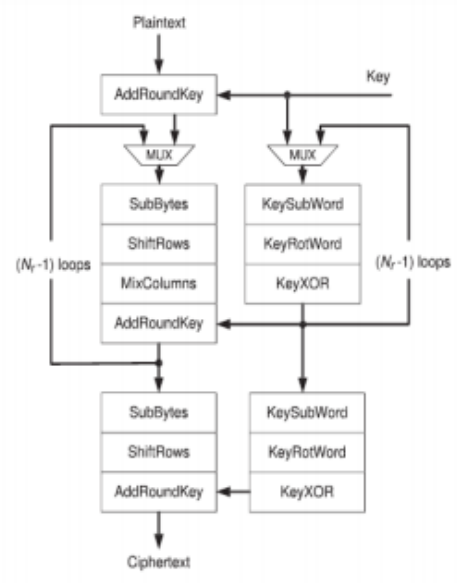
Tujuan utama dari penelitian ini adalah menyediakan mekanisme keamanan pada komunikasi *end-to-end* antara *middleware* dengan *subscriber* pada protokol HTTP untuk *IoT Middleware*. *Middleware* yang digunakan pada penelitian ini adalah middleware yang dikembangkan oleh Husnul Anwari dalam penelitiannya (Anwari, 2017). Arsitektur dari *middleware* tersebut ditunjukkan dalam Gambar 1.



Gambar 1. Arsitektur *Middleware*

2.3. Advanced Encryption Standard (AES)

AES mengambil input sebagai blok data berukuran 128-bit dan melakukan putaran transformasi untuk menghasilkan cipher text sebagai output. Blok input data diproses oleh *array byte* 4-by-4. Ukuran kunci putaran bisa berukuran 128, 192 atau 256 bits (Bhajaj, 2016). Panjang kunci putaran akan menentukan jumlah putaran yang diulang pada AES. Kunci putarannya adalah 10, 12 atau 14 untuk masing-masing panjang kunci 128, 192 atau 256 bits (Bhajaj, 2016). Gambar 2 menunjukkan diagram block dari algoritma AES.



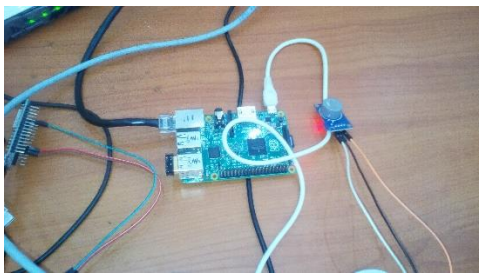
Gambar 2. Diagram block algoritma AES

2.4 TLS/SSL

Protokol *TLS/SSL* memiliki dua bagian, yang pertama adalah *handshaking protocol*, yang kedua adalah *record protocol* . *Handshaking protocol* menegoisasi *suite cipher*, mengotentikasi *server* dan secara opsional mengotentikasi klien dan menetapkan *session keys* (Turner, 2014). Sedangkan *record protocol* mengamankan data aplikasi dengan *session keys* yang dibuat pada *record protocol* dan memverifikasi keaslian dan integritas aplikasi (Turner, 2014).

3. METODOLOGI PENELITIAN

Replika *middleware* dibuat sebagai lingkungan uji untuk menjalankan pengambilan data. Dalam Gambar 3 terlihat hasil dari perangkat replika *middleware*.



Gambar 3. Replika *middleware*

Implementasi mekanisme keamanan dilakukan dengan mengubah *source code* yang terdapat di dalam *middleware* dan *data center*.

3.1. Pengambilan Data

Pengambilan dilakukan menggunakan fitur

capture-packet pada program *tcpdump*. Hasil dari program *tcpdump* dibuka menggunakan aplikasi wireshark. Dengan wireshark, peneliti dapat mengetahui paket-paket yang dikirimkan dan diterima oleh *middleware*. Data-data yang didapat dari wireshark akan diolah untuk dianalisis.

Penggunaan CPU dan memory. Pengambilan data penggunaan cpu dan memory dilakukan dengan menggunakan program pada penelitian Fahrur (2017).

Parameter uji yang digunakan dalam penelitian ini adalah *packet loss*, *delay*, dan *jitter*. Nilai *packet loss* didapatkan dengan rumus berikut:

$$Packet\ loss = \frac{x-y}{x}\ x\ 100 \qquad (1)$$

Variabel x merupakan jumlah pesan yang seharusnya dikirimkan. Variabel y adalah jumlah pesan yang diterima. Nilai *delay* didapatkan dengan mengambil nilai *Delta time* pada *wireshark*. Nilai *jitter* didapatkan dengan rumus berikut:

$$jitter = \frac{total\ variasi\ delay}{total\ paket\ yang\ diterima-1} \qquad (2)$$

Skenario Pengiriman Data Tanpa Mekanisme Keamanan

Skenario ini adalah pengiriman data tanpa menggunakan mekanisme keamanan, kemudian selama sistem melakukan pengiriman, penulis menjalankan program *tcpdump* untuk meng-*capture* paket-paketnya.

Skenario Pengiriman Data Menggunakan *AES-256*

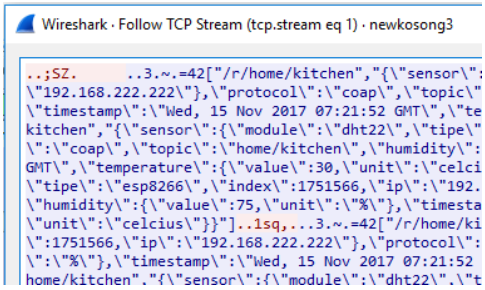
Skenario ini adalah pengiriman data menggunakan AES-256, kemudian selama sistem melakukan pengiriman, penulis menjalankan program *tcpdump* untuk meng-*capture* paket-paketnya.

Skenario Pengiriman Data Menggunakan *TLS/SSL*

Skenario ini adalah pengiriman data menggunakan TLS/SSL, kemudian selama sistem melakukan pengiriman, penulis menjalankan program *tcpdump* untuk meng-*capture* paket-paketnya.

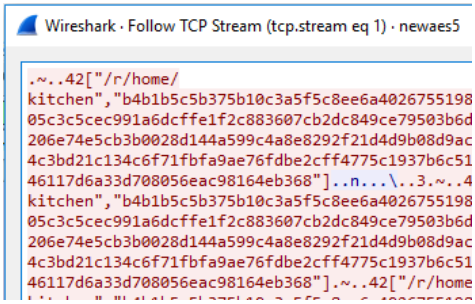
4. HASIL DAN PEMBAHASAN

Hasil berguna untuk menunjukkan hasil daripada penelitian yang dilakukan serta data yang didapatkan. Bab ini menunjukkan hasil pengambilan data dan pembahasan berdasarkan skenario-skenario yang sudah dirancang.



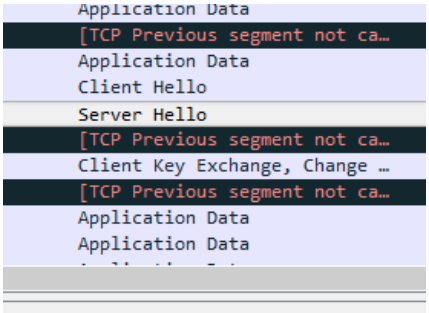
Gambar 4. Hasil *capture* data tanpa mekanisme keamanan

Dalam Gambar 4 terlihat bahwa data yang dikirimkan dari *middlware* ke *data center* tanpa menggunakan mekanisme keamanan mudah sekali dibaca. Hal ini menunjukkan bahwa proses pengiriman datanya memiliki lubang keamanan yang bisa dieksploitasi.



Gambar 5. Hasil *capture* data menggunakan *AES-256*

Dalam Gambar 5 terlihat bahwa data yang dikirimkan dari *middleware* ke *data center* menggunakan *AES-256* tidak bisa dibaca dengan mudah. Hal ini dikarenakan data yang dikirimkan telah dienkripsi menggunakan algoritma tersebut.



Gambar 6. Hasil *capture* proses *handshake TLS/SSL*

Dalam Gambar 6 terlihat bahwa terdapat proses pertukaran kunci atau *key exchange* yang terjadi pada proses pengiriman data

menggunakan *TLS/SSL*. Hal ini merupakan proses otentikasi yang digunakan oleh mekanisme tersebut, jika proses pertukaran kunci berhasil, maka koneksi akan dibuka dan pengiriman data dapat dilakukan.

Gambar 7. Paket data hasil *capture* data menggunakan *TLS/SSL*

Dalam Gambar 7 terlihat bahwa data yang dikirimkan dari *middleware* ke *data center* menggunakan TLS/*SSL* tidak bisa dibaca. Paket data terlihat lebih tidak jelas jika dibandingkan dengan *AES-256*, yang membuktikan bahwa TLS/*SSL* memiliki kekuatan keamanan melebihi *AES-256*.

4.1 Penggunaan CPU dan Memory

Pengambilan data dilakukan pada masing-masing mekanisme keamanan. Program mencatat penggunaan CPU dan memory dilakukan setiap 20 detik.

Tabel 3. Penggunaan CPU dan memory

Variabel	Tanpa mekanisme keamanan	AES-256	TLS/SSL
CPU Usage (%)	0.263330052	0.404019691	0.466400697
Memory usage (%)	2.849116228	4.428192053	4.482035297

4.2 Packet loss

Pengambilan data dilakukan sebanyak lima kali, kemudian dicari rata-rata dari nilai *packet loss*. Pengambilan data pada skenario 1 menghasilkan rata-rata *packet loss* sebesar 1.333333333%. Pengambilan data pada skenario 2 menghasilkan rata-rata sebesar 1%. Pengambilan data pada skenario 3 menghasilkan rata-rata sebesar 1.333333333%.

Tabel 2. Packet loss Skenario 1

Pengambilan data ke-	Jumlah paket dikirim	Jumlah paket diterima	Packet loss (%)
1	60	60	0
2	60	59	1.666666667
3	60	59	1.666666667
4	60	59	1.666666667
5	60	59	1.666666667
Rata-rata (%)			1.333333333

Tabel 3. Packet loss Skenario 2

Pengambilan data ke-	Jumlah paket	Jumlah paket retransmission	Packet loss (%)
1	60	60	0
2	60	59	1.666666667
3	60	59	1.666666667
4	60	60	0
5	60	59	1.666666667
Rata-rata (%)			1

Tabel 4. Packet loss Skenario 3

Pengambilan data ke-	Jumlah paket	Jumlah paket retransmission	Packet loss (%)
1	60	60	0
2	60	59	1.666666667
3	60	59	1.666666667
4	60	59	1.666666667
5	60	59	1.666666667
Rata-rata (%)			1.333333333

4.3 Delay

Pengambilan data dilakukan sebanyak lima kali, kemudian dicari rata-rata dari nilai *delay*. Pengambilan data pada skenario 1 menghasilkan rata-rata *delay* sebesar 0.156723573 detik. Pengambilan data pada skenario 2 menghasilkan rata-rata sebesar 0.168367841detik. Pengambilan data pada skenario 3 menghasilkan rata-rata sebesar 0.089008423 detik.

Tabel 5. Rata-rata Delay Skenario 1

Pengambilan data ke-	Rata -rata delay(detik)
1	0.153242513
2	0.136893707
3	0.148512657
4	0.164616042
5	0.180352948
Rata-rata akhir(detik)	0.156723573

Tabel 6. Rata-rata Delay Skenario 2

Pengambilan	Rata -rata delay(detik)

data ke-	
1	0.156576404
2	0.165701668
3	0.172463989
4	0.186801833
5	0.160295313
Rata-rata akhir(detik)	0.168367841

Tabel 7. Rata-rata Delay Skenario 3

Pengambilan data ke-	Rata -rata delay(detik)
1	0.118611342
2	0.080793532
3	0.071402141
4	0.09284886
5	0.081386241
Rata-rata akhir(detik)	0.089008423

4.4 Jitter

Pengambilan data dilakukan sebanyak lima kali, kemudian dicari rata-rata dari nilai *jitter*. Pengambilan data pada skenario 1 menghasilkan rata-rata *jitter* sebesar 0.006562287 detik. Pengambilan data pada skenario 2 menghasilkan rata-rata sebesar 0.008928956 detik. Pengambilan data pada skenario 3 menghasilkan rata-rata sebesar 0.003208877 detik.

Tabel 8. Rata-rata Jitter Skenario 1

Pengambilan data ke-	Jitter (detik)
1	0.001769156
2	0.007975226
3	0.002230408
4	0.011811921
5	0.009024724
Rata-rata (detik)	0.006562287

Tabel 9. Rata-rata Jitter Skenario 2

Pengambilan data ke-	Jitter (detik)
1	0.003256756
2	0.009690845
3	0.014197192
4	0.00209117
5	0.015408818
Rata-rata (detik)	0.008928956

Tabel 10. Rata-rata Jitter Skenario 3

Pengambilan data ke-	Jitter (detik)
1	0.001660939
2	0.004390695
3	0.006967346
4	0.001294216
5	0.001731188
Rata-rata (detik)	0.003208877

4.5 Perbandingan parameter uji antar mekanisme

Perbandingan dilakukan dengan mengambil rata-rata nilai setiap parameter uji pada setiap mekanisme. Perbandingan tersebut ditampilkan pada Tabel 10 di bawah ini.

Tabel 11. Perbandingan antar mekanisme

Mekanisme keamanan	Rata-rata Packet loss (%)	Rata -rata Delay (detik)	Rata-rata Jitter (detik)
Tanpa mekanisme keamanan	1.333333333	0.156723573	0.006562287
AES-256	1	0.168367841	0.008928956
TLS/SSL	1.333333333	0.089008423	0.003208877

5. KESIMPULAN

Berdasarkan penelitian yang telah dilakukan, dapat disimpulkan bahwa mekanisme keamanan AES-256 mampu mengamankan paket data menggunakan enkripsi. Pengaruh dari penggunaan AES-256 pada kinerja *middleware* adalah penurunan jumlah *packet loss* dari 1.333333333% menjadi 1%, peningkatan rata-rata *delay* dari 0.156723573 detik menjadi 0.168367841 detik, dan peningkatan rata-rata *jitter* dari 0.006562287 detik menjadi 0.008928956 detik.

Jika dibandingkan dengan TLS/SSL, mekanisme keamanan AES-256 lebih lemah dalam hal keamanan, karena pada TLS/SSL terdapat proses *key exchange* sebagai proses *authentication*. Sedangkan pengaruh TLS/SSL terhadap kinerja *middleware* adalah tidak ada peningkatan jumlah *packet loss* yaitu 1.333333333% dan penurunan rata-rata *jitter* menjadi 0.003208877 detik, lebih kecil dibandingkan dengan AES-256. Pada parameter *delay*, peningkatan rata-rata *delay*-nya lebih sedikit dibandingkan AES-256, yaitu 0.089008423 detik.

Kedua mekanisme keamanan dapat mengamankan paket yang dikirim menggunakan metode enkripsi. Keduanya juga mempengaruhi penggunaan CPU dan memory, tetapi perbedaan antara kedua mekanisme keamanan tidak terlalu besar. TLS/SSL memberikan mekanisme

keamanan yang lebih kuat dengan proses *key exchange* sebagai metode *authentication*. Walaupun TLS/SSL memberi penurunan kinerja yang lebih besar pada *packet loss* dan *jitter*, penurunan kinerja tersebut tidak berdampak besar pada *middleware*. Jadi, TLS/SSL lebih direkomendasikan untuk digunakan sebagai mekanisme keamanan pada komunikasi *end-to-end IoT middleware* dengan subscriber berbasis protokol HTTP, dibandingkan dengan AES-256.

Untuk penelitian selanjutnya dapat dilakukan pada lingkungan uji yang berbeda, jumlah komunikasi dengan *middleware*, parameter uji, serta mekanisme keamanan yang berbeda.

7. DAFTAR PUSTAKA

Ahmad M, Taj S, Mustafa T, Asri Md., 2012. Performance Analysis of Wireless Network with the impact of Security Mechanisms. Pakistan.

Anwari H., 2017. Pengembangan IoT Middleware Berbasis Eventbase Dengan Potokol Komunikasi CoAP, MQTT dan Websocket. Malang. Universitas Brawijaya.

Bhajaj R.D, Gokhale Dr U.M., 2016. AES Algorithm for Encryption. Nagpur. G.H. Raisoni Institute of Engineering and Technology for Women.

Fahrur R.M., 2017. *Analisis Performansi dan Skalabilitas pada Event-Based IoT Middleware*. Malang. Universiter Brawijaya.

Islam N, Chandra B.C, Hasan J, Islam S.A., 2016. Quality of Service Analysis of Ethernet Network Based on Packet Size. Bangladesh.

Maqsood, Faiqa & Ahmed, Muhammad & Mumtaz, Muhammad & Shah, Munam. (2017). Cryptography: A Comparative Analysis for Modern Techniques. International Journal of Advanced Computer Science and Applications. 8. . 10.14569/IJACSA.2017.080659.

Pramukantoro E.S, W. Yahya, G. Arganata, A. Bhawiyuga and A. Basuki, "Topic based IoT data storage framework for heterogeneous sensor data" 2017 11th International Conference on Telecommunication Systems Services and Applications (TSSA), Lombok, Indonesia, 2017, pp. 1-4. doi: 10.1109/TSSA.2017.8272895

Rajra M.B.B, J Deepa ME., 2015. A Survey on Network Security Attacks and Prevention Mechanism. Nagercoil. Department of Computer Science Ponjesly College of Engineering Nagercoil.

Ramdhansya A.F, Ariyanto E, Nuha H.H., 2014. Implementasi Advanced Encryption (AES) Pada Sistem Kunci Elektronik Kendaraan Berbasis Sistem Operasi Android dan Mikrokontroler Arduino. Bandung. Universitas Telkom.

Tehupeiory N, Chandra D.W., 2016. Analisis Perbandingan Mekanisme Secure Socket Layer (SSL) dan Transfer Layer Security (TLS) Pada Koneksi File Transfer Protocol (FTP) Server Ubuntu. Salatiga. Universitas Kristen Satya Wacana.

S. Turner, "Transport Layer Security," in *IEEE Internet Computing*, vol. 18, no. 6, pp. 60-63, Nov.-Dec. 2014. doi: 10.1109/MIC.2014.126