Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer e-ISSN: 2548-964X
Vol. 5, No. 5, Mei 2021, hlm. 2050-2057 http://j-ptiik.ub.ac.id

Implementasi Algoritme *PRESENT* untuk Proses Enkripsi pada Modul Komunikasi LoRa

Fitra Firdaus[1], Ari Kusyanti[2], Reza Andria Siregar[3]

Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya
Email: [1]fitraf56@student.ub.ac.id, [2]ari.kusyanti@ub.ac.id, [3]reza.jalin@ub.ac.id

Abstrak

Internet of Things (IoT) pada dasarnya merupakan gabungan dari dua komponen utama, yaitu internet dan *things*. Dalam mewujudkan visi IoT, teknologi *Low Power Wide Area (LPWA) networks* menarik banyak perhatian karena kemampuannya untuk menawarkan konektivitas yang terjangkau untuk perangkat berdaya rendah yang tersebar pada cakupan area yang luas *(long range).* LoRa adalah teknologi nirkabel *unlicensed* berdaya rendah *(low battery consumption)* yang memiliki jangkauan jarak yang cukup jauh dengan menggunakan teknik *chirp-spread-spectrum(CSS),* namun LoRa belum memiliki pengamanan pada proses pengiriman data. Dengan tidak adanya pengamanan pada LoRa, maka sangat rentan terhadap serangan oleh pihak yang tidak bertanggung jawab diantaranya adalah serangan *sniffing*. Untuk dapat menerapkan sistem keamanan pada data yang dikirimkan diantaranya adalah dengan menggunakan algoritme enkripsi *PRESENT*. Hal ini dibuktikan dengan hasil pengujian pada penelitian dengan melakukan serangan pasif menggunakan teknik *sniffing* serta serangan aktif menggunakan teknik *Known Plaintext Attack (KPA).* Pada serangan *sniffing* penyerang hanya mendapatkan data yang terenkripsi dalam bentuk *chipertext* sehingga data tidak dapat dibaca. Kemudian untuk hasil serangan KPA, penyerang tidak berhasil mendapatkan *key* algoritme. Dari hasil uji dengan menggunakan serangan *sniffing* dan KPA, maka dapat diambil sebuah kesimpulan bahwa implementasi algoritme *PRESENT* berhasil dalam melakukan pengamanan data yang dikirimkan menggunakan Lora.

Kata kunci: *Algoritme PRESENT, LoRa, Internet of Things.*

Abstract

Internet of Things (IoT) is basically a combination of two main components, namely the internet and things. To realize the vision of IoT, Low Power Wide Area (LPWA) networks are attracting a lot of attention mainly because of their ability to offer affordable connectivity for low-power devices that are spread over a wide area (long range). LoRa is a low power unlicensed wireless technology (low battery consumption) that has a long range using the chirp-spread-spectrum (CSS) technique, but LoRa does not have any security in the data transmission process. With no security at LoRa, it is very vulnerable to attacks by irresponsible parties, one of which is a sniffing attack. To be able to apply the security system to the data sent among others is to use the encryption algorithm PRESENT. It's evidenced by the results of testing in research conducted by performing passive attacks using sniffing techniques and active attacks using Known Plaintext Attack (KPA) techniques. In a sniffing attack, the attacker only gets encrypted data in the form of ciphertext so that the data cannot be read. Then for the results of the attack with KPA, the attacker did not succeed in obtaining the algorithm key. From the test results using sniffing attacks and KPA, it can be concluded that the implementation of the PRESENT algorithm was successful in securing the data sent using Lora.

Keywords: *PRESENT Algorithms, LoRa, Internet of Things.*

1. PENDAHULUAN

Kemajuan teknologi pada era modern saat ini berkembang sangat pesat khususnya pada pemanfaatan teknologi internet, salah satu diantaranya adalah Internet of Things (IoT). Istilah IoT pertama kali diperkenalkan oleh Kevin Ashton pada tahun 2009 yang sebenarnya 10 tahun sebelumnya ia sudah memaparkan istilah tersebut pada sebuah presentasi di Procter

& Gambler (P&G) (Ashton, 2009). Pada dasarnya IoT terdiri dari dua buah komponen utama, yaitu internet dan *things*. Internet adalah jaringan komputer yang saling terhubung satu sama lain dalam skala yang besar dengan standar protokol komunikasi tertentu sehingga bisa saling berkomunikasi, berinteraksi, dan bertukar informasi. *Things* adalah perangkat yang memiliki identitas, atribut, serta karakteristik tertentu yang dapat berkomunikasi satu sama lain melalui sebuah media transmisi (Pramukantoro , et al., 2019).

Dalam mewujudkan visi IoT, teknologi *Low Power Wide Area (LPWA) networks* menarik banyak perhatian terutama karena kemampuannya untuk menawarkan konektivitas yang terjangkau untuk perangkat berdaya rendah yang tersebar pada cakupan area yang luas *(long range)* (Raza, et al., 2017). Salah satu teknologi yang menerapkan konsep LPWA *networks* adalah LoRa. LoRa adalah teknologi nirkabel *unlicensed berdaya* rendah *(low battery consumption)* yang memiliki jangkauan jarak yang cukup jauh dengan menggunakan teknik *chirp-spread-spectrum(CSS)* (Wixted , et al., 2016). Teknik CSS sebenarnya dikembangkan untuk aplikasi radar pada tahun 1940 dan telah umum digunakan oleh militer, namun beberapa tahun belakangan ini telah mulai diadopsi oleh beberapa aplikasi komunikasi karena kebutuhan daya yang relatif rendah dan ketahanan terhadap interferensi. Dari berbagai macam kelebihan yang dimiliki, LoRa belum memiliki pengamanan pada proses pengiriman data (Aras, et al., 2017).

Pada penelitian sebelumnya yang telah dilakukan oleh Arijuddin, et al., pada tahun 2019 yang berjudul "Pengembangan Sistem Perantara Pengiriman Data Menggunakan Modul Komunikasi LoRa dan Protokol MQTT Pada *Wireless Sensor Network*" telah dikembangakan sebuah sistem *gateway* sebagai perantara pengiriman data antara modul komunikasi LoRa dengan protokol MQTT. Namun, skema komunikasi pada *gateway* tersebut masih terdapat celah keamanan, karena belum diterapkannya sistem keamanan pada transmisi data. Celah keamanan tesebut dapat mengakibatkan sebuah ancaman serangan dari sisi *confidentiality*, yaitu *eavesdropping* (Zou, et al., 2016). Dilihat dari hal tersebut maka dibutuhkan sebuah metode pengamanan data berupa enkripsi dengan mempertimbangkan efisiensi serta daya yang dibutuhkan.

PRESENT adalah sebuah algoritme *Ultra-lightweight block cipher* dengan panjang *block* 64 bit dan *key* 80 bit atau 128 bit yang resmi distandarisasi oleh ISO/IEC sebagai algoritme untuk kriptografi ringan. Penelitian yang menjadi rujukan utama dari algoritme *PRESENT* ini adalah penelitian yang dilakukan oleh Bogdanov, A. et al., (2007) yang berjudul " *PRESENT: An Ultra-Lightweight Block Cipher* " meneliti mengenai algoritma baru yang mereka buat dengan tujuan membuat algoritme yang sangat ringan *(Ultra-Lightweight)* dengan menawarkan tingkat keamanan yang sepadan dengan ukuran *block* 64 bit dan *key* 80 bit.

Dengan melakukan implementasi algoritme *PRESENT* pada data yang dikirimkan dengan menggunakan modul komunikasi LoRa, diharapkan ancaman serangan dari sisi *confidentiality* dapat diantisipasi, sehingga keamanan data menjadi lebih terjamin. Apabila ada pihak yang tidak bertanggung jawab melakukan serangan dalam bentuk *sniffing*, maka data akan tetap aman karena data asli tetap tidak akan terlihat.

2. LANDASAN KEPUSTAKAAN

2.1. Kajian Literatur

Pada penelitian Pengembangan Sistem Perantara Pengiriman Data Menggunakan Modul Komunikasi LoRa dan Protokol MQTT pada WSN yang disusun oleh Arijuddin, et al., pada tahun 2019, melakukan penelitian menggunakan LoRa untuk proses pengiriman data dari sensor dengan mengembangkan sebuah gateway. *Gateway* ini digunakan sebagai perantara pengiriman data antara modul komunikasi LoRa dengan protokol MQTT. LoRa tidak dapat langsung mengirimkan data yang diterima dari sensor ke *server*, sehingga dibutuhkan sebuah perantara untuk mengirimkan data tersebut dengan mengembangkan sebuah gateway untuk meneruskan ke *server* menggunakan protokol MQTT.

Dalam penelitian *"PRESENT: An Ultra-Lightweight Block Cipher"* yang dilakukan pada tahun 2007 oleh Bogdanov, et al., (2007), peneliti berhasil menciptakan sebuah algoritme *block cipher* baru yang sangat ringan dengan panjang block 60 bit dan *key* 80 bit atau 128 bit dengan jumlah *round* sebanyak 31. Algoritme yang dihasilkan adalah algoritme *PRESENT*. Algoritme *PRESENT* memberikan sebuah terobosan baru mengenai teknik enkripsi pada

perangkat dengan sumber daya terbatas. Sebelumnya untuk penggunaan algoritme enkripsi, AES menjadi salah satu algoritme yang paling disukai. Bahkan sejak adanya AES, kebutuhan untuk *block cipher* baru sangat berkurang. Namun seiring berkembangya teknologi, maka AES tidak cocok untuk digunakan pada sistem dengan *environments* yang terbatas, seperti pada RFID *tags* ataupun jaringan sensor (Bogdanov, et al., 2007).

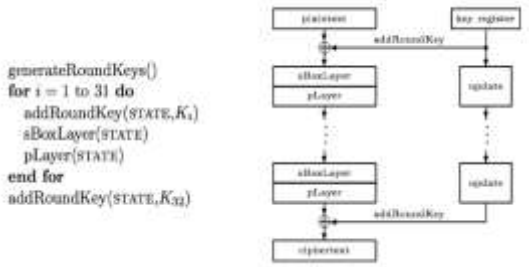
Pada penelitian yang dilakukan oleh Wixted , et al., (2016) dengan judul *Evaluation of LoRa and LoRaWAN for Wireless Sensor Networks*, meneliti mengenai evaluasi kemampuan kinerja teknologi LoRa pada *Wireles Sensor Network*. Evaluasi yang dilakukan diantaranya yaitu testing *wireless performance*, serta *reliability*. Dari hasil evaluasi yang dilakukan, hasilnya menggembirakan karena teknologi LoRa dapat diandalkan untuk *low cost remote sensing applications* dengan cakupan area yang jauh *(long range).*

2.2. LoRa

LoRa adalah teknologi nirkabel *unlicensed* berdaya rendah *(low battery consumption)* yang memiliki jangkauan jarak yang cukup jauh dengan menggunakan teknik *chirp-spread-spectrum(CSS)* (Wixted , et al., 2016). LoRa merupakan salah satu contoh teknologi *Low Power Wide Area (LPWA) networks* dengan memiliki keunggulan *low data rate, low power, low cost* dan *long range* sensor *applications*. *Low Power Wide Area (LPWA) networks* mampu untuk menawarkan konektivitas yang terjangkau untuk perangkat berdaya rendah yang tersebar pada cakupan area yang luas *(long range)* (Raza, et al., 2017). LoRa menggunakan sebuah protokol *proprietary* yang dikembangkan dan dikomersilkan oleh Semtech.

2.3. Algoritme *PRESENT*

Algoritme PRESENT merupakan salah satu algoritme *block cihper* yang menggunakan konsep *SP-network* dan terdiri dari 31 *round* (Bogdanov, et al., 2007). Algoritme *PRESENT* adalah sebuah algoritme *Ultra-lightweight block chiper* dengan panjang *block* 64 bit dan *key* 80 bit atau 128 bit yang resmi distandarisasi oleh ISO/IEC sebagai algoritme untuk kriptografi ringan. Pada algoritme *PRESENT* terdapat dua jenis pilihan *key* yang digunakan, yaitu 80 bit dan 128 bit.



Gambar 1. Algoritme *PRESENT*

Pada algoritme *PRESENT* terdiri dari 31 *round*, dimana pada tiap *round* terdiri dari operasi XOR untuk memperkenalkan sebuah *round key*, untuk setiap nilai i yaitu $1 \leq i \leq 32$ dan pada K32 digunakan sebagai *post-whitening*. Dalam proses enkripsi algoritme *PRESENT* secara garis besar terdapat 3 buah proses penting, yang pertama adalah *addRoundKey*, *sBoxLayer* dan *pLayer*. Pada tahapan *addRoundKey* ini melakukan XOR antara *plaintext* yang dijadikan menjadi bit yang terdiri dari 64 bit (satu *block)* dengan *key* 80 bit. Kemudian selanjutnya kita masukkan ke dalam tabel *s-Box* menjadi masing masing 4 bit *words* sehingga menjadi sebuah *state* nilai baru. Selanjutnya hasil dari *state* yang sudah dimasukkan ke dalam *s-Box* dilakukan *moving* sesuai dengan tabel *pLayer* sehingga menghasilkan nilai enkripsi di setiap *round*.

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
S(x)	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

Gambar 2. Tabel *s-Box*

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
P(i)	0	16	32	48	1	17	33	49	2	18	34	50	3	19	35	51
i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
P(i)	4	20	36	52	5	21	37	53	6	22	38	54	7	23	39	55
i	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
P(i)	8	24	40	56	9	25	41	57	10	26	42	58	11	27	43	59
i	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
P(i)	12	28	44	60	13	29	45	61	14	30	46	62	15	31	47	63

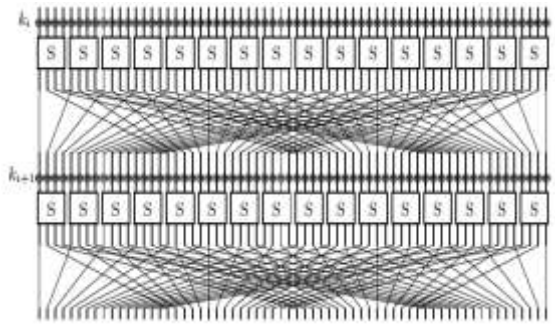
Gambar 3. Tabel *bit permutation* (*pLayer*)

Selanjutnya dari hasil enkripsi tiap *round* sebelumnya, maka untuk melakukan *round* berikutnya dibutuhkan sebuah *key* yang baru, atau dikenal dengan teknik *key scheduling*. Pada *key scheduling* terdapat 3 tahapan utama, diantaranya adalah dengan melakukan pergeseran posisi sebanyak 61 bit kekiri, kemudian 4 bit barisan yang paling kiri dimasukkan ke dalam tabel *s-Box* dan pada 5 bit dari bit *k19 k18 k17 k16 k15* dilakukan XOR dengan *round_counter value i*.

1. $[k_{79}k_{78}\ldots k_1k_0] = [k_{18}k_{17}\ldots k_{20}k_{19}]$
2. $[k_{79}k_{78}k_{77}k_{76}] = S[k_{79}k_{78}k_{77}k_{76}]$
3. $[k_{19}k_{18}k_{17}k_{16}k_{15}] = [k_{19}k_{18}k_{17}k_{16}k_{15}] \oplus round\_counter$

Gambar 4. Key Scheduling

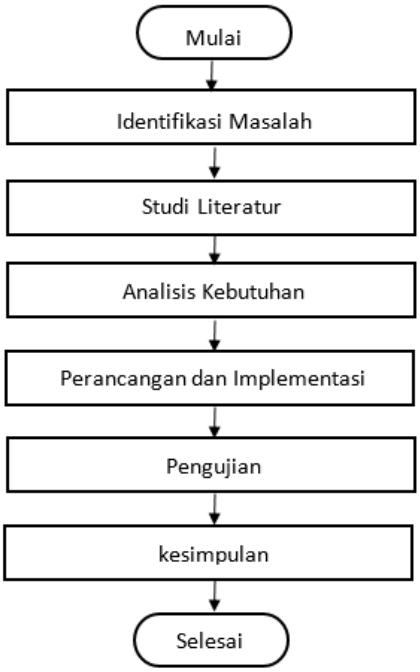
Pada tahapan pertama *key scheduling*, dilakukan pergeseran posisi sebanyak 61 bit ke kiri, sehingga membentuk sebuah posisi baru dari tiap-tiap bit nya. Jika kita analogikan total 80 bit tadi kedalam sebuah *array*, maka posisi *key* dari bit yang ke-79, akan menempati sebuah posisi baru pada *index* ke-18, kemudian posisi *key* dari bit ke-78 ia akan menempati posisi baru pada *index* ke-17, dan begitu seterusnya. Kemudian setelah membentuk posisi baru pada tahapan kedua nilai 4 bit yang paling kiri dimasukkan ke dalam tabel *s-Box*, yang artinya 4 bit yang paling kiri tersebut adalah bit pada index ke-79 sampai ke-76 dari hasil posisi yang baru. Selanjutnya akan menghasilkan sebuah nilai baru dari nilai *s-Box*, dan pada tahapan ketiga adalah melakukan XOR pada 5 bit dari bit *k19k18k17k16k15* dengan *round_counter value i*. Dari hasil XOR tersebut akan membentuk nilai baru yang merupakan nilai *key final* dari sebuah *round*.



Gambar 5. S/P network Algoritme PRESENT

3. METODOLOGI PENELITIAN

Penelitian akan dilakukan dengan tahapan sesuai dengan diagram alir pada Gambar 6, penelitian diawali dengan melakukan identifikasi masalah, studi literatur, analisis kebutuhan, perancangan dan implementasi, pengujian dan kesimpulan.



Gambar 6. Diagram Alir Metodologi Penelitian

3.1. Analisis Kebutuhan

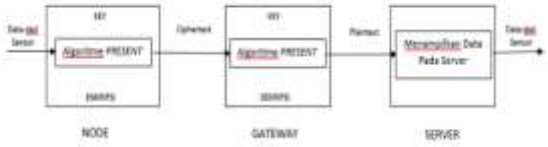
Analisis kebutuhan bertujuan untuk mengetahui semua kebutuhan yang akan diterapkan dalam penelitian ini. Tahap awal untuk melakukan analisis kebutuhan dalam penelitian ini adalah dengan mengidentifikasi kebutuhan fungsional dan non-fungsional, kebutuhan fungsional diantaranya adalah membahas berbagai kebutuhan yang mampu dilakukan oleh sistem, kemudian kebutuhan non-fungsional membahas segala kebutuhan pembangunan sistem sebagai batasan terhadap sistem yang dibangun.

3.2. Perancangan Sistem

Perancangan sistem dilakukan setelah mendapatkan hasil analisis kebutuhan. Tahap awal perancangan adalah dengan melakukan pembuatan diagram alir dan perancangan mengenai sistem yang akan dibuat. Perancangan diagram alir dilakukan untuk memberikan gambaran mengenai proses pengamanan data pada sudut pandang perangkat lunak melalui sebuah proses enkripsi dan dekripsi. Sistem yang digunakan menggunakan modul komunikasi LoRa terdiri dari *node* dan *gateway*. Pada *node* memiliki sebuah ID yang diatur secara manual dan pada *gateway* diatur ID berapa yang dimiliki *node*. Sehingga koneksi antara *node* dan *gateway* akan terjadi ketika ID dari keduanya sesuai.

3.2.1 Perancangan Pengamanan

Perancangan pengamanan berfungsi untuk memberikan keamanan dari aspek *confidentiality*, sehingga apabila penyerang melakukan *eavesdropping* maka data yang dikirim tetap aman dan tidak dapat dibaca oleh penyerang. Pada sisi *node* akan ditambahkan proses enkripsi dengan menggunakan algoritme *PRESENT* lalu pada *gateway* akan dilakukan proses dekripsi sehingga data dapat dibaca kembali dan dikirim ke *server* dalam bentuk data yang dapat dibaca. Sehingga dapat dipastikan bahwa hanya *gateway* yang sah dan memiliki akses yang dapat membaca data yang dikirim oleh node sensor.



Gambar 7. Perancangan Umum Pengamanan

3.2.2 Perancangan Skenario Pengujian

Pada skenario pengujian akan dilakukan beberapa uji coba pengujian untuk memastikan keberhasilan dari penelitian ini. Pengujian dilakukan berdasarkan skenario implementasi sistem dan implementasi algoritme. Pengujian yang akan dilakukan diantaranya adalah pengujian *test vector*, pengujian fungsional, pengujian kinerja, pengujian serangan pasif, dan pengujian serangan aktif.

Perancangan pengujian *test vector* dilakukan dengan menjalankan algoritme *PRESENT* dengan memastikan *key* dan *plaintext* sesuai dengan skenario yang dibuat oleh pencipta algoritme *PRESENT*. Untuk memastikan hasil dari implentasi sesuai dan valid maka *output* yang dihasilkan harus sesuai dengan skenario dari pencipta algoritme *PRESENT*.

Perancangan pengujian fungsional dilakukan untuk memastikan sistem yang diimplementasikan sudah sesuai. Pada pengujian fungsional akan dilakukan sebuah validasi apakah sistem yang berjalan sudah sesuai secara fungsionalitasnya berdasarkan pada perancangan. Jika sistem sudah berjalan sesuai perancangan pengujian fungsional maka sistem dapat dinyatakan valid.

Perancangan pengujian serangan pasif memiliki tujuan utama untuk memastikan tingkat *confidentiality* dari sistem yang telah dibuat dengan menggunakan algoritme *PRESENT*. Pengujian serangan pasif dilakukan menggunakan teknik *sniffing*, dengan menggunakan skenario *gateway* palsu. Pada serangan pasif ini penyerang hanya memiliki sejumlah informasi mengenai algoritme yang digunakan pada sistem namun tidak memiliki informasi lainnya termasuk ID sistem, *key* algoritme, maupun nilai dari *chipertext*. Penyerangan dilakukan dengan mencoba mencari ID dari sistem dengan melakukan percobaan berulang-ulang untuk mendapatkan ID yang cocok (menggunakan teknik *brute force*) dengan memanfaatkan karakter pengiriman dari lora yaitu menggunakan konsep *broadcast*. Ketika sudah mendapatkan nilai ID maka penyerang dapat masuk kedalam sistem sehingga penyerang bisa mendapatkan data yang dikirim namun masih dalam bentuk *chipertext*.

Perancangan pengujian serangan aktif dilakukan dengan menggunakan teknik *known plaintext attack(KPA)*. Tujuan dari penyerangan ini yaitu untuk mengetahui *key* dari sebuah *chipertext* dengan nilai *plaintext* tertentu. Pada penyerangan ini pihak *attacker* memiliki sejumlah informasi mengenai algoritme yang digunakan oleh sistem, ID dan nilai dari *chipertext* beserta nilai *plaintext* tertentu. Penyerangan dilakukan dengan mencoba untuk mencari nilai dari *key* menggunakan nilai *random* dengan panjang bit tertentu sesuai dengan *key* yang digunakan oleh sistem (80 bit). Sehingga penyerang akan melakukan percobaan berulang-ulang (*brute force*) untuk mendapatkan *key* dari sistem berdasarkan nilai *plaintext* dan *chipertext* yang diperoleh. Ketika nilai *plaintext* dan *chipertext* cocok maka hasil yang didapatkan adalah valid, dan penyerang dapat mengambil data asli yang dikirim oleh sistem dari *node* menuju *gateway*.

4. IMPLEMENTASI

Pada proses implementasi ini dilakukan berdasarkan pada bagian perancangan yang sudah disusun. Proses implementasi dibagi menjadi dua bagian besar, yaitu proses implementasi algoritme *PRESENT* dan implementasi sistem secara keseluruhan.

4.1 Implementasi Algoritme *PRESENT*

Pada Implementasi proses enkripsi algoritme *PRESENT* terdapat 3 proses tahapan utama yang harus dilalui. Pada setiap tahapan memiliki fungsi tersendiri dalam mengolah data

yang dienkripsi. Pada tahapan pertama yaitu dilakukan proses XOR dari *plaintext* yang diperoleh dari *node* dengan nilai *key* yang sudah ditentukan. Untuk *round* pertama maka nilai *subkey* awal sesuai dengan nilai *key* yang ditetapkan. Untuk *round* selanjutnya nilai *subkey* diperoleh dari hasil *round* sebelumnya yang kemudian dimasukkan ke dalam proses *key scheduling* .

Fungsi enkripsi pada tahapan awal yaitu *addRoundKey*, pada *addRoundKey* ini dilakukan proses XOR. Pada *state* awal, nilai dari masukan *key* dan *plaintext* sesuai dengan masukan berdasarkan panjang *block* yang ditentukan dengan total *round* sebanyak 31 kali. Pada setiap *round* dilakukan 3 buah tahapan, setelah tahapan XOR di bagian awal, selanjutnya hasil dari XOR dimasukkan ke dalam sebuah tabel *sBoxLayer* dengan ketentuan penataan *state* 4 bit untuk tiap *words*. Setelah hasil dari *sBoxLayer* diperoleh maka selanjutnya dilakukan tahapan *pLayer* berdasarkan nilai yang diperoleh dari proses *sBoxLayer*. Pada tahapan *pLayer* ini nilai bit dari *state* akan digeser sesuai dengan ketentuan dari tabel *pLayer*. Sehingga untuk setiap posisi akan berpindah dan menghasilkan nilai yang baru untuk selanjutnya digabungkan dan menghasilkan sebuah nilai *hexadecimal* baru.

Fungsi selanjutnya yaitu fungsi *key scheduling* yang terdiri dari tiga tahapan utama. Pada fungsi ini maka nilai *plaintext* dan *key* tetap sama berdasarkan nilai ketentuan dari algoritme *PRESENT*. Jumlah bit *key* yang digunakan adalah 80 bit, maka dilakukan *shifting* kekiri sebanyak 61 bit. Dari hasil *shifting* maka akan membentuk nilai baru yang selanjutnya dari *key* tersebut dimasukkan ke dalam tabel *sBoxLayer* pada nilai 4 bit terakhir. Setelah itu proses selanjutnya adalah XOR pada bit ke 15 sampai bit 19 dengan nilai *round_counter*. Sehingga dari beberapa proses tersebut menghasilkan sebuah nilai *key update* baru yang nantinya digunakan sebagai proses enkripsi pada *round* selanjutnya.

4.2 Implementasi Sistem

Pada bagian implementasi sistem yang akan dibahas adalah sistem yang sudah ditingkatkan keamanannya dengan menggunakan algoritme *PRESENT*. Pada bagian sistem *node* tahapan awal adalah dilakukan *set key* algoritme *PRESENT*. Selanjutnya pada fungsi sensor *nodes* terdapat sebuah *looping* dan kondisi untuk melakukan *sensing humidity* dan *temperature*. Selanjutnya nilai dari *humidity* dan *temperature* di *decode* ke dalam bentuk *hexadecimal* sebagai nilai *plaintext*. Kemudian dari nilai *plaintext* tersebut dilakukan proses enkripsi dengan melakukan pemanggilan fungsi *encrypt*. Setelah data sudah berhasil dienkripsi maka selanjutnya hasil dari data tersebut dimasukkan ke dalam sebuah *struct* supaya data bisa dikirim ke sisi *gateway* dengan terstruktur dalam bentuk *bytearray*.

Pada bagian sistem *gateway* tahapan awal adalah dilakukan *set key* algoritme *PRESENT*. Selanjutnya pada fungsi lora *gateway* terdapat sebuah *looping* dan kondisi untuk melakukan proses penerimaan data dari *node*. Data pengiriman yang diterima kemudian dilakukan *unpack* supaya selanjutya bisa dilakukan proses dekripsi. Setelah berhasil di *unpack* selanjutnya dilakukan *decode* nilai *chipertext* ke dalam bentuk *hexadecimal*. Setelah itu dilakukan pemanggilan fungsi *decrypt* dengan nilai *key* yang sudah ditentukan sebelumnya. Kemudian setelah berhasil dilakukan dekripsi maka dilakukan pencocokan nilai id dari *node* untuk kemudian hasil dekripsi dikirimkan ke *server* dalam bentuk *json*.

Pada sisi *server* tidak menggunakan modul komunikasi LoRa karena komunikasi yang digunakan antara *gateway* dan *server* menggunakan protokol MQTT dengan mekanisme *publish-subscribe*. Pada sistem *server* untuk menjalankan sistem terdapat 3 fungsi utama untuk melakukan *subscribe* topik. Untuk pemanggilan fungsi tersebut diawali dengan pembuatan objek baru mqtt untuk selanjutnya dilakukan pemanggilan fungsi untuk menjalankan mekanisme *publish-subscribe* topik. Kemudian hasil dari *subscribe* topik ditampilkan pada *monitor server*.

5. PENGUJIAN DAN PEMBAHASAN

5.1 Pengujian *Test Vector*

Pengujian *test vector* dilakukan dengan mencocokkan dan memvalidasi hasil keluaran dari algoritme *PRESENT* yang diimplementasikan pada sistem dengan *test vector* yang dimiliki oleh jurnal resmi dari pembuat algoritme. Pengujian *test vector* dilakukan dengan melakukan pengujian sebanyak 4 skenario uji. Dari keseluruhan hasil yang didapatkan, menghasilkan nilai valid.

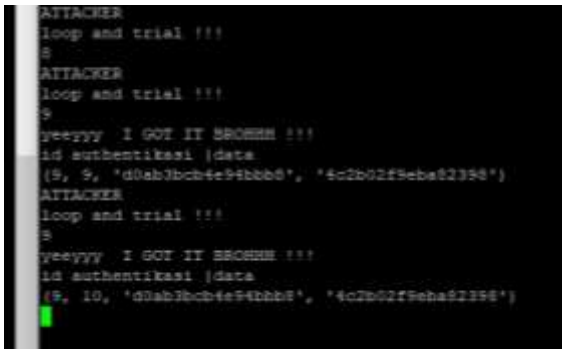
5.2 Pengujian Fungsional

Pengujian fungsional dilakukan dengan

melakukan cek validitas dari setiap fungsionalitas sistem yang sudah dibuat. Mulai fungsionalitas sensor dalam melakukan *sensing* data, pengiriman data dari *node* menuju *gateway* hingga *gateway* meneruskannya ke *server*, beserta sistem algoritme yang sudah diimplementasikan. Dari keseluruhan hasil yang didapatkan, menghasilkan nilai valid sehingga secara fungsional sistem berjalan dengan baik.

5.3 Pengujian Keamanan

Pengujian keamanan dibagi menjadi 2 berdasarkan serangan yang digunakan, yaitu pengujian serangan pasif dengan teknik *sniffing* dan pengujian serangan aktif dengan teknik *known plaintext attack (KPA)*. Pengujian serangan pasif dengan teknik *sniffing* dilakukan dengan menggunakan *gateway* tambahan *(attacker)* untuk melakukan *listen* terhadap *node*, kemudian penyerang melakukan percobaan pencocokan nilai id secara berulang-ulang untuk mendapatkan id yang cocok (menggunakan teknik *bruteforce*). Setelah mendapatkan ID yang sesuai maka informasi yang dikirim oleh *node* dapat diambil dan terbaca. Namun karena data yang dikirim sudah dalam bentuk enkripsi, maka penyerang tidak dapat mengetahui apa isi pesan asli data tersebut. Hal ini dapat dilihat pada Gambar 8.



Gambar 8 Hasil Serangan *Sniffing*

Kemudian pada serangan aktif, menggunakan teknik KPA, pada teknik ini *attacker* memiliki sejumlah informasi mengenai algoritme yang digunakan sistem, nilai id, dan nilai *plaintext* beserta nilai *ciphertext* tertentu. Dari sejumlah informasi ini akan dicari tahu mengenai *key* yang digunakan. hasil yang diperoleh dari serangan ini nihil setelah dilakukan percobaan berulang kali, sehingga penyerangan dengan KPA dinyatakan gagal karena penyerang tidak dapat menemukan *key* dari nilai *plaintext* dan *chipertext* yang ditentukan.

Gambar 9 Hasil Serangan KPA

6. PENUTUP

Pada penelitian ini telah berhasil mengimplementasikan algoritme *PRESENT* untuk proses enkripsi pada pengiriman data menggunakan modul komunikasi LoRa. Sistem pada awal mulanya memiliki kekurangan dari sisi keamanan, sehingga penyerang dapat memperoleh data dengan hanya melakukan serangan *sniffing*. Dengan dilakukannya implementasi algoritme *PRESENT* untuk proses enkripsi data, maka sistem pengiriman data menggunakan modul komunikasi LoRa dari *node* menuju *gateway* menjadi aman dari pihak yang tidak bertanggung jawab. Data yang dikirim berhasil dienkripsi dengan menggunakan algoritme *PRESENT*, sehingga bentuk data yang dikirimkan berupa *chipertext*. Maka apabila terdapat penyerang yang melakukan *sniffing*, data akan tetap aman, karena data yang didapat berupa *chipertext*, sehingga penyerang tidak akan mengetahui data asli yang dikirimkan. Kemudian pada serangan aktif menggunakan *Known Plaintext Attack (KPA)* yang bertujuan untuk mendapatkan nilai *key* dari *plaintext* dan *chipertext*, dinyatakan gagal karena penyerang tidak berhasil mendapatkan nilai *key*.

7. DAFTAR PUSTAKA

Arijuddin, . H., Bhawiyuga, A. & Amron, K., 2019. Pengembangan Sistem Perantara Pengiriman Data Menggunakan Modul Komunikasi LoRa dan Protokol MQTT Pada Wireless Sensor Network. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer* , Volume Vol. 3, pp. 1655-1659.

Pramukantoro , E. S., Bakhtiar, F. A., Aji, A. L. B., & Dewa, D. H. P. 2019. Implementasi Mekanisme End-To-End Security pada IoT Middleware. *Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK)* , Volume 6, pp. 335-340 .

Al-Fuqaha, A. et al., 2015. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE,* pp. 2347 - 2376.

Aras, E., Ramachandran, G. S., Lawrence, P. & Hughes, D. 2017. Exploring The Security Vulnerabilities of LoRa. *IEEE.*

Ashton, K. 2009. 'That "Internet of Things" Thing, in the Real World Things Matter More than Ideas'. *RFID Journal.*

Bogdanov, A. et al. 2007. PRESENT: An Ultra-Lightweight Block Cipher. *Springer,* pp. 450-466.

Lara-Nino, C. A., Diaz-Perez, A. & Morales-Sandoval, M. 2017. Lightweight Hardware Architectures for the Present Chiper in FPGA. *IEEE,* pp. 2544 - 2555.

Lavric, A. & Popa , V. 2017 . Internet of Things and LoRaTM Low-Power WideArea Networks: A Survey. *IEEE.*

Paar, C. & Pelzl, J. 2010. *Understanding Cryptograph.* Bochum: Springer.

Raza, U., Kulkarni, P. & Sooriyabandar, M. 2017. Low Power Wide Area Networks: An Overvie. *IEEE,* 19(2), pp. 855 - 873.

Wixted , A. J. et al. 2016. Evaluation of LoRa and LoRaWAN for Wireless Sensor Networks. *Orlando: IEEE SENSOR,* pp. 1-3.

Zou, Y., Zhu, J., Wang, X. & Hanzo, L. 2016. A Survey on Wireless Security: Technical Challenges, Recent Advances, and Future Trends. *Proceedings of the IEEE* , 104(9), pp. 1727 - 1765.