Impelentasi dan Analisis Perbandingan Performa Wireless Sensor Network Security Protocol TinySec dan SPINS

Aditya Bhagus Aria Hutomo[1)], Setyorini[2)], Sidik Prabowo[3)]

Prodi S1 Teknik Informatika, Telkom School of Computing, Telkom University
Jalan Telekomunikasi no. 1, Dayeuhkolot, Bandung 42057, Indonesia

[1]adit.aria@yahoo.co.id, [2]setyorini@telkomuniversity.ac.id, [3]pakwowo@telkomuniversity.ac.id

Abstrak

***Wireless Sensor Network* merupakan seperangkat alat yang digunakan untuk mengambil data dari lingkungan sekitar dengan cara *sensing*, lalu mengubahnya menjadi data digital untuk kemudian diteruskan ke *base station* melalui komunikasi nirkabel dan diproses. Dalam komunikasi nirkabel terdapat tujuh hal yang menjadi *security requirement*, antara lain *message confidentiality, message integrity, message authentication, freshness, availability, self-organization,* dan *secure localization.* Dalam penelitian ini akan dilakukan pengujian untuk membandingkan dua jenis protokol keamanan (*security protocol*), yaitu TinySec dan SPINS. Aspek yang menjadi tolak ukur performansi keduanya adalah autentikasi, *size overhead*, dan *energy consumption*. Masing-masing aspek tersebut memiliki *output packet* capture yang menunjukkan adanya paket yang diterima / di-drop, berupa jumlah byte yang ditambahkan pada *message*, dan grafik penggunaan energi beserta persentase *energy overhead*. Simulasi pengujian akan dilakukan pada *software* NS3 dengan memberikan penyerangan jenis *Packet Injection* kepada kedua protokol dengan tiga skenario berbeda. Hasil pengujian menunjukkan TinySec dan SPINS sama baiknya pada aspek *size overhead* dan autentikasi dengan hasil 11 byte overhead dan 100% kesuksesan autentikasi. Namun pada aspek *energy & power consumption* TinySec lebih unggul daripada SPINS dengan *energy overhead* masing-masing 2,8321% dan 8,0413%.**

Kata Kunci : *Security Protocol, Wireless Sensor Network, TinySec, SPINS*

1. PENDAHULUAN

Penggunaan sensor dilatarbelakangi oleh pengaplikasian di ranah militer. Sensor digunakan untuk mendeteksi lingkungan daerah kekuasaan musuh maupun daerah kekuasaan sendiri. Dalam hal ini, sensor perlu memenuhi standar agar dapat bertahan dari keadaan yang terbilang tidak ramah. Salah satu yang menjadi standar tersebut adalah pengaplikasian protokol keamanan pada sensor *node*.

Dalam pengaplikasiannya, protokol keamanan juga memiliki standar agar dapat memenuhi kewajibannya untuk memberikan keamanan pada komunikasi antar sensor *node*. Adapun yang menjadi standar bagi protokol keamanan disebut sebagai *security requirements*. Hal-hal yang menjadi security requirements adalah sebagai berikut[18].

- ***Message Confidentiality.*** *Message confidentiality* merupakan jaminan kerahasiaan informasi pada *message* dari pihak yang tak terotorisasi. Pada umumnya *confidentiality* didapatkan melalui proses enkripsi
- ***Message integrity.*** Merupakan jaminan bahwa *message* yang ditransmisikan terhindar dari perubahan secara ilegal saat transit. *Message integrity* didapatkan dari *Message Authentication Code* (MAC).
- ***Message Authentication.*** Node *receiver* harus melakukan otentikasi *message* yang diterima untuk memastikan bahwa *message* tersebut memang berasal dari pihak yang terotorisasi.
- ***Freshness.*** *Freshness* merupakan yang memberikan jaminan bahwa *message* ditransmisikan secara *real time* dan tepat waktu sebelum info dari *message* tersebut dirasa "basi".
- ***Availability.*** Tujuan utama dari *security protocol* adalah menyediakan efisiensi dari segi performa dan keamanan. Sebuah *security protocol* harus memenuhi spesifikasi *less processing & communication power*.
- ***Self-Organization.*** *Self-oraganization* dirasa sangat penting karena instalasi node-node WSN belum tentu di tempat yang selalu terjangkau oleh manusia.
- ***Secure Localization.*** Informasi lokasi dari sensor dirasa sangat penting untuk dimonitor ketika *point identification* perlu dilakukan.

Disamping pengaplikasian protokol keamanan, pihak pengembang juga perlu memperhatikan efisiensi untuk menekan penggunaan *resource* karena WSN sangat lekat dengan keterbatasan *resource*. Saat ini telah terdapat banyak jenis protokol keamanan yang tentunya memiliki keunggulan masing-masing. Protokol-protokol keamanan yang dikembangkan akhir-akhir ini memiliki tingkat sekuritas yang relatif tinggi jika dibandingkan dengan protokol keamanan terdahulu. Namun tingkat sekuritas yang tinggi berdampak pada konsumsi energi pada *node* sensor. Semakin kompleks algoritma *security* pada protokol tersebut, maka semakin tinggi pula konsumsi energi perangkat yang menyebabkan baterai perangkat lebih cepat habis.

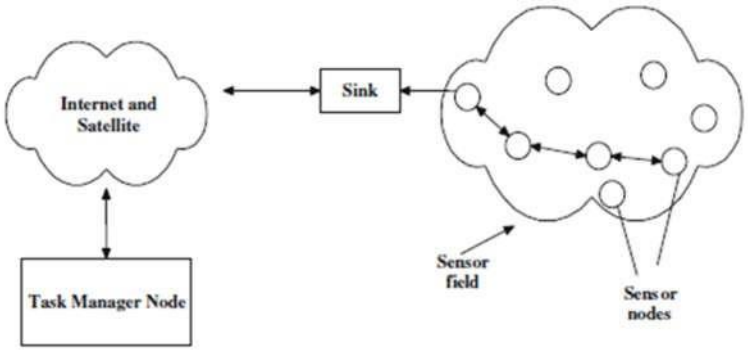
Dalam penelitian ini, dilakukan analisis terhadap perbandingan performa dua jenis *security protocol* yang sudah ada sejak tahun 2004, yaitu TinySec dan SPINS. TinySec menggunakan algoritma Skipjack untuk enkripsi, sedangkan SPINS menggunakan algoritma RC5. Namun kedua protokol tersebut menggunakan algoritma CBC-MAC untuk otentikasi. Keduanya juga sama-sama memiliki ketahanan terhadap *data spoofing* dan *message replay attack*[15].

Aspek yang diuji pada penelitian ini antara lain *size overhead*, autentikasi, dan *energy & power consumption*. *Size overhead* dipilih karena ukuran suatu paket yang dikirim akan mempengaruhi konsumsi energi dan daya. Aspek autentikasi dipilih karena autentikasi merupakan aspek yang paling mendasar dan dilakukan pada saat *initial handshake* dalam ranah keamanan jika dibandingkan dengan kedua aspek keamanan lainnya yang menjadi *security requirements* pada WSN (*confidentiality* dan *integrity*)[6]. Hal ini bukan berarti *cinfidentiality* dan *integrity* dapat dikesampingkan dan dianggap tidak lebih penting dari autentikasi.

2. TINJAUAN PUSTAKA

2.1 *Wireless Sensor Network* (WSN)

Javier Lopez (Computer Science Department University of Malaga Spanyol) meyatakan bahwa WSN merupakan sebuah sistem yang berbasis jaringan nirkabel, yang melakukan pemindaian data pada lingkungan sekitarnya di dunia nyata (*real world*) lalu mengubahnya menjadi data digital pada dunia komputer (*computer world*)[5].



Gambar 2.1 – *Wireless Sensor Network*[5]

Pada Gambar 2.1 menjelaskan konsep WSN, data lingkungan akan dikumpulkan oleh *sensor node* yang terdiri dari puluhan bahkan ratusan *sensor node* lalu data tersebut dikirimkan melalui sink node yang terkoneksi melalui internet atau satelite. Setelah itu data akan diterima oleh task manager yang berupa aplikasi yang memungkinkan untuk dilakukan monitoring terhadap kondisi lingkungan tersebut.

WSN sangat lekat dengan keterbatasan *resource*, dari mulai komputasi, memori, daya, hingga *bandwidth*. Hal tersebut menjadi tantangan dalam pengembangan WSN.

.

2.2 *Packet Injection Attack*

Packet injection merupakan jenis serangan dimana suatu pihak yang tak terautorisasi menyusup ke dalam suatu jaringan dan meng-*inject junk packet*[13]. Jika sudah berhasil memasuki jaringan, node penyerang dapat melakukan *inject packet* pun dapat mengubah jalur *routing* yang telah ada sebelumnya sehingga autentikasi pada rute yang seharusnya bisa dilewati. Node penyerang biasanya menempatkan diri di tengah-tengah node pengirim dan node penerima. Pada gambar 2, Na adalah node penyerang. Tanpa autentikasi yang

kuat, Node Na dapat meng-*inject* paket dengan *source* N1, sehingga seolah-olah Na meneruskan paket dari node N1 ke N4.

Gambar 2.2 – Skenario *Packet Injection Attack*[13]

2.3 TinySec

TinySec dibuat oleh Chris Karlof, Naveen Sastry, dan David Wagner dari University of California. Kemunculan TinySec dilatarbelakangi oleh kerentanan *security protocol* pada jaringan nirkabel seperti 802.11b dan GSM. Dengan sumber daya yang sedemikian terbatas sangat menjadi tantangan bagi para *engineer* untuk mendesain protokol keamanan yang mampu bekerja secara efisien. TinySec diklaim mampu bekerja hanya dengan menambahkan konsumsi energi kurang dari 10%[12].

Para perancang TinySec menargetkan protokol keamanan yang mampu memenuhi parameter yang dijabarkan sebagai berikut[12].

- ***Access control*.** Memastikan hanya node yang terautorisasi (memiliki *shared key*) sebelumnya yang mendapatkan akses.
- ***Integrity*.** Memastikan hanya data asli yang tak mengalami perubahan ilegal saat dalam perjalanan yang diterima. Serangan ini dikenal dengan *attack on information in transit*, dimana terdapat serangan di tengah perjalanan lalu pelaku yang disebut dengan *man-in-the-middle* mengubah data kemudian membroadcast ulang data palsu tersebut.
- ***Confidentiality*.** Memastikan pihak yang tak terautorisasi tidak dapat memodifikasi pesan.
- ***Ease of use*.** TinySec didesain untuk memberikan keamanan akses dan kemudahan untuk digunakan. Penyempurnaan dari protokol-protokol terdahulu yang terbilang sulit digunakan.

2.4 SPINS

Kemunculan SPINS dilatarbelakangi oleh permasalahan keamanan yang andal pada perangkat base station dan node sensor yang memiliki keterbatasan daya, bandwidth, storage, dan energi. SPINS dibangun oleh dua building block, yaitu SNEP dan µTesla. SNEP menjamin kerahasiaan data, data *freshness*, dan otentikasi antar 2 node. Sedangkan µTesla memberikan *authenticated routing*. SPINS didesain untuk mengoptimasi sensor network yang sangat terbatas dalam hal sumber daya.

2.4.1 *Sensor Network Encryption Protocol* (SNEP)

Sebagai protokol keamanan, SNEP mampu memberikan *data confidentiality, authentication, integrity,* dan *freshness*. *Data confidentiality* merupakan primitif keamanan yang paling mendasar, bahkan hampir semua protokol keamanan memiliki fitur tersebut. Bentuk paling sederhananya adalah enkripsi. Namun enkripsi biasa tidak cukup untuk menjamin kerahasiaan data. Jika enkripsi terlalu rumit, maka *cost* akan bertambah. Maka dari itu, Adrian Perrig beserta rekan-rekannya menyusun protokol keamanan SNEP sedemikian rupa. Adapun fitur utama SNEP adalah *semantic security, data authentication, replay protection,* dan *weak freshness*[7].

2.4.2 *µTESLA*

µTESLA merupakan bentuk kecil dari protokol TESLA yang didesain untuk menyediakan *authenticated broadcast* yang efisien. Namun TESLA tidak didesain untuk komputasi yang terbatas. TESLA memiliki overhead yang diperkirakan mencapai 24 byte per paket. Sedangkan sensor node hanya mampu mengirimkan message sepanjang 30 byte. Maka dari itu dibuat bentuk kecil dari TESLA, yaitu µTESLA dengan metode alternatif untuk menjaga level keamanan µTESLA agar tidak terdapat jarak yang terlalu jauh dengan TESLA.

Untuk mengatasi hal di atas, µTESLA didesain untuk menutupi kekurangan TESLA[7]:

- TESLA melakukan otentikasi paket awal dengan *digital signature* yang terlalu berat untuk node sensor, µTESLA hanya menggunakan mekanisme simetris.
- Membuka *key* pada setiap paket membutuhkan energi yang cukup besar untuk mengirim dan menerima paket. µTESLA membuka *key* sekali per *epoch*.

- Menyimpan *one-way key chain* dalam node sensor berdampak memberatkan node sensor tersebut. µTESLA membatasi jumlah pengirim yang terotentikasi.

2.5 Skipjack

Skipjack merupakan algoritma enkripsi yang memiliki 80-bit *key*, 64-bit *block cipher* dan bersifat simetris dimana *block cipher* tersebut dibagi menjadi 4 blok sehingga masing-masing blok terdiri atas 16-bit *words*. Skipjack dikenal dengan cipher yang sederhana, namun paling kuat di antara deretan konstruksi *cryptosystem* sekelasnya. Dalam penelitian ini algoritma Skipjack digunkan pada TinySec sebagai algoritma enkripsi.

Skipjack bekerja dengan dua tipe *rounds*, yaitu *A-rounds* dan *B-rounds*. Kedua tipe *rounds* dijalankan sebanyak total 32 kali, pertama-tama 8 *A-rounds* kemudian 8 *B-rounds*, dan sekali lagi, 8 *A-rounds* dan terakhir 8 *B-rounds*[11]. Dua tipe *rounds* tersebut dideskripsikan sebagai berikut.

***A-Rounds* (enkripsi)[11]**

$$\begin{aligned} w_1^{k+1} &= G^k(w_1^k) \oplus w_4^k \oplus \text{counter}^k \\ w_2^{k+1} &= G^k(w_1^k) \\ w_3^{k+1} &= w_2^k \\ w_4^{k+1} &= w_3^k \end{aligned} \tag{2.1}$$

***B-Rounds (enkripsi)*[11]**

$$\begin{aligned} w_1^{k+1} &= w_4^k \\ w_2^{k+1} &= G^k(w_1^k) \\ w_3^{k+1} &= (w_1^k) \oplus w_2^k \oplus \text{counter}^k \\ w_4^{k+1} &= w_3^k \end{aligned} \tag{2.2}$$

***A*$^{-1}$*-Rounds (dekripsi)*[11]**

$$\begin{aligned} w_1^{k-1} &= [G^{k-1}]^{-1}(w_2^k) \\ w_2^{k-1} &= w_3^k \\ w_3^{k-1} &= w_4^k \\ w_4^{k-1} &= w_1^k \oplus w_2^k \oplus \text{counter}^{k-1} \end{aligned} \tag{2.3}$$

***B*$^{-1}$*-Rounds (dekripsi)*[11]**

$$\begin{aligned} w_1^{k-1} &= [G^{k-1}]^{-1}(w_2^k) \\ w_2^{k-1} &= [G^{k-1}]^{-1}(w_2^k) \oplus w_3^k \oplus \text{counter}^{k-1} \\ w_3^{k-1} &= w_4^k \\ w_4^{k-1} &= w_1^k \end{aligned} \tag{2.4}$$

Pada notasi di atas, "counter" merupakan *counter* yang menghitung perulangan *rounds* dari 1 hingga 32. w_1,w_2,w_3,w_4 mewakili *words*. G_k merupakan kotak berisikan 16-bit input dan 4-byte *subkey k* dan juga merupakan Feistel chiper 4 *round* yang menggunakan byte permutasi *F* dan key-byte k_i. perlu diketahui bahwa isi dari *G* dan G^{-1} adalah sama, hanya saja pada G^{-1} menggunakan *subkey* dengan urutan terbalik.

2.6 RC5

Algoritma enkripsi RC5 memiliki mekanisme enkripsi dan dekripsi yang sangat sederhana dan memiliki *block cipher* simetris dan dapat bekerja dengan cepat. RC5 merupakan algoritma enkripsi yang berorientasikan *word* dimana operasi komputasi dasar harus merupakan operator yang bekerja sepenuhnya pada data dalam bentuk word. Pada penelitian ini, algoritma RC5 digunakan pada SPINS.

Algoritma RC5 dibagi menjadi tiga komponen: algoritma enkripsi, algoritma dekripsi, dan algoritma *key expansion*. Notasi yang digunakan pada operasi RC5 dijelaskan sebagai berikut[4].

1. Penjumlahan dua komplemen dinotasikan dengan tanda "+", namun operasi ini menunjukkan penjumlahan modulo-2^w. Sedangkan operasi pengurangan dinotasikan dengan tanda "-".
2. "$\oplus$" menunjukkan operasi XOR antara *word*.
3. "$x <<< y$" menunjukkan operasi rotasi *word* ke kiri (*left-spin*) dimana *y* adalah mod *w*, sehingga jika *w* merupakan bilangan kuadrat, maka hanya lg(*w*) *low-order* bit dari *y* yang menentukan jumlah rotasi.
4. "$x >>> y$" menunjukkan operasi *right-spin*

Enkripsi

Asumsi input *block* berupa dua *w*-bit register, yaitu *A* dan *B*. diasumsikan pula operasi *key expansion* sudah dilakukan, sehingga array *S*[0..*t* – 1] sudah diproses. Maka algoritma enkripsi yang dijalankan adalah[4]:

```
A = A + S[0]; B
= B + S[1]; for
i = 1 to r do
    A = ((A ⊕ B) <<< B) + S(2 * i];
    B = ((B ⊕ A) <<< A) + S[2 * i + 1];
```

Dekripsi

Algoritma dekripsi didapatkan dari melakukan operasi kebalikan dari algoritma enkripsi[4].

```
for i = r downto 1 do
    B = ((B – S[2 * i + 1]) >>> A) ⊕ A;
    A = ((A – S[2 * i]) >>> B) ⊕ B;
B = B – S[1];
A = A – S[0];
```

Operasi dekripsi ini merupakan kebalikan dari enkripsi dimana operasi penjumlahan menjadi pengurangan dan operasi *rotate left* menjadi *rotate right.*

Key Expansion

Algoritma ini bertujuan untuk mengembangkan *secret key K* untuk mengisi array *S* sehingga *S* membentuk array dari t = 2(r+1) *words* biner acak yang ditentukan oleh *K*. Algoritma *key expansion* tersusun atas tiga algoritma sederhana:

1. Konversi *secret key* dari format *byte* menjadi *words*
2. Inisialisasi array *S*
3. Penggabungan *secret key*

2.7 CBC-MAC

CBC-MAC memiliki beberapa versi pada detilnya seperti *padding*, *length variability*, *penguatan key search*. Umumnya, *padding* pada CBC-MAC dengan cara mempertimbangkan blok input akhir sebagai blok parsial dari data. Angka nol akan ditambahkan untuk memenuhi blok jika diperlukan.

Operasi penentuan banyak blok yang terdiri dari string biner *M* dengan panjang *l*, dinotasikan sebagai berikut[14].

$$M = M_1, M_2, ..., M_m \text{ dimana } |M_i| = l \quad (2.5)$$

Selanjutnya setiap blok melalui proses enkripsi *E* dengan *key K* dimana hasilnya akan dilakukan operasi XOR dengan blok selanjutnya. Proses tersebut dinotasikan dengan persamaan[14]:

$$(\ ) = \sum [\ \oplus\ ] \text{ untuk } i = 1 \text{ dan } C_0 = 0^l \quad (2.6)$$

2.8 Parameter Pengujian

Parameter yang akan dipakai yaitu *Size Overhead,* Autentikasi, dan *Energy Consumption.* Rincian parameter diatas adalah sebagai berikut :

1. Autentikasi
   Pengujian dengan parameter ini dilakukan dengan tujuan untuk mengukur kemampuan kedua protokol keamanan dalam menangani serangan yang diberikan. Pengujian dengan parameter autentikasi dilakukan dengan 3 skenario berbeda. Pada masing-masing skenario, disisipkan node penyerang dengan jumlah yang telah ditentukan.

2. *Size Overhead*
   Pada parameter ini akan diukur jumlah dari byte yang ditambahkan pada paket pada masing-masing protokol keamanan. Pengujian dengan parameter *size overhead* dilakukan pada topologi-1 yang akan dijelaskan pada bagian 3.

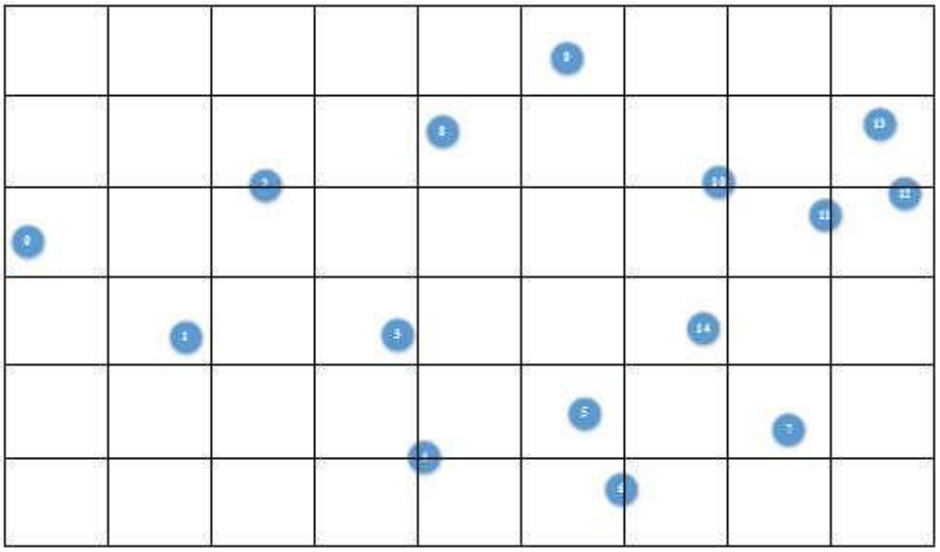
3. *Energy & Power Consumption*
   Skenario uji dengan parameter ini dilakukan pada topologi ke-1. Pada topologi ke-1 dilakukan pengukuran konsumsi energi topologi tanpa protokol keamanan, dengan protokol TinySec, dan dengan protokol SPINS. Hasil pengukuran energi akan diturunkan terhadap waktu sehingga menghasilkan konsumsi daya dan disandingkan dalam bentuk grafik disertakan dengan persentase *energy overhead.*

3. PERANCANGAN DAN IMPLEMENTASI

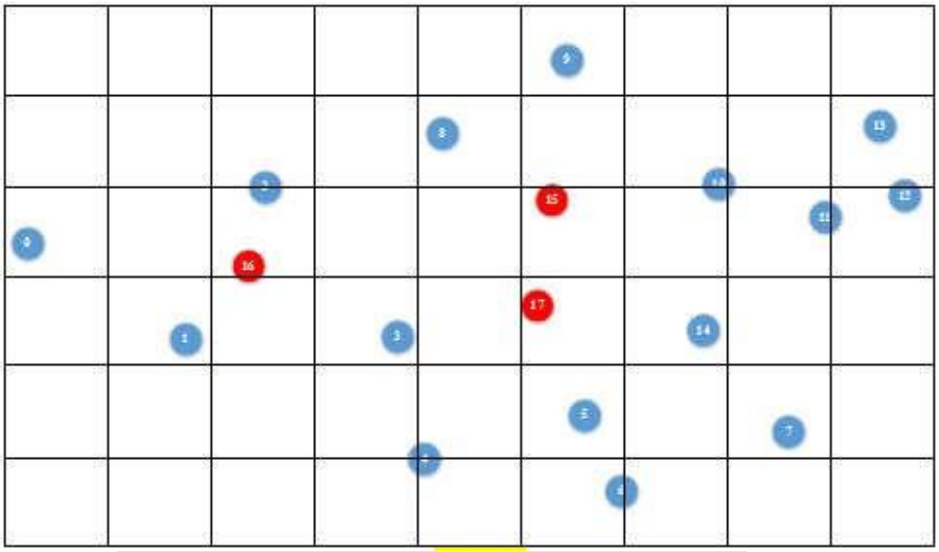
Sistem yang dibangun merupakan implementasi dalam bentuk simulasi. Simulasi dijalankan pada *software* NS3 di atas sistem operasi Linux Ubuntu 15.04. Protokol TinySec dan SPINS diimplementasikan ke dalam topologi yang berisikan node-node valid dan beberapa node penyerang. Jenis serangan yang dilakukan adalah "*Packet Injection*".

3.1 Topologi Jaringan

Pengujian dilakukan berdasarkan dua topologi untuk mengetahui perbandingan performansi dari TinySec dan SPINS pada aspek autentikasi, *size overhead*, dan *energy consumption*. Topologi-1 digunakan untuk skenario pengujian *size overhead* dan *energy consumption*. Topologi-2 digunakan untuk skenario pengujian dengan parameter autentikasi. Berikut merupakan topologi yang digunakan dalam pengujian.



Gambar 3.1 – Topologi - 1



Gambar 3.2 – Topologi - 2

Keterangan:
Node merah : node tak terautorisasi
Node biru : node terautorisasi

3.2 Skenario Simulasi

Skenario pengujian dilakukan berdasarkan spesifikasi yang dipaparkan pada tabel di bawah.

Skenario	1	2	3
Waktu Simulasi	1500 detik		
Panjang Transmisi Node	100m		
Kecepatan Bandwidth	1 Mbps		
Jumlah Node	15		
Parameter Uji	*Size overhead*	Autentikasi	*Energy consumption* & performa algoritma enkripsi
Topologi	1	2	1
Sub-Skenario-1	-	1 *attacker*	-
Sub-Skenario-2	-	2 *attacker*	-
Sub-Skenario-3	-	3 *attacker*	-
Jenis Serangan pada Jaringan	-	*Packet Injection*	-
Parameter Uji	Output		
Size overhead (byte)	Jumlah byte yang ditambahkan pada paket		
Autentikasi	Persentase paket valid yang diterima & paket non valid yang di-*drop*		
Energy consumption	Grafik penggunaan energi		
Performa algoritma enkripsi	Waktu yang dibutuhkan untuk menghasilkan ciphertext		

Tabel 3.1 – Skenario Pengujian

4. HASIL DAN ANALISIS

Pada bagian ini diuraikan hasil dan analisis dari pengujian berdasarkan skenario yang telah ditentukan sebelumnya. Penelitian ini menggunakan program simulasi dengan implementasi protokol keamanan jaringan TinySec dan SPINS menggunakan library Crypto++ yang dijalankan pada *software* NS3 yang berjalan di atas sistem operasi Linux Ubuntu. Pengujian dilakukan pada kedua protokol tersebut dengan parameter uji dan skenario yang telah ditentukan bertujuan untuk mengetahui performansi TinySec dan SPINS pada parameter yang telah ditentukan.

4.1 *Size Overhead*

Ukuran paket merupakan hal yang perlu diperhatikan dalam implementasi protokol pada jaringan sensor nirkabel mengingat karakteristik sensor yang memiliki sumber daya terbatas. Pada skenario pengujian dengan parameter ini yang didapatkan hasil bahwa topologi tidak berpengaruh terhadap *size overhead*. Dengan menggunakan masukan *plain text* yang sama, didapatkan hasil sebagai berikut.

	Plain text	***key***	***digest***	**IV**	**Ciphertext**	**Overhead**
TinySec	13 byte	10 byte	-	8 byte	16 byte	11 byte
SPINS	13 byte	16 byte	16 byte	8 byte	16 byte	11 byte

Tabel 4.1 – Hasil Pengujian Size Overhead

4.2 *Size Overhead*

Aspek autentikasi dibutuhkan untuk mencegah pihak yang tidak terautorisasi berpartisipasi dalam jaringan. Pengujian dalam aspek autentikasi dilakukan dengan cara verifikasi paket melalui perhitungan MAC yang mana akan dapat diketahui bahwa paket yang diterima itu berasal dari anggota jaringan atau pihak tak terautorisasi.

Protokol TinySec dan SPINS sama-sama menggunakan CBC-MAC dalam aspek autentikasi. Yang membedakan adalah SPINS menggunakan µTESLA untuk *authenticated routing*. Pada skenario ini pengujian dilakukan pada topologi-2 seperti pada gambar 3.2. Dari tiga skenario yang dilakukan, didapatkan hasil sebagai berikut.

Protokol	**Skenario**	**Total paket dikirim**	**Jumlah paket diterima**	**Jumlah paket di-*drop***	**% paket tak valid di drop**
TinySec	1	11121	10375	746	100%
	2	11477	10051	1426	100%
	3	12202	10016	2186	100%
SPINS	1	10766	10020	746	100%
	2	11444	10008	1436	100%
	3	11946	9813	2133	100%

Tabel 4.2 – Hasil Pengujian Autentikasi

Paparan hasil di atas menunjukkan bahwa TinySec dan SPINS mampu menangani serangan *packet injection* dengan baik dengan tingkat kesuksesan 100%.

4.3 *Energy Consumption* dan Waktu Mekanisme Enkripsi

Pada skenario pengujian dengan parameter *energy consumption* dan waktu mekanisme enkripsi dilakukan pada topologi-1 selama 1500 detik. Pada topologi ini dilakukan pengukuran konsumsi energi

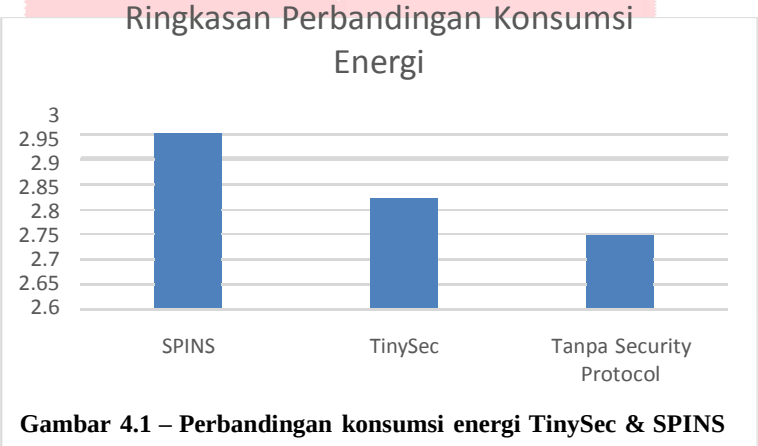
dengan skenario topologi tanpa protokol keamanan, topologi dengan TinySec, topologi dengan SPINS. Secara ringkas, berikut merupakan sajian data perbandingan konsumsi energi kedua protokol dengan topologi tanpa *security protocol*.

Waktu (s)	Konsumsi Energi (J)		
	SPINS	TinySec	Tanpa *Security Protocol*
1498.27	2.962069	2.819253	2.741609

Tabel 4.3 – Perbandingan konsumsi energi TinySec & SPINS

Waktu (s)	Konsumsi Daya (W)		
	SPINS	TinySec	Tanpa Security Protocol
1498.27	0.001976993	0.001881672	0.00182985

Tabel 4.4 – Perbandingan konsumsi daya TinySec & SPINS



Gambar 4.1 – Perbandingan konsumsi energi TinySec & SPINS

Dari sajian data di atas, dapat terlihat bahwa dari awal topologi yang disematkan protokol keamanan SPINS menggunakan lebih banyak dari TinySec. Hal ini dikarenakan SPINS memiliki dua *bulding block* (SNEP dan µTESLA) yang mengharuskannya melakukan lebih banyak aktivitas sehingga menambah *draw* yang secara otomatis akan menambah *energy cost*. Aktivitas yang dilakukan SPINS meliputi enkripsi dan juga aktivitas komputasi untuk menghasilkan *digital signature* (*digest*) bagi node-node pada jaringan. Digital signature tersebut juga ditistribusikan oleh node-node sehingga *state receive* dan *transmit* pada node-node dalam topologi yang disematkan protokol SPINS lebih banyak daripada *state idle* dan *sleep*.

Dari paparan data konsumsi energi pada tabel 4.3, maka didapatkan persentase *energy overhead* sebagai berikut.

Energy Overhead	
TinySec	SPINS
2,8321%	8,0413%

Tabel 4.5 – *Energy Overhead* TinySec & SPINS

Diperlukan suatu *tool* khusus yang dapat mengukur detil *power* yang digunakan pada setiap mekanisme yang dilakukan oleh masing-masing protokol. Untuk menggantikan hal tersebut, dipaparkan performansi algoritma enkripsi yang digunakan pada masing-masing protokol keamanan yang diuji dalam aspek waktu yang ditempuh untuk membentuk ciphertext. Berikut paparannya.

Percobaan ke -	Waktu Mekanisme Enkripsi (s)		
	TinySec (SKIPJACK)	SPINS (RC5+MD5)	SPINS (RC5)
1	0.076	0.084	0.04
2	0.064	0.088	0.072
3	0.056	0.108	0.056
4	0.068	0.08	0.052
5	0.064	0.076	0.064
Rata-rata	**0.0656**	**0.0872**	**0.0568**

Tabel 4.6 – Perbandingan performa algoritma enkripsi

Paparan hasil di atas merupakan waktu yang ditempuh untuk mengenkripsi *plain text* saja, tidak termasuk mekanisme dekripsi. Grafik di atas menunjukkan bahwa algoritma RC5 yang ada pada protokol SPINS terbukti lebih cepat daripada algoritma SKIPJACK pada TinySec dengan selisih waktu rata-rata 0,0088 detik. Namun jika SPINS diimplementasikan secara penuh (menyertakan SNEP dan µTESLA) algoritma RC5 yang dipadukan dengan *hash function* MD5 pada protokol SPINS menghasilkan rata-rata waktu lebih lambat dengan selisih waktu rata-rata dengan TinySec sebesar 0,0216 detik.

5. KESIMPULAN

Dari hasil analisis pengujian dapat ditarik kesimpulan bahwa TinySec mampu bekerja secara lebih efisien pada jaringan sensor nirkabel dengan autentikasi yang sama baiknya dengan SPINS (tingkat kesuksesan 100%) dan dengan konsumsi energi yang lebih sedikit (menghasilkan *energy overhead* sebesar 2,8321%). Hal ini dikarenakan SPINS memiliki dua *bulding block* (SNEP dan µTESLA) yang mengharuskannya melakukan lebih banyak aktivitas sehingga menambah arus yang secara otomatis akan menambah *energy cost*. Aktivitas yang dilakukan SPINS meliputi enkripsi dan juga aktivitas komputasi untuk menghasilkan *digital signature* (*digest*) bagi node-node pada jaringan. Digital signature tersebut juga didistribusikan oleh node-node sehingga *state receive* dan *transmit* pada node-node dalam topologi yang disematkan protokol SPINS lebih banyak daripada *state idle* dan *sleep*. Maka dari itu, secara otomatis, konsumsi energi TinySec lebih rendah daripada SPINS walaupun hanya terpaut 0,1428J.

6. DAFTAR PUSTAKA

[1] H. Wu, S. Nabar and R. Poovendran, "Energy Framework for the Network Simulator 3," *Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering*, 2011.

[2] R. Sharma, Y. Chaba and Y. Singh, "Analysis of Security Protocols in Wireless Sensor Network," *International Journal Advanced Networking and Applications*, 2010.

[3] M. Rouse and B. Pawliw, "Cipher Block Chaining," Juni 2007. [Online]. Available: http://searchsecurity.techtarget.com/definition/cipher-block-chaining.

[4] R. L. Rivest, "The RC5 Encryption Algorithm," 1997.

[5] I. P. A. E. Pratama and S. Suakanto, Wireless Sensor Network, Bandung: Informatika, 2015.

[6] T. Pornin, "Difference between authentication, integrity and data origin authentication," 7 Juli 2015. [Online].

Available: http://security.stackexchange.com/questions/93322/difference-between-authentication-integrity-and-data-origin-authentication.

[7] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen and D. E. Culler, "SPINS: Security Protocols for Sensor Networks," *Wireless Networks,* 2002.

[8] A. Muhammad, "Implementation and Verification of SKIPJACK Algorithm using Verilog," *IEEE,* 2002.

[9] B. Mbarek and A. Meddeb, "Energy Efficient Security Protocols for Wireless Sensor Networks: SPINS vs TinySec," *Institute of Electrical and Electronics Engineers,* 2016.

[10] T. G. Lupu, "Main Types of Attacks in Wireless Sensor Networks," *Recent Advances in Signals and Systems,* 2009.

[11] L. Knudsen and D. Wagner, "On the Structure of Skipjack," *Discrete Applied MAthematics,* pp. 103-116, 2001.

[12] C. Karlof, N. Sastry and D. Wagner, "TinySec: A Link Layer Security Architecture for Wireless Sensor Networks," 2004.

[13] Q. Gu, P. Liu, C.-H. Chu and S. Zhu, "Defense Against Packet Injection in Ad Hoc Networks," *Global Telecommunication Conference,* 2005.

[14] J. Deepakumara, H. M. Heys and R. Venkatesan, "Comparison of Message Auhtentication Code (MAC) Algorithm for the Internet Protocol Security (IPSEC)," *Newfoundland Electrical and Computer Engineering Conference,* 2003.

[15] T. Chen and D. Yan, "Research and Implementation on TinySec Rekeying for Wireless Sensor Networks," *International Conference on Multimedia and Information Technology,* 2008.

[16] S. M. Bellovin, "Modes of Operation," *Computer Science of Columbia University,* 2009.

[17] M. Bellare, J. Kilian and P. Rogaway, "The Security of the Cipher Block Chaining Message Authentication Code," *Journal of Computer and System Sciences,* 2001.

[18] A. S. Ahmed, "An Evalution of Security Protocols on Wireless Sensor Network," *Seminar on Internetworking,* 2009.