Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) DOI: 10.25126/jtiik.2023107119
Vol. 10, No. 6, Desember 2023, hlm. 1311-1318 p-ISSN: 2355-7699
Akreditasi KEMENRISTEKDIKTI, No. 36/E/KPT/2019 e-ISSN: 2528-6579

IMPLEMENTASI ALGORITME SPONGENT SEBAGAI ALGORITME HASHING UNTUK INTEGRITAS PADA MODUL KOMUNIKASI LORA

Ari Kusyanti[*1], I Putu Agrahita Wiguna[2], Fariz Andri Bakhti[3]

[1,2,3] Universitas Brawijaya, Malang
Email: [1]ari.kusyanti@ub.ac.id, [2]putuagrahita@student.ub.ac.id, [3]fariz@ub.ac.id
[*]Penulis Korespondensi

(Naskah masuk: 03 Maret 2023, diterima untuk diterbitkan: 07 Desember 2023)

Abstrak

Internet of Things (IoT) merupakan konsep terbaru dalam ranah teknologi dan informasi komunikasi dengan fondasinya berdasarkan pada pertukaran informasi antara objek-objek pintar. Komunikasi dalam lingkup IoT memiliki peran penting untuk saling terhubung dengan beberapa perangkat untuk mengirimkan pesan. Teknologi yang secara khusus menargetkan situasi di mana cakupan area yang luas dengan biaya penerapan yang rendah dan konsumsi daya yang rendah menggunakan frekuensi radio ialah LoRa. LoRa dioptimalkan untuk implementasi pada perangkat dengan sumber daya yang terbatas, namun pada penerapan IoT, keamanan data menjadi tantangan selain keterbatasan sumber daya. Pengamanan data dapat dilakukan dengan algoritme *hashing* seperti algoritme SPONGENT. Algoritme SPONGENT dipilih untuk menjamin integritas data. Berdasarkan hasil penelitian, algoritme SPONGENT telah berhasil diimplementasikan sebagai keamanan integritas data pada LoRa. Pengujian terhadap keamanan data dengan pengujian serangan aktif berhasil dilakukan karena ketika penyerang mengirimkan data yang sudah diubah, telah dilakukan pengecekan pada gateway dan data yang dihasilkan tidak valid. Algoritme SPONGENT menunjukkan kinerja yang lebih baik dari Algoritme QUARK berdasarkan kinerja waktu hashing.

Kata kunci: *Internet of Things*, LoRa, Algoritme SPONGENT

THE IMPLEMENTATION OF SPONGENT ALGORITHM AS HASHING MECHANISM ON LORA COMMUNICATION MODULE

Abstract

The Internet of Things (IoT) is a recent concept in the realm of technology and information communication, built on the foundation of information exchange among smart objects. Communication within the scope of IoT has an important role to connect with multiple devices to send messages. Technology that specifically targets situations where large area coverage with low deployment costs and low power consumption use a radio frequency is LoRa. LoRa is optimized for implementation on limited resource devices, but in IoT deployment, data security becomes a challenge in addition to resource limitations. Data security can be done with hashing algorithms such as SPONGENT algorithm. The SPONGENT algorithm was chosen to ensure data integrity. Based on the research results, the SPONGENT algorithm has been successfully implemented as data integrity security in LoRa. Testing of data security with active attack testing is successful because when the attacker sends the modified data, it has been checked on the gateway node and the resulting data is invalid. SPONGENT algorithm shows better performance than QUARK algorithm based on hashing time performance.

Keywords: *Internet of Things, LoRa, SPONGENT Algorithm*

1. PENDAHULUAN

Komunikasi dalam lingkup IoT memiliki peran penting untuk saling terhubung dengan beberapa perangkat untuk mengirimkan pesan. *Long Range* (LoRa) adalah teknologi nirkabel jarak jauh dengan daya rendah yang menggunakan frekuensi radio. Perangkat LoRa dirancang dengan sederhana agar biaya produksinya tetap rendah, di mana pendekatan ini juga membantu mengurangi biaya tambahan yang terjadi selama proses komunikasi karena kurangnya kompleksitas (Sundaram et al., 2019).

LoRa dinilai merupakan teknologi yang sesuai untuk diimplementasikan pada lingkungan IoT dibandingkan dengan WiFi, karena jika dibandingkan dengan IoT berbasis WiFi yang memiliki jangkauan komunikasi data yang pendek dan perlu menambahkan *extender* untuk menambah jangkauan, LoRa mampu memberikan efisiensi dalam jangkauan

komunikasi data dengan tetap menjaga biaya pengembangan yang tetap rendah (Zourmand et al., 2019).

Namun LoRa tidak dapat langsung mengirimkan data menuju *cloud* dikarenakan LoRa merupakan protokol *non-IP based* sehingga tidak dapat terkoneksi dengan internet. *Gateway* dan *cloud* membutuhkan sebuah protokol komunikasi untuk saling berkomunikasi agar dapat saling bertukar informasi.

Lapisan aplikasi pada IoT bertanggung jawab dalam menyediakan layanan dengan meyakinkan komunikasi yang efektif antar perangkat IoT berbiaya rendah dan sumber daya yang terbatas (Sharma et al., 2018). Dalam situasi dengan jumlah pesan dalam ukuran kecil atau sedang dalam jumlah besar, MQTT memiliki performa yang baik daripada AMQP. Penggunaan UDP pada CoAP menyebabkan *throughput* yang lebih rendah dari MQTT, yang juga memiliki perfoma lebih baik daripada AMQP (Glaroudis et al., 2020).

LoRa dioptimalkan untuk implementasi pada perangkat dengan biaya murah yang dijalankan menggunakan baterai dengan sumber daya yang terbatas. Namun dalam penerapan IoT, keamanan data menjadi tantangan lain selain tentang keterbatasan sumber daya (Rahman et al., 2020). Salah satu kelemahan utama dari sistem yang mengadopsi teknologi LoRa adalah kekurangan dalam hal keamanan komunikasi data antar perangkat (Manuel & Daimi, 2021). Keamanan pada IoT sangat penting, karena sebagian besar data yang dikumpulkan pada perangkat IoT bersifat pribadi dan membutuhkan privasi (Hameed & Alomary, 2019).

Kriptografi adalah ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi seperti kerahasiaan, integritas data, serta autentikasi (Munir, 2019). Terdapat banyak algoritme kriptografi yang dapat diterapkan dalam proses pengamanan data. Namun dengan beberapa keterbatasan pada perangkat yang digunakan membuat algoritme kriptografi tradisional sulit diterapkan untuk keamanan informasi (Dhanda et al., 2020).

Penerapan algoritme kriptografi ringan dapat menjadi solusi untuk keamanan data perangkat dengan sumber daya yang terbatas. Algoritme ringan membutuhkan lebih sedikit daya untuk beroperasi dan menghemat banyak energi dari perangkat yang dioperasikan dengan baterai (Gupta & Kumar, 2021). Algoritme SPONGENT termasuk fungsi *hash* ringan berdasarkan konstruksi spons yang digunakan dengan permutasi algoritme PRESENT (Bogdanov et al., 2013). Algoritme SPONGENT memiliki keunggulan di mana keunggulan ini berasal dari tingkat keamanan *second-preimage* yang dikurangi, sambil mempertahankan tingkat standar *collision resistance* (Bogdanov et al., 2013).

Algoritme SPONGENT menjamin aspek integritas pada keamanan data. Penerapan pada penelitian ini dilakukan dengan menggunakan algoritme SPONGENT sebagai algoritme *hashing* untuk integritas pada modul komunikasi LoRa. Algoritme SPONGENT diterapkan untuk melakukan mekanisme pengamanan data pada protokol LoRa.

2. LANDASAN KEPUSTAKAAN

2.1. *Internet of Things* (IoT)

Internet of Things (IoT) merupakan konsep terbaru dalam ranah teknologi dan informasi komunikasi dengan fondasinya berdasarkan pada pertukaran informasi antara objek-objek pintar (Mousavi et al., 2021). IoT memungkinkan benda dan orang dapat terhubung kapan saja dan di mana saja dengan memanfaatkan jalur atau jaringan apa pun. IoT memberikan pengaruh besar terhadap perkembangan teknologi informasi.

2.2. LoRa

Long Range (LoRa) teknologi yang mengatur lapisan fisik untuk komunikasi dengan jangkauan yang panjang dan konsumsi daya yang rendah, serta dapat digunakan pada pita radio tanpa lisensi (Sarker et al., 2019). Lapisan fisik pada teknologi LoRa menggunakan sebuah modulasi *Spread Spectrum* yaitu *Chirp Spread Spectrum* (CSS). *Chirp* adalah sinyal yang memiliki frekuensi bervariasi secara linear dengan waktu dalam *bandwidth* yang tersedia. Hal ini membuat sinyal *chirp* tahan terhadap *noise*, *fading*, dan intereferensi (Sundaram et al., 2019).

2.3. Algoritme SPONGENT

Algoritme SPONGENT termasuk fungsi hash ringan berdasarkan konstruksi spons yang digunakan dengan permutasi algoritme PRESENT . Algoritme SPONGENT memiliki parameter sebagai SPONGENT-n/c/r untuk ukuran *hash* dengan parameter n, kapasitas dengan parameter c, dan *rate* dengan parameter r. Algoritme SPONGENT memiliki keunggulan di mana keunggulan ini berasal dari tingkat keamanan *second-preimage* yang dikurangi, sambil mempertahankan tingkat standar *collision resistance*.

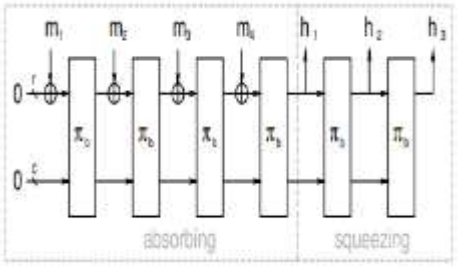
Second-preimage resistance adalah komponen dari fungsi *hash* di mana secara komputasi tidak mungkin menemumakn *input* kedua yang memiliki *output* yang sama dengan *input* yang sudah ditentukan, sebagai contoh diberikan *input* x untuk menemukan *second-preimage* $x' \neq x$ sedemikian rupa sehingga $h(x) \neq h(x')$. *Collision resistance* adalah komponen dari fungsi *hash* di mana secara komputasi tidak mungkin menemukan 2 *input* yang berbeda x, x' yang memiliki *output* yang sama, sehingga $h(x) = h(x')$. *Collision resistance* menyiratkan *second-preimage resistance* dari fungsi *hash*

Selain itu SPONGENT mempertahankan fungsi putaran tetap sederhana, dengan tujuan mengurangi

kompleksitas logika menjadi sekecil mungkin secara teoritis. Hal ini menghasilkan desain yang lebih kompleks dan minimalis

Algoritme SPONGENT bergantung pada konstruksi spons, dengan desain iterasi sederhana yang mengambil *input* dengan panjang variabel dan dapat menghasilkan *ouput* dengan panjang yang berubah-ubah berdasarkan permutasi π_b yang beroperasi pada keadaan b *bits* tetap. Ukuran dari *internal state* $b = r + c \geq n$ disebut *width*, di mana r adalah *rate* dan c adalah *capacity*. Algoritme SPONGENT terdiri dari 45 *round*, di mana fase *absorbing* dilakukan sebanyak 45 kali perulangan untuk menghasilkan *state* (Bogdanov et al., 2013). Konstruksi dari spons dapat dilihat pada Gambar 1 Konstruksi spons berlangsung dalam tiga tahap:

1. Initialization phase
   Pesan diisi dengan satu *bit* 1 diikuti dengan jumlah 0 bit yang diperlukan hingga kelipatan r *bit* (jika r = 8 makan pesan 1-*bit* '0' diubah menjadi '01000000') . Kemudian dipotong menjadi blok r-*bit*.
2. Absorbing phase
   Blok pesan *input* r-*bit* dilakukan proses XOR menjadi *bit* r pertama dari *state*, lalu disisipkan dengan aplikasi permutasi π_b.
3. Squeezing phase
   Bit r pertama dari *state* dikembalikan sebagai *output*, lalu disisipkan dengan aplikasi permutasi π_b, sampai n *bit* dikembalikan.



Gambar 1. Konstruksi Spons
Sumber: (Bogdanov et al., 2013)

Permutasi $\pi_b : \mathbb{F}_2^b \rightarrow \mathbb{F}_2^b$ adalah transformasi putaran R dari *state input* b *bit* yang dapat diuraikan sebagai:

for $i = 1$ *to* R do

$$STATE \leftarrow retnuoCl_b(i) \oplus STATE \oplus lCounter_b(i)$$
$$STATE \leftarrow sBoxLayer_b(STATE)$$
$$STATE \leftarrow pLayer_b(STATE)$$

end for (1)

di mana $sBoxLayer_b$ dan $pLayer_b$ menggambarkan bagaimana state berubah. $lCounter_b$ adalah keadaan LFSR yang bergantung pada b pada waktu i yang menghasilkan putaran konstan di putaran i dan ditambahkan ke bit paling kanan dari *state*. $retnuoCl_b(i)$ adalah nilai $lCounter_b(i)$ dengan bit-bit dalam urutan terbalik dan ditambahkan ke bit-bit *state* paling kiri.

1. $sBoxLayer_b$: ini menunjukkan penggunaan S-box 4-bit hingga 4-*bit* $S : \mathbb{F}_2^4 \rightarrow \mathbb{F}_2^4$ yang diterapkan $b/4$ kali secara paralel.

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S[x]$	E	D	B	0	2	1	4	F	7	A	8	5	9	C	3	6

Gambar 2. Tabel S-Box
Sumber: (Bogdanov et al., 2013)

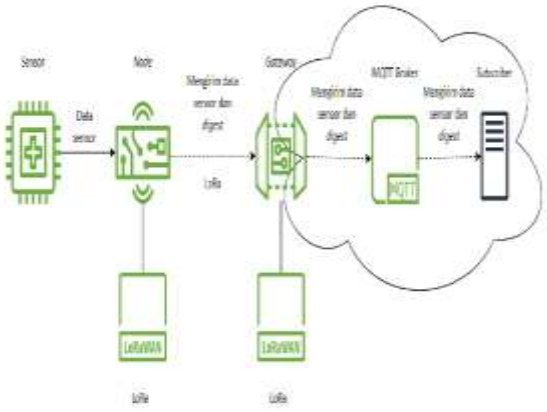
2. $pLayer_b$: merupakan perpanjangan dari (*inverse*) permutasi *bit* PRESENT dan memindahkan bit j dari *state* ke posisi $P_b(j)$ di mana
   $P_b(j) =$
   $$\begin{cases} j \cdot \frac{b}{4} \mod b - 1, jika \in \{0, \dots, b-2\} \\ b-1, \ jika\ j = b-1 \end{cases} \quad (2)$$

3. $lCounter_b$: merupakan salah satu dari empat $[log_2 R]$–bit LFSR. LFSR di-*clock* sekali setiap kali *state*-nya digunakan.

3. PERANCANGAN SISTEM

Tahap perancangan sistem menjelaskan tentang rancangan sistem yang dibuat di mana hasil dari rancangan sistem tersebut akan digunakan untuk proses implementasi. Perancangan penelitian ini terdiri dari gambaran umum sistem, perancangan alur kerja sistem, dan perancangan pengamanan.

3.1. GAMBARAN UMUM SISTEM



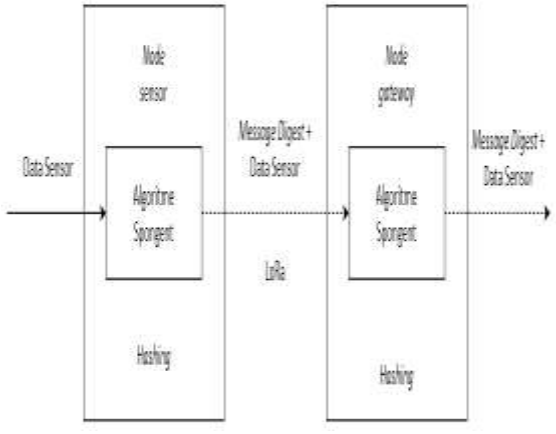
Gambar 3. Gambaran Umum Sistem

Gambar 3 merupakan gambaran umum sistem. Sistem dimulai dengan proses pengambilan data dari sensor detak jantung yang dipasangkan pada tubuh yang kemudian akan diterima oleh *node* sensor.

Setelah *node* sensor mendapatkan data, maka *node* sensor akan melakukan proses *hashing* pada data tersebut sehingga dihasilkan *message digest*. Data sensor dan *message digest* yang didapatkan dari proses hashing akan dikirimkan oleh *node* sensor menuju *node gateway* menggunakan modul komunikasi LoRa. Setelah *node gateway* menerima *message digest* dan data sensor, *node gateway* akan melakukan *hashing* pada data sensor yang telah didapatkan dari *node* sensor dan nilai *message digest* yang dihasilkan oleh *node gateway* akan dibandingkan dengan nilai *message digest* yang diterima dari *node* sensor untuk mengetahui validitas dari nilai *message digest* tersebut. Setelah dilakukan cek validasi dan nilai *message digest* bernilai sama, maka node gateway akan meneruskan nilai *message digest* dan data sensor tersebut menuju *broker* dengan menggunakan protokol MQTT.

Gateway akan melakukan proses *publish* menuju MQTT *broker* dan MQTT *broker* akan melakukan *publish* data sesuai dengan topik apa yang di-*subscribe* oleh *subscriber*. *Broker* akan menerima *message digest* , data sensor dan topik dari *gateway* yang akan dikirimkan menggunakan protokol MQTT. *Broker* akan melakukan *publish* data sesuai dengan topik yang di-*subscribe* oleh *subscriber*. Setelah mendapatkan data sensor dan *message digest*, *subscriber* akan melakukan *hashing* pada data sensor yang telah didapatkan dari proses *subscribe* untuk melakukan validasi. Data sensor, nilai *message digest*, dan topik akan ditampilkan pada *subsciber* jika nilai *message digest* yang dihasilkan oleh *subscriber* bernilai sama dengan nilai *message digest* yang diterima dari *broker*.

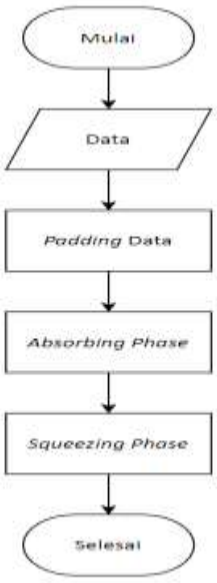
3.2. PERANCANGAN PENGAMANAN



Gambar 4. Rancangan Umum Pengamanan Sistem

Gambar 4 merupakan rancangan umum pengamanan yang dilakukan pada sistem. *Node* akan mengambil data sensor yang berupa detak jantung. Data yang didapatkan akan diamankan menggunakan algortime SPONGENT melalui proses *hashing* yang kemudian akan menghasilkan *message digest*. *Node* akan mengirimkan *message digest* dan data sensor menuju *node gateway* menggunakan komunikasi LoRa. *Gateway* akan mengamankan data sensor yang telah didapat menggunakan algortime SPONGENT untuk menghasilkan *message digest. Message digest* yang diterima dari *node* akan dilakukan pengecekan dengan *message digest* yang telah dihasilkan *node gateway*. Jika menghasilkan nilai yang sama maka bernilai valid.

3.3. PERANCANGAN ALUR KERJA ALGORITME SPONGENT



Gambar 5. Alur Algoritme SPONGENT

Gambar 5 merupakan alur dari algoritme SPONGENT. Proses *hashing* algoritme SPONGENT dimulai dengan masukannya berupa data. Data akan diubah menjadi *bit* dan dilakukan *padding* dan dipotong menjadi beberapa *block* dengan besar 8 *bit*. Setelah dilakukan *padding*, akan memasuki fase *absorbing*, di mana pada fase ini terdapat proses LFSR, substitusi dengan menggunakan *sbox* dan permutasi dengan menggunakan *pLayer*. Algoritme SPONGENT terdiri dari 45 *round*, di mana fase absorbing dilakukan sebanyak 45 kali perulangan untuk menghasilkan *state*. Setelah fase *absorbing* telah selesai, akan dilanjutkan dengan fase *squeezing*. 8 *bit* pertama dari *state* akan dikembalikan sebagai ouput, lalu disisipkan pada proses substitusi-permutasi hingga 88 bit dikembalikan.

3.4. PERANCANGAN PENGUJIAN

Perancangan pengujian menjelaskan terkait dengan pengujian apa saja yang dilakukan pada penelitian ini. Pengujian pada penelitian ini meliputi pengujian kinerja algoritme, dan pengujian keamanan.

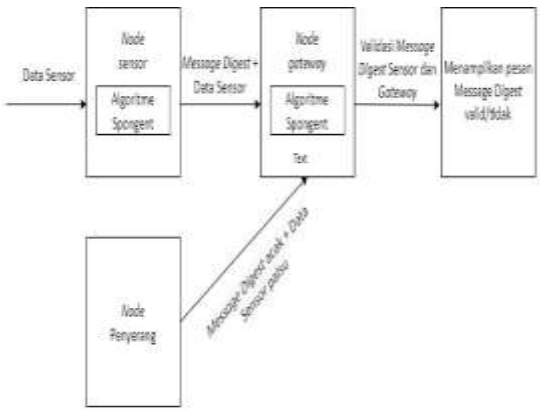
Salah satu tantangan yang menjadi fokus utama bagi pengguna dan produsen perangkat IoT adalah masalah keamanan data. Setiap perangkat IoT perlu

selalu terhubung dengan *Internet* yang menyiratkan penggunaan jaringan publik, dan ini meningkatkan kerentanannya terhadap berbagai serangan berbahaya (Caraveo-Cacep et al., 2023).

Banyak perangkat IoT dirancang dengan ukuran kecil dan melakukan pemrosesan yang lebih sedikit karena harus beroperasi pada lingkungan yang di mana pengisian daya yang teratur tidak memungkinkan (Garg & Kumar Singh, 2021). Penggunaan baterai perlu digunakan pada kondisi dengan lingkungan seperti ini, namun komputasi yang lebih banyak berarti konsumsi baterai yang lebih tinggi.

Pengujian kinerja algoritme dilakukan dengan menghitung lama waktu *hashing* yang dilakukan oleh sebuah algoritme dalam menjalankan sistem pada penelitian ini. Kinerja algoritme SPONGENT akan dibandingkan dengan algoritme lain sebagai pembanding. Algoritme pembanding yang dipilih ialah algoritme QUARK. Dasar pemilihan algoritme QUARK dikarenakan merupakan algoritme *hashing* yang menggunakan konstruksi spons dalam melakukan *hashing* yang sama seperti algoritme SPONGENT.

Pengujian keamanan algoritme dilakukan untuk menguji aspek integritas pada sistem yang dibuat. Pengujian ini terdiri dari beberapa perangkat yang berperan sebagai *node* sensor, *node gateway* dan *node* penyerang.



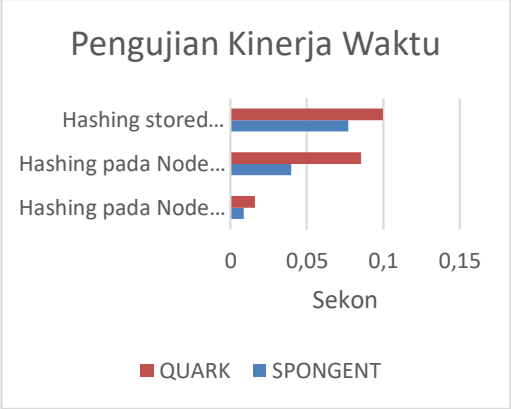
Gambar 6. Arsitektur Pengujian Keamanan

Gambar 6 merupakan arsitektur pengujian keamanan. Pengujian ini menggunakan *node* tambahan yaitu *node* penyerang, di mana *node* ini akan melakukan penyerangan dengan cara mengirimkan data palsu berupa data ecg dan *message digest* yang dibuat acak ketika *node* sensor dan *node gateway* sedang melakukan komunikasi. Ketika data palsu dikirimkan dan diterima oleh *node gateway*, *node gateway* akan melakukan pengecekan untuk memastikan bahwa *message digest* yang diterima bernilai sama dengan yang dihasilkan oleh *gateway*. Jika data bernilai tidak sama, maka dipastikan bahwa data yang dikirimkan telah mengalami perubahan oleh pihak yang tidak berwenang.

4. PENGUJIAN DAN PEMBAHASAN

4.1. PENGUJIAN KINERJA

Pengujian ini akan menjalankan dua jenis sistem, yaitu sistem dengan algoritme SPONGENT dan sistem dengan algoritme QUARK. Pengujian ini dilakukan untuk mengukur lama waktu yang dibutuhkan untuk *hashing* saat menjalankan sistem.



Gambar 7. Hasil Pengujian Kinerja Waktu

Gambar 7 menunjukkan hasil pengujian kinerja berdasarkan waktu hashing pada *node* sensor dan *node gateway*. Hasil pengujian kinerja waktu didapatkan dengan melakukan uji sebanyak 30 kali. Waktu rata-rata yang diperlukan algoritme SPONGENT dalam melakukan *hashing* pada *node* sensor adalah 0,008643067 sekon, sedangkan waktu yang diperlukan algoritme QUARK dalam melakukan *hashing* pada *node* sensor adalah 0,0160912 sekon. Algoritme SPONGENT lebih cepat dalam melakukan *hashing* pada *node* sensor dengan selisih 0,007448133 sekon.

Waktu rata-rata yang diperlukan algoritme SPONGENT dalam melakukan *hashing* pada *node gateway* adalah 0,039795 sekon, sedangkan waktu yang diperlukan algoritme QUARK dalam melakukan *hashing* pada *node gateway* adalah 0,085421 sekon. Algoritme SPONGENT lebih cepat dalam melakukan *hashing* pada *node gateway* dengan selisih 0,045626 sekon.

Waktu rata-rata yang diperlukan algoritme SPONGENT dalam melakukan *hashing stored* data pada *node gateway* adalah 0,076944 sekon, sedangkan waktu yang diperlukan algoritme QUARK dalam melakukan *hashing stored* data pada *node gateway* adalah 0,099652 sekon. Algoritme SPONGENT lebih cepat dalam melakukan *hashing stored* data pada *node gateway* dengan selisih 0,022708 sekon.

4.2. PENGUJIAN KEAMANAN

Pengujian keamanan ini menggunakan serangan aktif dengan menyisipkan data palsu saat node sensor mengirimkan data asli menuju node gateway selama proses pengiriman data berlangsung. Data yang

dikirimkan berupa data sensor serta *message digest* yang dibuat secara acak oleh *node* penyerang. Prosedur pengujian ini dilakukan dengan membuat *node* palsu yang berperan sebagai *node* penyerang. *Node* penyerang akan mengirimkan data sensor dan *message digest* palsu menuju *node gateway* ketika komunikasi antara *node* sensor dengan *node gateway* sedang berlangsung, sehingga data yang dikirimkan oleh node penyerang akan dianggap sebagai data yang dikirimkan oleh *node* sensor. Fokus *node* penyerang hanya menyisipkan data palsu sehigga *node gateway* akan mendapatkan data yang berisi data sensor ECG dan *message digest* yang menyerupai data yang dikirimkan oleh *node* sensor yang asli.

```
String sensor, digest;
sensor = "507";
digest = "58CF970560616011879948";
kirim(sensor, digest);
```

Gambar 8. Data yang Dikirimkan Penyerang

Gambar 8 merupakan data yang dikirimkan oleh penyerang. Penyerang membuat data palsu berupa data sensor ECG dan *message digest* yang dibuat secara acak seperti yang dikirimkan *node* sensor asli menuju *gateway*. Data tersebut akan dikirimkan menuju *node gateway* dengan modul komunikasi LoRa, sehingga *node gateway* akan menerima data yang sama dengan yang dikirimkan *node* asli.

```
COM9

Waktu Mulai(Mikro detik): 839452965
Message (String):285
Waktu sebelum hash(Mikro detik): 839453045
Hash: B02B479DF1AAFF4840CD11
Waktu sesudah hash(Mikro detik): 839461181
Waktu hash(Mikro detik): 8136
Nilai: 285
Nilai: B02B479DF1AAFF4840CD11
 success!
Datarate:       777.75 bps

Waktu selesai(Mikro detik): 839875866
Waktu(Mikro detik): 422901
```

Gambar 9. Pengiriman Data Asli oleh *Node* Sensor

Gambar 9 Merupakan pengiriman data asli oleh *node* sensor. Data yang didapatkan dari sensor ECG menunjukkan nilai 285. Kemudian data tersebut diubah menjadi *message digest* menggunakan algoritme SPONGENT. Data sensor dan *message digest* kemudian dikirimkan menuju *node gateway*.

```
COM3

success!
Nilai Sensor: 285
Nilai digest: B02B479DF1AAFF4840CD11
Message (String):285
Waktu sebelum hash(Mikro detik): 14370442
Hash: B02B479DF1AAFF4840CD11
Waktu sesudah hash(Mikro detik): 14502393
Waktu hash(Mikro detik): 131951

Nilai Digest Valid
nilai awal:
Store: 285
String: 285
Length: 3
Waktu selesai(Mikro detik): 14625171
Waktu(Mikro detik): 702928
```

Gambar 10. Penerimaan Data Asli pada *Node Gateway*

Node gateway menerima data sensor dan *message digest* yang ditunjukkan pada Gambar 10 bahwa *node gateway* menerima data ECG dengan nilai 285, di mana data yang diterima sama dengan yang dikirimkan oleh *node* sensor asli. Kemudian data sensor yang diterima digunakan sebagai masukkan untuk menghasilkan *message digest* pada sisi *node gateway*. Gambar 10 juga menujukkan bahwa nilai *message digest* yang diterima oleh *gateway* dan yang dihasilkan oleh *node gateway* bernilai sama sehingga bernilai valid.

```
COM3

success!
Nilai Sensor: 507
Nilai digest: 58CF970560616011879948
Message (String):507
Waktu sebelum hash(Mikro detik): 246916056
Hash: C44BE0CC128338115AFF9F1
Waktu sesudah hash(Mikro detik): 247047783
Waktu hash(Mikro detik): 131727

Nilai Digest Tidak Valid
Waktu selesai(Mikro detik): 247170552
Waktu(Mikro detik): 837933
```

Gambar 11. Penerimaan Data Palsu pada *Node Gateway*

Gambar 11 merupakan penerimaan data palsu pada *node gateway*. Serangan yang dilakukan oleh penyerang menyebabkan *node gateway* menerima data yang berbeda. Berdasarkan Gambar 9 data yang dikirimkan oleh *node* sensor asli menujukkan data ECG bernilai 285 serta hasil *message digest*. Gambar 10 juga menunjukkan bahwa *node gateway* menerima data ecg yang bernilai 285 dan hasil perbandingan *message digest* menujukkan *message digest* bernilai valid. Namun ketika penyerang berhasil melakukan serangan dengan mengirimkan data palsu, hasil akhir perbandingan menjadi bernilai tidak valid karena terdapat perbedaan hasil *message digest* yang diterima oleh *gateway* dengan *message digest* yang dihasilkan oleh *gateway*. Hal ini menunjukkan bahwa

kinerja keamanan yang dilakukan terhadap aspek integritas pada data berhasil dilakukan.

5. KESIMPULAN

Penelitian ini telah menghasilkan implementasi algoritme SPONGENT sebagai algoritme *hashing* untuk integritas pada modul komunikasi LoRa, dapat disimpulkan bahwa implementasi algoritme SPONGENT untuk mengamankan integritas data pada modul komunikasi LoRa berhasil diimplementasikan. Algoritme SPONGENT yang diimplementasikan terbukti mendapatkan hasil bahwa sistem berhasil melakukan pengecekan integritas data terhadap serangan pihak ketiga dengan hasil akhir yang tidak valid.

DAFTAR PUSTAKA

BOGDANOV, A., KNEŽEVIĆ, M., LEANDER, G., TOZ, D., VARICI, K., & VERBAUWHEDE, I. 2013. SPONGENT: The design space of lightweight cryptographic hashing. *IEEE Transactions on Computers*, *62*(10), 2041–2053. https://doi.org/10.1109/TC.2012.196

CARAVEO-CACEP, M. A., VÁZQUEZ-MEDINA, R., & HERNÁNDEZ ZAVALA, A. 2023. A survey on low-cost development boards for applying cryptography in IoT systems. *Internet of Things (Netherlands)*, *22*(March), 100743. https://doi.org/10.1016/j.iot.2023.100743

DHANDA, S. S., SINGH, B., & JINDAL, P. 2020. Lightweight Cryptography: A Solution to Secure IoT. In *Wireless Personal Communications* (Vol. 112, Issue 3). Springer US. https://doi.org/10.1007/s11277-020-07134-3

GARG, P., & KUMAR SINGH, D. 2021. Analysis of cryptographic encryption algorithm design to Secure IoT Devices: A review. *Materials Today: Proceedings*, *51*(xxxx), 810–814. https://doi.org/10.1016/j.matpr.2021.06.240

GLAROUDIS, D., IOSSIFIDES, A., & CHATZIMISIOS, P. 2020. SURVEY, COMPARISON AND RESEARCH Challenges of IoT application protocols for smart farming. *Computer Networks*, *168*, 107037. https://doi.org/10.1016/j.comnet.2019.107037

GUPTA, D. N., & KUMAR, R. 2021. Sponge based Lightweight Cryptographic Hash Functions for IoT Applications. *2021 International Conference on Intelligent Technologies, CONIT 2021*, 9–13. https://doi.org/10.1109/CONIT51480.2021.9498572

HAMEED, A., & ALOMARY, A. 2019. Security issues in IoT: A survey. *2019 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies, 3ICT 2019*. https://doi.org/10.1109/3ICT.2019.8910320

MANUEL, M. P., & DAIMI, K. 2021. Implementing cryptography in LoRa based communication devices for unmanned ground vehicle applications. *SN Applied Sciences*, *3*(4), 1–14. https://doi.org/10.1007/s42452-021-04377-y

MOUSAVI, S. K., GHAFFARI, A., BESHARAT, S., & AFSHARI, H. 2021. Security of internet of things based on cryptographic algorithms: a survey. In *Wireless Networks* (Vol. 27, Issue 2). Springer US. https://doi.org/10.1007/s11276-020-02535-5

MUNIR, R. 2019. *Kriptografi* (2nd ed.). Informatika Bandung.

RAHMAN, A. G. A., PRAMUKANTORO, E. S., & AMRON, K. 2020. *Implementasi Mekanisme End-To-End Security Menggunakan Algoritma AES dan HMAC pada Pengiriman Data Sensor ECG Berbasis LoRa*. *4*(1), 166–173.

SARKER, V. K., QUERALTA, J. P., GIA, T. N., TENHUNEN, H., & WESTERLUND, T. 2019. A survey on LoRa for IoT: Integrating edge computing. *2019 4th International Conference on Fog and Mobile Edge Computing, FMEC 2019*, 295–300. https://doi.org/10.1109/FMEC.2019.8795313

SHARMA, C., MATA, S., & DEVI, V. 2018. Constrained IoT Systems. *2018 3rd International Conference On Internet of Things: Smart Innovation and Usages (IoT-SIU)*, 1–6.

SUNDARAM, J. P. S., DU, W., & ZHAO, Z. 2019. A Survey on LoRa Networking: Research Problems, Current Solutions, and Open Issues. *IEEE Communications Surveys and Tutorials*, *22*(1), 371–388. https://doi.org/10.1109/COMST.2019.2949598

ZOURMAND, A., KUN HING, A. L., WAI HUNG, C., & ABDULREHMAN, M. 2019. Internet of Things (IoT) using LoRa technology. *2019 IEEE International Conference on Automatic Control and Intelligent Systems, I2CACIS 2019 - Proceedings*, *June*, 324–330. https://doi.org/10.1109/I2CACIS.2019.8825008

Halaman ini sengaja dikosongkan.