Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer e-ISSN: 2548-964X
Vol. 3, No. 4, April 2019, hlm. 4006-4013 http://j-ptiik.ub.ac.id

Implementasi Algoritme Poly1305-AES pada Protokol MQTT

Tista Pamungkas Ragil Alit[1], Ari Kusyanti [2], Widhi Yahya[3]

Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya
Email: [1]tista.pam@gmail.com, [2]ari.kusyanti@ub.ac.id, [3]widhi.yahya@ub.ac.id

Abstrak

Message Queue Telemetry Transport (MQTT) merupakan salah satu protokol *Internet of Things* (IoT) yang dirancang khusus untuk komunikasi antar mesin yang memiliki karakteristik dapat bekerja pada *low power*, menggunakan *bandwidth* yang kecil, keandalan dalam pengiriman paket dan protokol ini menggunakan arsitektur *publish-subscribe*. Protokol MQTT hanya menyediakan mekanisme autentikasi untuk keamanannya yang secara *default* tidak menjamin keamanan data dalam transmisinya sehingga privasi data dan integritas data menjadi masalah dalam implementasi protokol. Oleh sebab itu, dilakukan penerapan metode *Message Authentication Code* (MAC) menggunakan algoritme Poly1305-AES yang berbasis *block cipher*. Berdasarkan pengujian, algoritme Poly1305-AES memiliki peningkatan penggunaan *memory* 0,013 MB pada *publisher* dan 0,028 MB pada *subscriber* dan algoritme Poly1305-AES mampu menangani serangan perubahan, penyisipan dan penyubtitusian data. Penelitian ini memberikan hasil bahwa algoritme Poly1305-AES memiliki performa yang cukup bagus berdasarkan nilai peningkatan penggunaan *memory* dan ketahanan terhadap serangan.

Kata kunci: *IoT, MQTT, integritas, Message Authentication Code, Poly1305-AES*

Abstract

Message Queuing Transport Telemetry (MQTT) is one of the Internet of Things (IoT) protocols specifically designed for communication between machines that have characteristics that can be used at low power, uses small bandwidth, connectivity in packet delivery and protocols using this technology to publish- subscribe. The MQTT protocol only provides authentication security for security which defaults do not guarantee the security of data in transmission so that data privacy and data integrity are problems in the implementation of the protocol. Therefore, the application of the Message Authentication Code (MAC) method uses the Poly1305-AES algorithm based on block ciphers. Based on testing, the Poly1305-AES algorithm has an increase in memory usage of 0.013MB to publishers and 0.028MB to customers and the Poly1305-AES algorithm can support changes, insertions and substitution data. This study presents the results of the Poly1305-AES algorithm which has a pretty good performance based on the value of increasing memory usage and resistance to attacks.

Keywords: *IoT, MQTT, integrity, Message Authentication Code, Poly1305-AES*

1. PENDAHULUAN

Pada tahun 1999 *Internet of Things* (IoT) dikenalkan oleh Kevin Ashton. Sejak 20 tahun stelah dikenalkan, hingga saat ini belum mempunyai sebuah kesepakatan mengenai definisi *Internet of Things* (IoT). *Internet of Things* (IoT) adalah sistem informasi global yang sangat besar terdiri dari banyak objek yang dapat diidentifikasi, dirasakan dan diproses berdasarkan standar dan interoperabilitas protokol komunikasi (Wangbong, et al., 2016). Ada dua aristektur yang dapat diterapkan pada jaringan IoT, yaitu *client-server* dan *publish-subscribe*.

Protokol *Message Queue Telemetry Transport* (MQTT) adalah protokol yang dirancang khusus untuk komunikasi antar mesin yang memiliki karakteristik dapat bekerja pada kondisi *low power*, kondisi *bandwidth* yang kecil, keandalan dalam pengiriman paket dan protokol ini berarsitektur *publish-subscribe* (Atmoko, et al., 2017). Proses pengumpulan data pada arsitektur *publish-subscribe* berdasarkan *topic* yang didefinisikan oleh *publisher* dan *subscriber* akan menyatakan ketertarikan atas *topic*. Sehingga *topic* pada protokol MQTT bersifat penting karena pada *publish-subscribe* pengumpulan data dikelompokkan berdasarkan *topic*.

Protokol *Message Queue Telemetry Transport* (MQTT) hanya menyediakan mekanisme autentikasi untuk keamanannya yang secara *default* tidak mengenkripsi data dalam transfer datanya sehingga privasi data dan integritas data menjadi masalah dalam implementasi protokol MQTT (Andy, et al., 2017).

Berdasarkan permasalahan umum implementasi jaringan IoT dan penggunaan protokol MQTT, penelitian ini hanya berfokus di permasalahan keamanan di jaringan IoT pada sisi integritas data. Integritas data berhubungan dengan penjagaan perubahan data atau manipulasi data oleh pihak-pihak yang tidak berhak seperti perubahan, penyubtitusian, penghapusan, dan penyisipan data yang berbeda ke dalam data yang asli. Maka metode pengecekan integritas data sangat dibutuhkan untuk implementasi IoT.

Permasalahan keamanan pada sisi integritas data sangatlah penting diperhatikan. Karena data yang diterima memengaruhi hasil pemrosesan lanjutan seperti pengambilan keputusan oleh unit kontrol dan penyimpanan pada basis data. Menurut penelitian yang dilakukan oleh Yindong, Liping, & Ziran yang berjudul *An Approach to Verifying Data Integrity for Cloud Storage*, membahas pendekatan verifikasi integritas data memberikan usulan menggunakan metode kriptografi *Message Authentication Code* (MAC) untuk melakukan pengecekan integritas data dan autentikasi. Metode MAC dapat dirancang dengan dua pendekatan. Pendekatan pertama fungsi *hash* satu arah, sedangkan pendekatan kedua menggunakan *symmetric cipher* berbasis *block cipher* (Munir R., 2006).

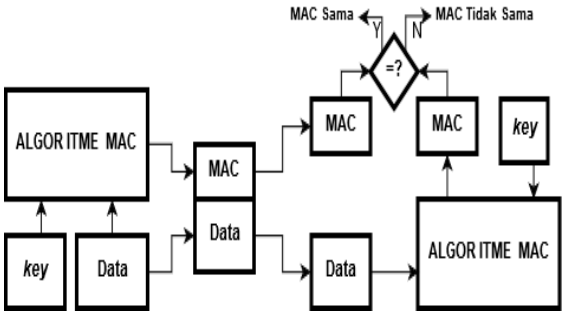
MAC yang dibuat berdasarkan pendekatan fungsi *hash* seperti HMAC-MD5 atau HMAC-SHA1 dianggap rentan terhadap serangan *collision* (Aumasson, et al., 2013). Oleh karena itu, penelitian ini mengimplementasikan MAC menggunakan *symmetric cipher* berbasis *block cipher* dengan algoritme Poly1305-AES. Algortime Poly1305-AES memiliki proses komputasi yang cepat, keamanan yang dapat dijamin, serta kecepatan komputasinya tidak hanya terbatas terhadap pesan yang berukuran pendek (Bernstein J. D., 2005).

Untuk implementasi integritas data pada protokol MQTT, terdapat tiga aktor, yaitu *broker*, *publisher* dan *subscriber*. Pada bagian *publisher* menggunakan perangkat keras Raspberry Pi yang akan mengirimkan data sensor atau mem-*publish* data sensor. Pada bagian *broker* bertugas untuk menjadi jembatan antara *publisher* dan *subscriber* dalam segala pengiriman alur data yang terjadi. Pada bagian *subscriber* bertugas menerima data sensor dari *publisher* atau *subscribe topic* yang disediakan oleh *publisher* dan memasukkan ke dalam basis data. Sedangkan implementasi fitur keamanan integritas data pada sistem ini menggunakan algoritme Poly1305-AES yang akan diterapkan pada *publisher* dan *subscriber*. Dengan dibuatnya sistem pengecekan integritas data sensor pada protokol MQTT menggunakan algoritme Poly1305-AES, diharapkan dapat memberi solusi keamanan data, khususnya pada sisi integritas data.

2. DASAR TEORI

2.1 MAC

Message Authentication Code (MAC) merupakan tanda pengenal sebagai pembuktian integritas suatu dokumen atau pesan yang didapatkan dengan membuat pesan tak bermakna yang dihasilkan dari pemrosesan isi dokumen dengan menggunakan sebuah *private key*. Secara teknis, dokumen diproses dengan *private key* sehingga dapat menghasilkan kode MAC. Kode MAC kemudian digabung dengan dokumen dan setalah itu dikirimkan ke penerima. Kemudian penerima juga menggunakan *private key* yang sama untuk membuat kode MAC dari dokumen yang diterima dan setelah itu, membandingkannya dengan kode MAC yang diterima dari pengirim. Cara kerja dari MAC akan ditunjukan pada Gambar 1.



Gambar 1. Cara kerja MAC

2.2 *Advanced Encryption Standard*

Advanced Encryption Standard (AES) dibuat oleh Dr. Joan Daemen dan Dr. Vincent Rijment, dipublikasikan oleh *National Institute of Standard and Technology* (NIST) tahun 2001

untuk digunakan menggantikan algoritme *Data Encryption Standard* (DES) yang semakin mudah untuk dibobol kuncinya. Algoritme ini diperoleh melalui kompetisi yang dilakukan pada tahun 1997.

Algoritme AES mempunyai *block* data berukuran 128, 168, 192, 224 dan 256 bit dan *key* 128, 192 dan 256 bit. Standar *block* data dan *key* yang digunakan pada algoritme AES adalah128 bit. Panjang *key* memengaruhi jumlah *round* pada algoritme AES. Operasi algoritme AES dilakukan pada *array* dua dimensi yang biasa disebut *state* dan *state* yang akan memproses *encryption* dan *decryption*. Proses *encrypt* pada algoritme AES terdiri dari empat jenis transformasi *bytes* yaitu *AddRoundKey*, *SubBytes*, *ShiftRows* dan *MixColumns*. Penjelasan proses *encryption* dan *decryption* dapat dilihat pada Tabel 1.

Tabel 1. *Pseudocode* Algoritme AES 128

```
Pseudocode Algoritme AES 128
    Initialization:
    Nk = 4, Nb = 4, Nr = 10
    Encryption:
      Input: PlainText, Key
      KeyExpansion((Key) (Nr, Nb))
      AddRoundKey (0, Nb)
      for i=1 in Nr
         SubBytes(Nb)
         ShiftRows(Nb)
         MixColumns(Nb)
         AddRoundKey(Nb, KeyExpansion, i)
      SubBytes(Nb)
      ShiftRows(Nb)
      AddRoundKey(Nb, KeyExpansion, i+1)
      Output: ChipherText
    Decryption:
      Input: ChipherText, Key
      KeyExpansion((Key) (Nr, Nb))
      AddRoundKey(0, Nb)
      For i=1 in Nr
         InvShiftRows(Nb)
         InvSubBytes(Nb)
         InvMixColumns(Nb)
         AddRoundKey(Nb, KeyExpansion, i)
      InvShiftRows(Nb)
      InvSubBytes(Nb)
      AddRoundKey(Nb, KeyExpansion, i+1)
      Output: PlainText
```

Proses pertama *encryption* adalah *input* yang disalin ke *state* akan mengalami transformasi *bytes* berupa *AddRoundKey*. Setelah itu, *state* akan ditransformasi *bytes* dengan *SubBytes*, *ShiftRows*, *MixColumns* dan *AddRoundKey* secara berulang sebanyak jumlah *round*. Perlu diketahui pada perulangan terakhir proses *encryption* tidak terjadi transformasi *MixColumns*. Sedangkan proses *decryption* AES merupakan proses *invers* dari proses *encryption* AES.

2.3 Poly1305-AES

Poly1305-AES adalah jenis algoritme MAC yang digunakan untuk autentikasi integritas pesan yang memakai 32 *bytes secret key* yang dibagi antara pengirim dan penerima agar penerima bisa membandingkan nilai autentikasi *tag* yang dihitung oleh pengirim dengan autentikasi *tag* yang dihitung sendiri oleh penerima. Algoritme Poly1305-AES memakai algoritme kriptografi *cipher block* AES untuk memproses enkripsi *key*. Tingkat keamanan algoritme Poly1305-AES setara dengan tingkat keamanan AES.

Algoritme Poly1305-AES mengkomputasi 16 *bytes* pesan yang merupakan urutan *byte-byte.* setiap *byte* adalah bentuk bilangan *integer* positif 0 sampai 255, *key* yang dipakai adalah 32 *bytes secret key* yang bagi antara pengirim dan penerima, dan dibagi lagi menjadi dua bagian. 16 *bytes* pertama *key* AES k, dan yang bagian kedua adalah *string* 16 *bytes*. Pada bagian kedua merepresentasikan 128 bit *integer* (r) dalam format *little endian* dan *nonces* (n). Algoritme Poly1305-AES menambahkan *nonce* denagan AESk untuk membangkitkan 16 *bytes string* AESk (n). *Pseudocode* algoritme Poly1305-AES akan dijelaskan pada Tabel 2.

Tabel 2. Pseudocode Algoritme Poly1305-AES

```
Pseudocode Algoritme Poly1305-AES
Input: k (key), r (little endian),
n(nonce), msg(pesan)
mod1305 = mod2^130 - 5
endianR = little_endian(r)
Panjang pesan = (len(msg) + 15) / 16
Padding = 0
Perulangan i dengan Panjang pesan
      Partisi pesan msg[i*16 :
i*16+16] + b"\x01"
      Partisi pesan += (17 -
len(Konversi)) * b"\x00"
      num = endianR(Partisi pesan)
      Padding (Padding + num) * endianR
Padding pesan % mod1305
Input AES(k, n)
Output: hasil (Padding pesan % mod1305
+ Input AES(k, n)) % (1 << 128)
```

2.3 MQTT

Protokol *Message Queue Telemetry Transport* (MQTT) merupakan salah satu protokol yang dirancang khusus untuk komunikasi antar mesin yang memiliki karakteristik mampu bekerja pada keadaan *low power*, konsumsi *bandwidth* yang sedikit, keandalan dalam pengiriman paket dan protokol ini menggunakan arsitektur *publish-subscribe* (Atmoko, et al., 2017). MQTT adalah protokol

komunikasi berarsitektur *publish-subscribe* yang dikelompokkan berdasarkan *topic* dan protokol yang sangat ringan dan sederhana. Protokol ini didesain untuk digunakan pada alat yang memiliki kemampuan terbatas dan jaringan yang kurang dapat diandalkan.

3. METODOLOGI

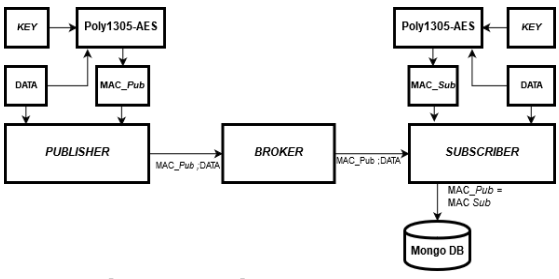
Metodologi menjelaskan tahapan-tahapan penelitian. Dimulai dari tahap studi literatur, tahap analisis kebutuhan, tahap perancangan, tahap implementasi, tahap pengujian dan analisis sistem serta tahap kesimpulan dan saran. Diagram alir metodologi penelitian ditunjukkan pada Gambar 2.

Gambar 2. Diagram alir metodologi penelitian

4. PERANCANGAN DAN IMPLEMENTASI

4.1 Gambaran Umum Sistem

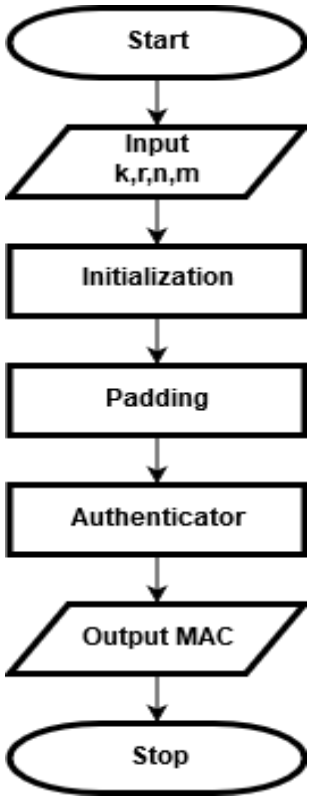
Pada penelitian ini, protokol MQTT memiliki beberapa komponen yaitu, *publisher*, *broker* dan *subscriber*. *Publisher* akan mengirimkan data sensor beserta MAC menuju *broker*. *Broker* akan menerima data dari *Publisher*. Sedangkan *subscriber* bertugas untuk mengambil data dari *broker* dan mencocokan MAC berdasarkan data yang diterima dari *broker* dan menyimpan data sensor ke dalam basis data. Gambar 3 menjelaskan gambaran umum dari implementasi protokol MQTT.



Gambar 3. Gambaran umum sistem

4.2 Perancangan Pengimplementasian Algoritme Poly1305-AES

Perancangan komponen algoritme bertujuan untuk menjelaskan alur dan cara kerja algoritme Poly1305-AES. Perancangan komponen algoritme Poly1305-AES diterapkan pada *publisher* dan *subscriber* akan dijelaskan dengan diagram alir. Algoritme Poly1305-AES diperlukan untuk pengecekan integritas data pada *publisher* dan *subscriber* dengan menghasilkan kode MAC. Secara garis besar akan ditunjukkan melalui Gambar 4.

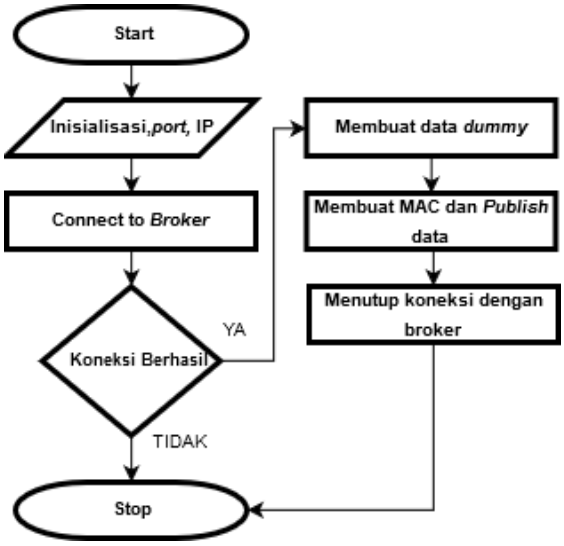


Gambar 4. Diagram alir Poly1305-AES

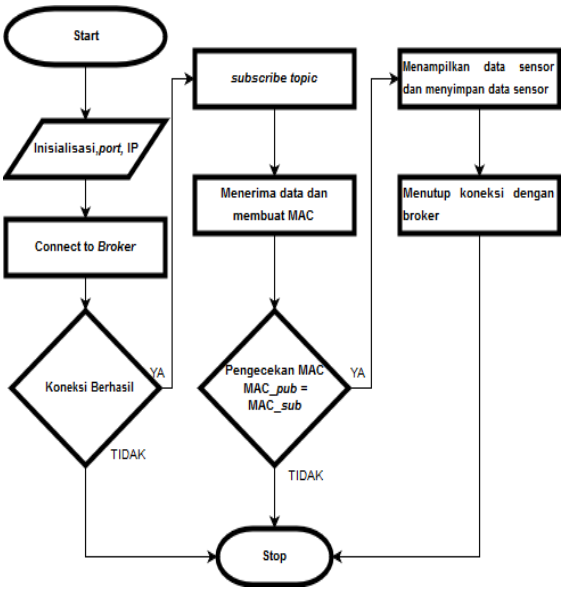
Berdasarkan Gambar 5, fungsi Poly1305-AES membutuhkan *input* berupa *key* (k), bilangan *little endian* (r), *nonce* (n) dan pesan (msg). Ada tiga bagian proses yang dijalankan sebelum membuat MAC Poly1305-AES yaitu *Initialization*, *Padding* dan *Autenticator*. Pada proses *Initialization* akan menginisiasi variabel parameter masukan. Setelah proses *Initialization* terdapat proses *Padding* untuk membuat pesan terbagi berdasarkan panjang kelipatan pesan dan menambahan bit-bit *dummies* sehingga pesan dapat menggenapi panjang blok yang ditentukan. Proses terakhir yaitu *Autenticator* yang berfungsi untuk mengitung hasil *padding* pesan ditambah dengan *key* AES dan *nonce* yang akan dijadikan sebagai hasil *output* MAC.

4.3 Perancangan Sistem

Perancangan sistem menggambarkan alur kerja sistem yang dibuat pada penelitian. Penelitian ini membuat sitem jaringan IoT pada protokol MQTT yang menyediakan fitur untuk mengirim data sensor, menerima data sensor dan pengecekan integritas data sensor menggunakan algoritme Poly1305-AES. Protokol MQTT merupakan protokol yang berbasis arsitektur komunikasi *publish-subscribe* yang terdiri dari tiga komponen yaitu *publisher*, *broker*, dan *subscriber*. Perancangan sistem dibagi berdasarkan komponen yang saling berinteraksi dengan komponen *broker* yaitu komponen *publisher* dan komponen *subscriber*. Perancangan sistem secara garis besar akan ditunjukkan melalui Gambar 5 dan Gambar 6.



Gambar 5. Perancangan komponen *publisher*



Gambar 6. Perancangan komponen *subscriber*

5. HASIL PENGUJIAN DAN PEMBAHASAN

5.1 Pengujian *Test Vector*

Pengujian *Test Vector* dilakukan untuk memastikan algoritme Poly1305-AES yang diterapkan pada penelitian ini sudah benar dan sesuai dengan ketentuan. *Message Authentication Code* yang dihasilkan algoritme Poly1305-AES akan dibandingkan dengan *test vector* yang merujuk pada parameter *test vector* yang diperoleh dari pembuat algoritme Poly1305-AES. Berikut ini hasil *test vector* algoritme Poly1305-AES akan ditunjukan pada Tabel 3.

Tabel 3. *Test Vector*

Test Vector	**Hasil**
0xf4c633c3044fc145f84f335cb81953de	Valid
0xdd3fab2251f11ac759f0887129cc2ee7	Valid
0x0ee1c16bb73f 0f4fd19881753c01cdbe	Valid
0x5154ad0d2cb26e01274fc51148491f1b	Valid

Pada Tabel 3 menunjukkan hasil pengujian yang dilakukan berdasarkan skenario pengujian yang dijelaskan pada perancangan pengujian *test vector*. Karena semua skenario yang telah dilakukan menunjukkan hasil yang sama antara perbandingan *input* parameter uji dengan *output vector*, maka dapat ditarik kesimpulan bahwa algoritme Poly1305-AES yang dibuat telah benar dan sesuai.

5.2 Pengujian Waktu Eksekusi Algoritme

Pengujian waktu eksekusi algoritme dilaksanakan untuk mengetahui berapa lama waktu algoritme Poly1305-AES untuk menghasilkan kode MAC. Sekenario pengujian eksekusi algoritme dengan menjalankan kode program Poly1305-AES yang diimplementasikan pada bagian *publisher* dan bagian *subscriber* sebanyak 30 kali dan 10 kali iterasi. Penghitungan waktu eksekusi akan dimulai saat *key* dan data diterima sebagai masukan algoritme sampai pemrosesan keluaran berupa MAC. Hasil pengujian waktu eksekusi algoritme pada *publisher* didapat rata-rata waktu 0,00123 ms, waktu maksimum 0,00150 ms dan waktu minimum 0,00092 ms. Sedangkan pada

subscriber memiliki rata-rata waktu eksekusi 0,00128 ms, waktu maksimum 0,00176 ms dan waktu minimum 0,00090 ms. Maka dapat disimpulkan hasil keseluruhan pengujian waktu eksekusi algoritme terdapat perbedaan yang tidak signifikan antara waktu eksekusi algoritme Poly1305-AES pada *publisher* dan *subscriber*.

5.3 Pengujian Peforma Sistem

Pengujian performa sistem dilakukan untuk mengetahui perbandingan kinerja sistem dan kinerja jaringan dalam pemrosesan algoritme berdasarkan penggunaan *memory* sistem, waktu dan *memory* pengecekan integritas data. Hasil pengujian penggunaan *memory* diperoleh dari 30 kali *publish-subscribe* dengan 10 kali iterasi saat sistem menggunakan algoritme dan tidak menggunakan algoritme. Berikut ini hasil pengujian penggunaan *memory*.

Tabel 4. Penggunaan *Memory*

Memory	***Publisher***		***Subscriber***	
	Dengan MAC	**Tanpa MAC**	**Dengan MAC**	**Tanpa MAC**
Rata-rata (MB)	0,350	0,363	0,771	0,799

Dari Tabel 4 maka dapat disimpulkan dari hasi pengujian penggunaan *memory* algoritme Poly1305-AES menggunakan ruang *memory* sebesar 0,0134 MB pada bagian *publisher* dan *memory* sebesar 0,0284 MB pada bagian *subscriber*. Pengujian waktu dan *memory* pengecekan integritas data diperoleh dari skenario *publish* data dari penggunaan *publisher* sebanyak 30, 60, 90, 120, dan 150 dalam waktu yang sama. Hasil pengujian waktu dan *memory* pengecekan integritas data ditunjukkan pada Tabel 5.

Tabel 5. Waktu dan *Memory* Pengecekan Integritas

Jumlah *Publisher*	**30**	**60**	**90**	**120**	**150**
Waktu (s)	13,7	20,3	28	36,8	44,8
***Memory* (MB)**	0,72	0,73	0,73	0,749	0,75

Dapat dilihat dari nilai rata-rata Tabel 4 bisa diambil kesimpulan bahwa peningkatan waktu pada saat mealkukan pengecekan integritas data adalah 7,764 detik pada setiap kelipatan penggunaan 30 *publisher* dan penggunaan *memory* saat melakukan pengecekan integritas data adalah 0,0067 MB setiap kelipatan penggunaan 30 *publisher*.

5.4 Pengujian Keamanan

Pengujian keamanan dilakukan untuk memastikan keamanan sistem yang dibuat. Pada pengujian ini peneliti menggunakan serangan *Man in The Middle Attack* (MITM) ke *broker* dengan metode *arp spoofing* yang memungkinkan untuk melakukan perubahan, penyisipan dan penyubtitusian data. Untuk melakukan serangan *arp spoofing* peneliti memakai aplikasi Ettercap. Serangan pertama yaitu perubahan data sensor ditunjukkan pada Gambar 7.

```
broker@subscriber:~$ python subscriber.py
Connected to Broker
Genarate MAC Poly1305-aes Subscriber  :  4829741ef83e30501b6ad1d32e
37276d
=====================================
Generate MAC Poly1d305-aes Publisher  :  4829741ef83e30501b6ad1d32e
37276d
Topik /sensor Payload  : 70  QoS 1
=====================================
-------------------------------------
Genarate MAC Poly1305-aes Subscriber  :  280954fed71e1030fb49b1b30e
17074d
=====================================
Generate MAC Poly1d305-aes Publisher  :  280954fed71e1030fb49b1b30e
17074d
Topik /sensor Payload  : 60  QoS 1
=====================================
-------------------------------------
Genarate MAC Poly1305-aes Subscriber  :  f3c306ba93dacbebb6056d6fca
d2c208
Generate MAC Poly1305-aes Publisher   :  8869b45e387f70905baa11146f
7767ad
Topik /sensor Payload  : 22 QoS 1
=====================================
MAC PUBLISHER DAN SUBSCRIBER TIDAK SESUAI
-------------------------------------
Genarate MAC Poly1305-aes Subscriber  :  f3c306ba93dacbebb6056d6fca
d2c208
Generate MAC Poly1305-aes Publisher   :  6849943e185f50703b8af1f34e
57478d
Topik /sensor Payload  : 22 QoS 1
=====================================
MAC PUBLISHER DAN SUBSCRIBER TIDAK SESUAI
```

Gambar 7. Penyerangan perubahan data

Pada Gambar 7 dapat dilihat bahwa *subscriber* telah menerima data dari *broker dan subscriber* menampilkan pesan MAC *publisher* dan *subscriber* tidak sesuai karena *subscriber* memiliki mekanisme pengecekan integritas data dengan membadingkan nilai MAC dari *publisher* dan *subscriber*. Serangan penyisipan ditunjukkan pada Gambar 8.

```
broker@subscriber:~$ python subscriber.py
Connected to Broker
Genarate MAC Poly1305-aes Subscriber  :  10ca14bf98dfd0f0bb0a7274cf
d7c70d
=====================================
Generate MAC Poly1d305-aes Publisher  :  10ca14bf98dfd0f0bb0a7274cf
d7c70d
Topik /sensor Payload  : 75  QoS 1
=====================================
-------------------------------------
Genarate MAC Poly1305-aes Subscriber  :  4829741ef83e30501b6ad1d32e
37276d
=====================================
Generate MAC Poly1d305-aes Publisher  :  4829741ef83e30501b6ad1d32e
37276d
Topik /sensor Payload  : 70  QoS 1
=====================================
-------------------------------------
Genarate MAC Poly1305-aes Subscriber  :  702cf7a07ac1b2d29dec5356b1
b9a9ef
Generate MAC Poly1305-aes Publisher   :  f0a9f49e78bfb0d09bea5154af
b7a7ed
Topik /sensor Payload  : 650 QoS 1
=====================================
MAC PUBLISHER DAN SUBSCRIBER TIDAK SESUAI
-------------------------------------
Genarate MAC Poly1305-aes Subscriber  :  d08c5701db211333fe4cb4b611
1a0a50
Generate MAC Poly1305-aes Publisher   :  500a55ffd81f1131fc4ab2b40f
18084e
Topik /sensor Payload  : 950 QoS 1
=====================================
MAC PUBLISHER DAN SUBSCRIBER TIDAK SESUAI
```

Gambar 8. Penyerangan penyisipan data

Pada Gambar 8 dapat dilihat bahwa *subscriber* telah menerima empat data dari *broker* dimana dua diantaranya telah diserang dengan menyisipkan angka 0 pada bagian akhir data sensor *dan subscriber* menampilkan pesan MAC *publisher* dan *subscriber* tidak sesuai karena *subscriber* memiliki mekanisme pengecekan integritas data dengan membadingkan nilai MAC dari *publisher* dan *subscriber.* Serangan penyubtitusian ditunjukkan pada Gambar 9.

```
broker@subscriber:~$ python subscriber.py
Connected to Broker
Genarate MAC Poly1305-aes Subscriber  :  d089d47e589f90b07bca31348f
9787cd
=====================================
Generate MAC Poly1d305-aes Publisher  :  d089d47e589f90b07bca31348f
9787cd
Topik /sensor Payload  : 55  QoS 1
=====================================
-------------------------------------
Genarate MAC Poly1305-aes Subscriber  :  500a55ffd81f1131fc4ab2b40f
18084e
=====================================
Generate MAC Poly1d305-aes Publisher  :  500a55ffd81f1131fc4ab2b40f
18084e
Topik /sensor Payload  : 95  QoS 1
=====================================
-------------------------------------
Genarate MAC Poly1305-aes Subscriber  :  7849943e185f50703b8af1f34e
57478d
Generate MAC Poly1305-aes Publisher   :  f0a9f49e78bfb0d09bea5154af
b7a7ed
Topik /sensor Payload  : 62  QoS 1
=====================================
MAC PUBLISHER DAN SUBSCRIBER TIDAK SESUAI
-------------------------------------
Genarate MAC Poly1305-aes Subscriber  :  5829741ef83e30501b6ad1d32e
37276d
Generate MAC Poly1305-aes Publisher   :  d089d47e589f90b07bca31348f
9787cd
Topik /sensor Payload  : 52  QoS 1
=========================================
MAC PUBLISHER DAN SUBSCRIBER TIDAK SESUAI
```

Gambar 9. Penyerangan penyubtitusian data

Pada Gambar 9 dapat dilihat bahwa *subscriber* telah menerima empat data dari *broker* dimana dua diantaranya telah diserang dengan subtitusi angka terakhir data sensor dengan angka pada bagian akhir data sensor dan *subscriber* menampilkan pesan MAC *publisher* dan *subscriber* tidak sesuai karena *subscriber* memiliki mekanisme pengecekan integritas data dengan membadingkan nilai MAC dari *publisher* dan *subscriber.* Oleh karena itu, dapat diambil kesimpulan bahwa *subscriber* mampu mengatasi serangan perubahan, penyisipan dan penyubtitusian data dan *subscriber* dapat menjamin integritas data yang ditransmisi.

6. KESIMPULAN DAN SARAN

Subscriber mampu mengetahui integritas data sensor dengan menghitung nilai MAC berdasarkan nilai data sensor yang diterima. MAC dapat dihitung dengan inputan *key, integer, nonce* dan data sensor yang diterima. Jika *output* MAC yang dibuat oleh *subscriber* sama dengan MAC yang dibuat oleh *publisher*, maka data sensor yang diterima dari *broker* bisa dijamin integritasnya dan tidak mengalami perubahan, penyisipan dan penyubtitusian saat transmisi data oleh *broker*.

Algoritme Poly1305-AES saat digunakan untuk pengecekan integritas data pada protokol MQTT memiliki rata-rata waktu 0,00123 ms untuk menghasilkan MAC pada *publisher* dan pada *subscriber* sebesar 0,00128 ms. Sedangkan peningkatan penggunaan *memory* ketika algoritme Poly1305-AES diterapkan pada sistem sebesar 0,013 MB pada *publisher* dan 0,028 MB pada *subscriber*. Pengecekan integritas data yang dilakukan *subscriber* meningkat 7,764 detik dan penigkatan penggunaan *memory* 0,0067 MB setiap kelipatan 30 *publisher*.

Dalam penelitian ini tidak melakukan penambahan aspek kerahasiaan data menggunakan algoritme enkripsi, tidak melakukan mekanisme *key management* dan tidak menggunakan perangkat keras sensor. Untuk itu, diharapkan adanya penambahan aspek kerahasian data, *key management* dan perangkat sensor agar data yang diperoleh lebih nyata dan akurat dan mendapatkan hasil yang lebih baik pada penelitian selanjutnya.

7. DAFTAR PUSTAKA

Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., & Ayyash, M. 2015. Internet of Things: A Survey on Enabling Technologies, Protocols and Applications. IEEE Communication Surveys & Tutorial, 17(4), 2347-2376.

Andy, S., Rahardjo, B., & Hanindhito, B. 2017. Attack Scenarios and Security Analysis of MQTT Communication Protocol in IoT System. International Conference on Electrical Engineering, Computer Science and Informatics (EECSI).

Atmoko, R. A., Rianti, R., & Hasin, M. K. 2017. IoT real time dataiiacquisition using MQTTiprotocol.

Bernstein, J., Daniel. (2005). The Poly1305-AES message-authentication code. Chicago: The University of Illinois at Chicago

Guoqiang, S., Yanming, C., Zuo, C., & Zhu, Y. 2013. Design and Implementation of a Smart IoT Gateway. 2013 IEEE International Confrence on Green Computing and Communications and IEEE Cyber, Physical and Social Computing, 720-723.

Hunkeler,0U., Truong, H. L., &0Clark, A. S. 2015. A Publish/Subscribe Protocol for Wireless Sensor Network. IEEE

Munir, R. 2005. Kriptografi.

MQTT. (2016). Frequently Asked Question. Retrieved Februari 14, 2018, from http://www.mqtt.org/faq

Wangbong, L., Kidong N., Hak-Gyun R., Sang-Ha K. 2016. A Gateway based Fog Computing Architecture for Wireless Sensors and Actuator Networks.

Yehia, L., Khedr, A., & Darwish, A. 2015. Hybrid Security Techniques for Internet of Things Healthcare Applications. SciRes.

Yindong, C., Liping, L., Ziran, C. 2017. An Approach to Verifying Data Integrity for Cloud Storage.