DISERTASI – EE 186601

AUTENTIKASI JARINGAN SENSOR NIRKABEL UNTUK MENGHADAPI SERANGAN DoS BERBASIS PKC

FARAH AFIANTI
NRP. 07111660010014

DOSEN PEMBIMBING
Dr. Ir. Titiek Suryani, M.T.
Dr. Ir. Wirawan, DEA

PROGRAM DOKTOR
DEPARTEMEN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI ELEKTRO
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2019

DISERTASI – EE 186601

AUTENTIKASI JARINGAN SENSOR NIRKABEL UNTUK MENGHADAPI SERANGAN DoS BERBASIS PKC

FARAH AFIANTI
NRP. 07111660010014

DOSEN PEMBIMBING
Dr. Ir. Titiek Suryani, M.T.
Dr. Ir. Wirawan, DEA

PROGRAM DOKTOR
DEPARTEMEN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI ELEKTRO
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2019

LEMBAR PENGESAHAN DISERTASI

Disertasi disusun untuk memenuhi salah satu syarat memperoleh gelar
Doktor (Dr)
di
Institut Teknologi Sepuluh Nopember
Oleh:

FARAH AFIANTI
NRP: 07111660010014

Tanggal Ujian: 11 Juli 2019
Periode Wisuda: September 2019

Disetujui oleh:
Pembimbing:

1. Dr. Ir. Titiek Suryani, M.T.
   NIP: 196411301989032001 .....

2. Dr. Ir. Wirawan, DEA
   NIP: 196311091989031011 .....

Penguji:

1. Ir. Hendrawan, M.Sc., Ph.D.
   NIP: 196007051987021002 .....

2. Prof. Dr. Ir. Joko Lianto Buliali, M.Sc.
   NIP: 196707271992031002 .....

3. Dr. Surya Sumpeno, S.T., M.Sc.
   NIP: 196906131997021003 .....

Kepala Departemen Teknik Elektro
Fakultas Teknologi Elektro

Dr. Eng. Ardyono Priyadi, S.T., M.Eng
NIP: 197309271998031004

PERNYATAAN KEASLIAN DISERTASI

Dengan ini Saya menyatakan bahwa isi sebagian maupun keseluruhan disertasi saya dengan judul :

AUTENTIKASI JARINGAN SENSOR NIRKABEL UNTUK MENGHADAPI SERANGAN DoS BERBASIS PKC

adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diizinkan dan bukan merupakan karya pihak lain yang Saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka.

Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya, 31 Juli 2019

Farah Afianti

NRP. 07111660010014

AUTENTIKASI JARINGAN SENSOR NIRKABEL UNTUK MENGHADAPI SERANGAN DoS BERBASIS PKC

Nama mahasiswa	:	Farah Afianti
NRP	:	07111660010014
Pembimbing I	:	Dr. Ir. Titiek Suryani, M.T.
Pembimbing II	:	Dr. Ir. Wirawan, DEA

ABSTRAK

Komunikasi broadcast pada jaringan sensor nirkabel (JSN) sangat efisien dan berdampak besar pada seluruh node sensor. Alasan tersebut mendasari pemanfaatan node sensor terutama pada proses diseminasi data, kode, pemeliharaan, menjalankan perintah atau query maupun sinkronisasi. Kondisi ini diikuti dengan rawannya keamanan yang dapat mengganggu ketersediaan komunikasi pada JSN. Oleh karena itu dibutuhkan proses autentikasi pengguna untuk memastikan apakah pengguna valid. Pemanfaatan digital signature untuk autentikasi memiliki keamanan yang tinggi akan tetapi harus diimbangi dengan komputasi yang tinggi. Kekurangan ini dimanfaatkan peretas untuk mengirimkan signature palsu dalam jumlah besar sehingga node sensor akan sibuk memverifikasi dan proses ini dikenal dengan serangan *Denial of Service* (DoS) berbasis *Public Key Cryptography* (PKC). Berbagai metode filter dikembangkan untuk mengatasi masalah ini. Mekanisme ini memiliki komputasi rendah dan bersifat mendampingi proses verifikasi signature bukan menggantinya. Skema puzzle merupakan salah satu filter dengan komputasi rendah namun dapat diandalkan dalam mengatasi serangan DoS berbasis PKC akan tetapi memiliki kelemahan pada delay yang tinggi dalam mencari solusi puzzle pada sisi pengirim.

Berkembangnya berbagai aplikasi pendukung yang akan mengakses node sensor secara langsung meningkatkan keragaman dan skalabilitas pada JSN. Sedangkan node sensor memiliki keterbatasan dalam sumber daya terutama pada ruang penyimpanan dan kemampuan komputasi. Oleh karena itu, penelitian ini mengajukan skema puzzle dinamis pada JSN menggunakan Multiuser-Dynamic Cipher Puzzle (M-DCP) yang dilengkapi dengan TinySet. Skema ini bertujuan untuk mengurangi waktu pemrosesan dan dapat dimanfaatkan oleh banyak pengguna atau pengirim dengan kebutuhan ruang penyimpanan pada JSN yang rendah. M-DCP memanfaatkan fungsi ambang untuk membatasi jumlah iterasi hash. Hasil percobaan menunjukkan bahwa fungsi ambang eksponensial dapat menurunkan delay pada sisi pengirim hingga 94% dengan peluang ditemukannya solusi hingga $1-(1.738x10^{-13})$. Sedangkan pemanfaatan TinySet yang telah diregularisasi bisa menghemat ruang penyimpanan hingga 77% dibandingkan dengan *Counting Bloom Filter* (CBF). Pemanfaatan skema ini berdampak pada meningkatnya komputasi pada proses verifikasi. Peningkatan ini bernilai hingga 36% dibandingkan dengan *Bloom Filter based Authentication* (BAS) atau pada

implementasinya membutuhkan waktu tidak lebih dari 0.5 detik. Performansi yang didapatkan diimbangi dengan meningkatnya keamanan terutama pada autentikasi, confidentiality dan ketahanan terhadap serangan DoS berbasis PKC. Hal ini dibuktikan dengan peningkatan kompleksitas serangan menggunakan brute force hingga $1.86x10^{137}$ trial yang didapatkan dari proses autentikasi menggunakan *Elliptic Curve Digital Signature Algorithm* (ECDSA), proses enkripsi menggunakan *Rivest Cipher* 5 (RC5) dan proses pembuatan puzzle menggunakan DCP.

Kata kunci: autentikasi, DoS, jaringan sensor nirkabel, multiuser, skema puzzle

AUTHENTICATION FOR WIRELESS SENSOR NETWORKS AGAINST PKC-BASED DENIAL OF SERVICE (DoS)

Student Name	:	Farah Afianti
NRP	:	07111660010014
Promotor	:	Dr. Ir. Titiek Suryani, M.T.
Co-Promotor	:	Dr. Ir. Wirawan, DEA

ABSTRACT

Broadcast communications on wireless sensor networks (WSN) are very efficient and have a large impact on all sensor nodes. These reasons underlie the use of sensor nodes, especially in the process of disseminating data, code, maintenance, running commands or queries and synchronization. This condition is followed by the vulnerability of security which can interfere with the availability of communication in the WSN. Therefore, a user authentication process is needed to determine whether the user is valid. The use of digital signatures for authentication has high security but must be balanced with high computation. This weakness is used by hackers to send fake signatures in large numbers so that the sensor nodes will be busy verifying and this process is referred as Public Key Cryptography (PKC) based Denial of Service (DoS) attack. Various filter methods were developed to overcome this problem. This mechanism has low computation and accompanies the main digital signature in the verification process instead of replacing it. The puzzle scheme is one of the low computational filters and can be relied upon in dealing with PKC-based DoS attacks but has a weakness in high delay in finding a puzzle solution on the sender side.

The development of various supporting applications that will access sensor nodes directly increases the diversity and scalability of WSN. However, the sensor nodes have resources limitations, especially in storage space and low computing capabilities. Therefore, this study proposes a dynamic puzzle scheme on WSN using Multiuser-Dynamic Cipher Puzzle (M-DCP) equipped with TinySet. This scheme aims to reduce processing time and can be accessed by many users or senders with low storage space requirements. M-DCP uses threshold functions to limit the number of hash iterations. The experimental results show that the exponential threshold function can reduce the delay on the sender side by up to 94% with the chance of finding solutions up to 1- ($1{,}738 \times 10^{-13}$). Moreover, the utilization of TinySet that has been regularized can save storage space up to 77% compared to Counting Bloom Filter (CBF). The utilization of this scheme has an impact on increasing computing in the verification process. This increase is worth up to 36% compared to Bloom Filter based Authentication (BAS) or the implementation takes no more than 0.5 seconds. The performance obtained is balanced with the increasing security, especially in authentication, confidentiality and resilience to PKC-based DoS attacks. This is evidenced by the increasing attack complexity using brute force to 1.86×10^{137} trials obtained from the

authentication process using the Elliptic Curve Digital Signature Algorithm (ECDSA), the encryption process using Rivest Cipher 5 (RC5) and the puzzle generation process using DCP.

Keywords: authentication, DoS, wireless sensor networks, multiuser, puzzle scheme

KATA PENGANTAR

Segala puji bagi Allah, Rabb alam semesta karena atas kuasa dan rahmatNya sehingga disertasi ini dapat diselesaikan. Disertasi ini disusun untuk memenuhi salah satu syarat akademik pada program doktor di Departemen Teknik Elektro, Fakultas Teknologi Elektro, Institut Teknologi Sepuluh Nopember (ITS), Surabaya.

Pada kesempatan ini, penulis menyampaikan terima kasih kepada berbagai pihak, antara lain :

1. Pemerintah Repulik Indonesia melalui program Beasiswa Pendidikan Indonesia (BPI) dari Lembaga Pengelola Dana Pendidikan (LPDP) yang telah memberikan dukungan baik dana maupun hal-hal terkait lainnya selama masa studi;
2. Para pembimbing yaitu Ibu Dr. Ir. Titiek Suryani, M.T. selaku promotor dan Bapak Dr. Ir. Wirawan, DEA selaku co-promotor atas kesabaran, arahan dan motivasi yang telah diberikan;
3. Para penguji yaitu Bapak Ir. Hendrawan, M.Sc., Ph.D, Bapak Prof. Dr. Ir. Joko Lianto Buliali, M.Sc dan Bapak Dr. Surya Sumpeno, S.T., M.Sc. atas saran dan masukannya;
4. Rekan-rekan mahasiswa di Laboratorium B303 dan B304 khususnya Ibu Ari Endang Jayati, Ibu Yuning Widiarti, Ibu Mike Yuliana, Ibu Lusia Rakhmawati dan Bapak Wahyu Pamungkas atas bantuan dan dukungannya;
5. Segenap pengelola, karyawan dan dosen dari Departemen Teknik Elektro, Fakultas Teknologi Elektro;
6. Segenap rekan-rekan mahasiswa S3 atas dukungannya;
7. Suami dan anak-anakku tersayang yang selalu berdoa dan memberikan dukungan juga semangat;
8. Ibuku Darti dan papaku M. Fadli (almarhum) beserta keluargaku tercinta atas doa, dukungan dan restunya;
9. Kerabat dan handai taulan;

Semoga Allah SWT membalas amal serta kebaikan Bak dan Ibu semua serta selalu melimpahkan karuniaNya kepada kita semua.

Penulis menyadari bahwa penulisan disertasi ini masih belum sempurna sehingga kritik dan saran yang membangun dari semua pihak sangat diharapkan. Hal ini dimaksudkan untuk perbaikan dan penyempurnaan dari buku disertasi termasuk kelanjutan dari penelitiannya.

Akhir kata, semoga buku disertasi ini bermanfaat bagi berbagai pihak khususnya demi kemajuan ilmu dan teknologi.

Surabaya, 29 Juli 2019

Penulis

DAFTAR ISI

Halaman ini sengaja dikosongkan

DAFTAR GAMBAR

Halaman ini sengaja dikosongkan

DAFTAR TABEL

Halaman ini sengaja dikosongkan

DAFTAR SIMBOL

M	:	Pesan
UID	:	Identitas pengguna
PK	:	Kunci publik
SK	:	Kunci privat
$H(.)$	:	Operasi hash
α	:	Total PKC valid yang dikirim oleh node sensor tetangga
β	:	Total PKC invalid yang dikirim oleh peretas
C	:	Kemampuan node sensor untuk memproses PKC
$W(i)$	:	Time window ke *i*, dengan *i* adalah bilangan integer lebih besar dari 0
$N(i)$	:	Jumlah signature yang diterima node sensor pada saat time window $W(i)$
T_w	:	Interval waktu pada time window W
W_{group}	:	Nilai komitmen group
q	:	Jumlah bit yang dimanfaatkan untuk perhitungan pada metode grouping
m	:	Jumlah kelompok pada metode grouping
$Hp_{[x]}$	:	Himpunan pada kondisi x
$P(X)$	:	Peluang kejadian *X*
$PS(X)$	:	Peluang kesuksesan
l	:	Kekuatan puzzle
l_{maks}	:	Maksimum kekuatan puzzle
K_{idx}	:	Kunci sesi pada indeks ke *idx*
$Pidx$	:	Puzzle pada indeks ke *idx*
Ns	:	Bilangan random yang dibangkitkan oleh server
Nc	:	Bilangan random yang dibangkitkan oleh client
Idx	:	Nilai indeks
Sig	:	Signature
ETM	:	Proses enkripsi kemudian bangkitkan MAC
tt	:	timestamp
n_{key}	:	Panjang kunci sesi
$ExpT$	:	Waktu kadaluwarsa dari sertifikat yang dibuat
q_{BFV}	:	Banyaknya operasi hash pada pembentukan BFV
r_{BFV}	:	Panjang vektor pada bloom filter (bit)
$PubK$	:	Kunci publik pada skema puzzle dinamis
B	:	Jumlah bucket pada TinySet
B_{akt}	:	Nilai aktual pada jumlah bucket pada TinySet
L	:	Ukuran indeks pada TinySet
R	:	Banyaknya fingerprint pada TinySet
$Cell$	:	Ukuran fingerprint
FPR	:	Peluang false positif
λ	:	Rata-rata jumlah fingerprint pada setiap chain

λ_{akt}	:	Nilai aktual dari rata-rata jumlah fingerprint pada setiap chain
N	:	Banyaknya jumlah anggota
TS_{size}	:	Ukuran TinySet
B_{cap}	:	Kapasitas bucket
E	:	Energi
V	:	Tegangan listrik (volt)
I	:	Arus listrik (ampere)
t	:	Waktu (detik)
n_{DCP}	:	Banyaknya iterasi hash pada skema DCP
P_{ZS}	:	Peluang tidak ditemukannya solusi puzzle
σ_l	:	Standar deviasi pada kekuatan puzzle l
Q	:	Kunci publik pada ECDSA
d	:	Kunci privat pada ECDSA
G	:	Base point pada kurva elips di Fp
SN_i	:	Node sensor ke-i

DAFTAR SINGKATAN

ACO	:	*Actual Chain Offset*
BAS	:	*Bloom Filter-based Authentication*
BFV	:	*Bloom Filter Vector*
CAS	:	*Certificate based Authentication Scheme*
CBF	:	*Counting Bloom Filter*
DAS	:	*Direct-storage based Authentication Scheme*
DCP	:	*Dynamic Cipher Puzzle*
DMSP	:	*Dynamic Message Specific Puzzle*
DoS	:	*Denial of Service*
ECC	:	*Elliptic Curve Cryptography*
ECDH-ECDSA	:	*Elliptic Curve-Diffie Hellman Elliptic Curve Digital Signature Algorithm*
ECDSA	:	*Elliptic Curve Digital Signature Algorithm*
HAS	:	*Hybrid based Authentication Scheme*
IoT	:	*Internet of Things*
JSN	:	Jaringan Sensor Nirkabel
LCO	:	*Logical Chain Offset*
MAC	:	*Message Authenticate Code*
MAD	:	*Mean Absolute Deviation*
MSP	:	*Message Specific Puzzle*
M-DCP	:	*Multiuser - Dynamic Cipher Puzzle*
NS3	:	*Network Simulator 3*
OSI	:	*Open Systems Interconnection*
PKC	:	*Public Key Cryptography*
RC5	:	*Rivest Cipher 5*
RFID	:	*Radio-Frequency IDentification*
RMSE	:	*Root Mean Square Error*
ROM	:	*Random Oracle Model*
SG	:	*Smart Grid*
SHA1	:	*Secure Hash Algorithm 1*

Halaman ini sengaja dikosongkan

DAFTAR ISTILAH

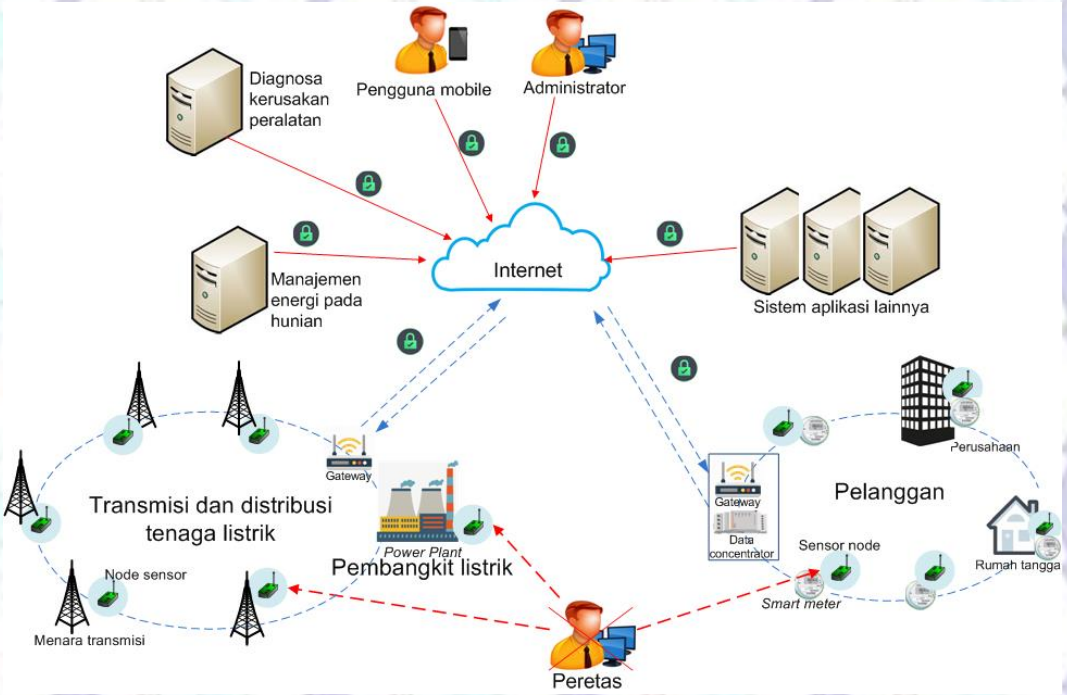
Autentikasi	layanan keamanan untuk memastikan identitas dari pengakses jaringan sehingga komunikasinya otentik
Bloom filter	Struktur data probabilistik yang sederhana dan efisien dalam merepresentasikan keanggotaan pada sebuah kelompok
Confidentiality	layanan keamanan untuk menjamin kerahasiaan data dari pihak ketiga yang tidak berhak untuk melihat konten menggunakan komunikasi apapun
Digital signature	Tambahan informasi pada sebuah unit data yang memungkinkan penerima untuk membuktikan sumber asal dan integritas dari unit data tersebut
Enkripsi	Transformasi kriptografi dari data asli menjadi kode ciphertext menggunakan algoritma tertentu
Fairness in answering	Bagian dari puzzle difficulty yang diartikan jika peretas gagal memecahkan puzzle maka proses kegagalan itu dibatasi oleh *t* iterasi pada kekuatan puzzle yang konstan,
Fairness in solving	Bagian dari puzzle difficulty yang diartikan jika peretas tidak dapat menghasilkan solusi puzzle dalam iterasi yang lebih rendah dari t_{max} dan dibatasi oleh kekuatan puzzle yang konstan
Fungsi *advantage*	sebuah pengukuran untuk mengetahui keberhasilan peretas dalam menyerang algoritma kriptografi dengan cara lain yang berbeda dengan versi ideal algoritma pembangkitannya
Fungsi hash kriptografi	kelas khusus dari fungsi hash yang memenuhi properti tertentu sehingga bisa dimanfaatkan pada kasus-kasus di bidang kriptografi
Integrity	layanan keamanan untuk menghindari modifikasi yang tidak resmi baik menambah, mengurangi maupun mengirim ulang tanpa terdeteksi oleh pengguna yang berhak
Jaringan Sensor Nirkabel	Kumpulan node sensor yang secara kooperatif mengawasi lingkungan fisik yang luas
Kekuatan puzzle	Jumlah bit yang sama antara hasil luaran hash dengan pola tertentu
Kompleksitas serangan	Sebuah metrik untuk menjelaskan kondisi di bawah kendali peretas dan harus ada untuk mengetahui kelemahan sebuah sistem
Peluang tidak ditemukannya solusi	Peluang pembangkitan puzzle yang tidak menemukan solusi hingga batas iterasi hash yang ditentukan

Penerima	Pihak yang menerima pesan atau paket, dalam penelitian ini yaitu node sensor
Pengirim	Pihak yang mengakses node sensor
Puzzle difficulty	Parameter keamanan puzzle yang mensyaratkan bahwa tidak ada peretas yang dapat memecahkan satu puzzle atau teka-teki lebih cepat dari beberapa batasan yang ditentukan
Puzzle unforgeability	Parameter keamanan puzzle yang mensyaratkan bahwa tidak ada peretas yang dapat menghasilkan puzzle yang tampak valid
Skalabilitas	Kemampuan untuk menangani jumlah pengirim atau pengguna node sensor
Tag	Informasi tambahan untuk mengirim parameter kekuatan puzzle dan indeks secara eksplisit

BAB 1
PENDAHULUAN

1.1. Latar Belakang

Jaringan sensor merupakan sekumpulan node sensor yang bertugas untuk menghubungkan kondisi lingkungan fisik dengan dunia digital (Dargie & Poellabauer 2010). Komunikasi ini bertujuan untuk menangkap fenomena alam, memproses, menyimpan dan bahkan menjalankan aksi tertentu. Sedangkan, node sensor sendiri merupakan perangkat keras berdaya rendah yang bertugas mengecap kondisi fisik tertentu dan dilengkapi dengan antena untuk menghubungkan dengan node yang lain secara nirkabel sehingga kumpulan dari node sensor pada ruang lingkup tertentu membentuk Jaringan Sensor Nirkabel (JSN). Beberapa aplikasi yang memanfaatkan JSN antara lain di bidang pertanian (penjadwalan irigasi dengan memantau kelembaban tanah dan kondisi cuaca ; informasi nutrisi tanah dalam rangka memprediksi kesehatan tanaman dan kualitas produksi) (Ojha et al. 2015), kesehatan (monitoring dan deteksi berbagai penyakit maupun status kesehatan manusia) (Alemdar & Ersoy 2010), keamanan militer (pelacakan target dan pengawasan medan perang) (Dong et al. 2013) ; bahkan untuk Vehicular ad hoc networks (VANETs) dalam rangka menghindari terjadinya kecelakaan dan kemacetan di jalanan (Grover & Lim 2015). Selain itu, JSN juga dimanfaatkan untuk aplikasi real-time pada sistem kelistrikan terkini, *Smart Grid* (SG), seperti pengawasan kualitas daya, pembangkitan terdistribusi di sisi pembangkitan daya ; deteksi kerusakan alat, deteksi kerusakan mesin pada sisi transmisi dan distribusi daya ; *wireless automatic meter reading* (WAMR), pengawasan peralatan grid daya pada sisi konsumen (Fadel et al. 2015). Ilustrasi pemanfaatan JSN pada bidang kelistrikan dapat dilihat pada Gambar 1.1. JSN tidak hanya diakses oleh administrator tetapi data terkini juga dapat diakses langsung oleh aplikasi-aplikasi pendukung SG maupun pelanggan. Aplikasi pengakses pun memungkinkan untuk memiliki keragaman baik platform



Gambar 1.1 JSN pada Smart Grid

perangkat keras juga kebutuhan akan data. Keragaman ini akan meningkat seiring dengan semakin berkembangnya berbagai teknologi dan kebutuhan akan informasi pada Internet of Things (IoT).

Pemanfaatan JSN pada sebuah sistem harus memperhatikan aspek keamanan dan performansi. Beragam informasi penting yang ditransmisikan memungkinkan peretas untuk mencuri dengar atau bahkan menyusupi pesan sehingga berakibat pada penanganan atau diagnosa yang salah terhadap sebuah objek (Li et al. 2015). Hal ini didukung dengan ruang lingkup JSN yang umumnya digunakan dalam jumlah yang besar untuk mengawasi lingkungan geografis yang luas. Berdasarkan karakteristik dari node sensor, topologi mesh merupakan mekanisme yang sesuai digunakan untuk menghubungkan antara satu node dengan node yang lain. Kondisi tersebut mengakibatkan sebuah node sensor harus melewati tingkatan hop agar dapat berkomunikasi dengan node tetangga lainnya. Mengingat node sensor memiliki keterbatasan dalam sumber daya, maka dibutuhkan manajemen dan pengaturan berbagai aspek untuk memperpanjang usianya (Yang 2013). Selain itu, JSN umumnya ditempatkan di tempat terpencil, jauh dari jangkauan manusia sehingga administrator hanya perlu mengakses dan

mengatur JSN dari jarak jauh. Berbagai alasan tersebut menjadi tantangan utama dalam pembangunan JSN yang aman dan efisien. Salah satu cara dalam menghemat komunikasi antara administrator ataupun pengguna dengan JSN adalah melalui broadcast. Komunikasi tersebut sangat efisien dalam mengirim pesan yang bernilai sama dan ditujukun untuk semua node sensor. Pemanfaatan broadcast ini tidak bisa digunakan secara langsung. Selain teknik routing untuk menghindari redudansi diterimanya pesan pada sebuah node sensor, hal lain yang perlu diperhatikan adalah serangan yang memanfaatkan broadcast. Hal tersebut penting, karena dampaknya cukup signifikan terhadap keberlangsungan JSN secara keseluruhan. Serangan yang memanfaatkan paket palsu dalam jumlah besar dan dikirim secara terus menerus sehingga mengakibatkan JSN tidak bisa diakses karena sibuk melayani verifikasi dari paket palsu disebut *Denial of Service* (DoS). Jika dibiarkan serangan ini dapat mengurangi usia node sensor atau bahkan mengakibatkan kerugian bagi perusahaan. Beberapa contoh dari dampak serangan DoS pada beberapa kasus nyata yaitu:

a. Pada tahun 2013, sistem pembayaran online dengan visa, mastercard dan paypal telah diserang menggunakan serangan DoS yang terdistribusi atau disebut dengan DDoS. Kerugian yang dicapai oleh ketiga perusahaan tersebut mencapai £3.5 juta. Tujuan dari peretas sebenarnya bukan untuk komersial melainkan merupakan langkah protes dengan cara mengirimkan paket palsu agar website perusahaan tidak bisa diakses (Turner 2013).
b. Pada tahun 2014, serangan DoS yang masif dan terdistribusi telah berhasil menyerang perusahaan Incapsula. Perusahaan ini bergerak dibidang platform pengiriman aplikasi berbasis cloud di Amerika. Akibat dari adanya serangan DoS ini, perusahaan setidaknya merugi sebesar $5.000 setiap jamnya. Selain itu, konsekuensi non-finansial juga dialami seperti menurunnya kepercayaan pelanggan, pencurian data pelanggan dan hilangnya properti intelektual (Kovacs 2014).
c. Pada tahun 2015, perusahaan desain grafis bernama Iconfactory menerima permintaan pada server e-mail dengan jumlah besar dalam waktu yang singkat. Gelombang besar itu disebabkan oleh sejumlah besar permintaan

yang seharusnya pergi ke situs lain, dari Facebook ke YouTube, tetapi akhirnya dialihkan ke Iconfactory. Dan semua permintaan itu datang dari negara China. Hal ini berdampak pada e-mail yang tidak dapat diakses baik untuk mengirim atau pun menerima (Brewster 2015).

d. Pada 21 Oktober 2016, penduduk Amerika tidak dapat mengakses Internet. Selain itu, situs-situs ternama seperti Twitter, the Guardian, Netflix, Reddit, CNN, dan lainnya tidak bisa diakses di wilayah Eropa dan Amerika. Hal tersebut dikarenakan serangan DoS yang menyerang perangkat IoT seperti kamera digital dan *DVR player*. Kekuatan serangannya mencapai 1.2Tbps dan melibatkan sekitar 100.000 node (Woolf 2016).
e. Pada tahun 2017, seorang *Web producer* dari *The Journal and Campus Technology* bernama Sri Ravipati melaporkan terdapat beberapa peretas telah menyerang 5000 perangkat IoT pada sebuah kampus. Hal ini berdampak pada jaringan komunikasi baik Internet maupun Intranet yang tidak dapat diakses. Peretas mengakses perangkat IoT dengan password palsu secara brute force dalam jumlah besar (Ravipati 2017).

Serangan DoS pada komunikasi broadcast dapat dihadapi dan diminimalisir dengan adanya proses autentikasi. Proses ini bertujuan untuk memastikan apakah pengguna adalah orang yang berhak atau tidak dalam hal berkomunikasi secara broadcast dengan JSN.

Autentikasi broadcast pada JSN terdiri dari dua jenis yaitu *Message Authentication Code* (MAC) dan *Digital Signature*. Pengelompokan tersebut dipilih berdasarkan mekanisme kriptografi yang dijalankan (Grover & Lim 2015). MAC memanfaatkan kunci yang sama untuk pembentukan maupun verifikasi autentikasi sebagai contoh μ TESLA (Perrig 2001). Hal ini menyebabkan komputasi yang tidak terlalu tinggi, akan tetapi tingkat keamanannya lebih rendah jika dibandingkan dengan penggunaan digital signature. Sedangkan digital signature memiliki dua jenis kunci yaitu kunci privat untuk pembuatan signature dan kunci publik untuk verifikasi. Kedua jenis kunci ini tidak sama tetapi memiliki hubungan dengan persamaan matematika tertentu sebagai contoh (Liu et al. 2012) yang memanfaatkan metode elliptic curve dan (Cheng et al. 2015) yang

memanfaatkan RSA. Kunci publik diketahui oleh penerima sedangkan kunci privat bersifat rahasia dan hanya diketahui oleh pengirim. Mekanisme ini tentunya akan membutuhkan komputasi lebih tinggi dari MAC. Sehingga pemanfaatan keduanya dalam mengamankan JSN menjadi tantangan bagi para peneliti. Tahun 2008, (Ning & Liu 2008) mengemukakan adanya kekurangan pada kedua mekanisme autentikasi tersebut. Baik MAC maupun digital signature masih rentan terhadap serangan DoS. Bahkan dampak terburuk terjadi pada digital signature karena penyerang memanfaatkan komputasi untuk verifikasi yang tinggi dengan mengirim signature palsu dalam jumlah besar. Dari sisi penyerang, penggunaannya jauh lebih efektif daripada pemutusan akses melalui interupsi frekuensi radio (Dong et al. 2013). Serangan ini selanjutnya disebut sebagai serangan DoS berbasis PKC.

Dalam rangka mengatasi permasalahan serangan DoS berbasis PKC, berbagai filter atau metode pre-autentikasi dibangun untuk mendampingi pemanfaatan digital signature. Pada tahun 2000, skema puzzle diperkenalkan oleh (Aura et al. 2000). Selanjutnya, metode tersebut dimodifikasi dengan menggunakan Message Specific Puzzle (Ning & Liu 2008). Mereka menggunakan fungsi hash untuk membangkitkan keychain sebagai kunci sesi dan menghasilkan solusi puzzle dengan tingkat keamanan tertentu. Metode ini memiliki kompleksitas yang rendah namun memerlukan pengirim yang memiliki sumber daya tak terbatas. Hal tersebut diakibatkan delay pada pembentukan solusi puzzle. Pada tahun yang sama, (Du & Chen 2008) mengubah metode puzzle yang memanfaatkan banyak keychain untuk membedakan setiap pengirim. Mekanisme ini bisa mengurangi delay pada sisi pengirim tetapi setiap node sensor harus menyimpan setiap kunci komitmen pengirim untuk proses verifikasi. Hal ini meningkatkan kebutuhan akan ruang penyimpanan pada node sensor. Selanjutnya, (Dong et al. 2013) menggunakan manajemen kunci yang sama tapi menghindari pemanfaatan puzzle, mereka justru memanfaatkan nilai komitmen yang ditambahkan pada setiap pengiriman pesan. Pada tahun 2013, (Tan et al. 2013) mengembangkan cipher puzzle yang menambahkan enkripsi dan mekanisme MAC untuk menjamin kerahasiaan isi dari pesan. Metode ini masih memiliki

kelemahan dalam delay pada sisi pengirim. Delay pada skema puzzle bersifat tidak bisa dikontrol. Nilainya bergantung pada hasil luaran dari fungsi hash karena nilai masukannya terdapat unsur bilangan random. Semakin besar tingkat kekuatan puzzle, jumlah bit yang sama antara luaran hash dengan pola tertentu, maka semakin tinggi rata-rata iterasi hash yang dibutuhkan yang berdampak pada delay pada sisi pengirim yang makin tinggi.

Berdasarkan penjelasan sebelumnya, dapat ditentukan beberapa alasan yang melatarbelakangi pengambilan topik penelitian ini, yaitu:

a. Keamanan merupakan salah satu aspek penting dalam pemanfaatan sebuah teknologi. JSN memiliki keterbatasan sumber daya yang dapat dimanfaatkan oleh peretas untuk mencuri informasi ataupun menjalankan aktifitas yang tidak semestinya dilakukan. Penggunaan metode keamanan yang tinggi bisa meningkatkan kompleksitas komputasi pada node sensor. Oleh karena itu, trade off antara keamanan dan performansi pada JSN perlu menjadi perhatian.
b. Komunikasi broadcast pada JSN sering dijalankan dengan dalih hemat dan efektif mengingat topologi JSN umumnya mesh. Pada implementasinya komunikasi broadcast rentan terhadap serangan DoS yang bisa memperpendek usia node sensor maupun kerugian secara finansial. Penanganan terhadap serangan ini dapat diatasi melalui metode autentikasi. Penggunaan autentikasi harus tepat dan sesuai dengan kondisi JSN dengan memperhatikan aspek performansi, konsumsi sumber daya, tingkat keamanan, dan skalabilitas.
c. Pemanfaatan JSN meningkatkan efektifitas pada aplikasi real time monitoring maupun diseminasi kode atau data sehingga campur tangan manusia pada pemantauan berbagai kondisi, peralatan, aktifitas alam, maupun pengukuran dapat diminimalisir. Kesalahan diagnosa pada penanganan kondisi yang dipantau pun bisa dihindari. Pada beberapa aplikasi, kebutuhan akan pengiriman data yang bersifat rahasia menjadi penting. Pemutakhiran kata sandi atau kunci merupakan salah satu contoh kasus yang sering dijumpai pada hampir semua sistem.
d. Integrasi antara JSN dengan berbagai aplikasi-aplikasi pendukung mulai berkembang. Pengolahan data yang didapatkan dari hasil penginderaan pada

berbagai node sensor bisa menghasilkan informasi untuk bidang-bidang terkait. Pihak-pihak yang mengakses node sensor pun beragam sehingga heterogenitas baik kebutuhan data, platform perangkat keras maupun skema jaringan menjadi tinggi. Skalabilitas dari skema yang akan dikembangkan pada JSN harus menjadi bahan pertimbangan.

1.2. Perumusan Masalah

Beberapa permasalahan pada penelitian ini dapat dirumuskan sebagai berikut:

a. Bagaimana menghadapi serangan DoS berbasis PKC pada JSN dengan kompleksitas yang rendah khususnya untuk node sensor sebagai penerima?
b. Bagaimana peningkatan keamanan dapat diimbangi dengan waktu pemrosesan yang singkat pada sisi pengirim dan penurunan performansi yang rendah pada sisi penerima atau node sensor?
c. Bagaimana membangun skema yang bisa mengirimkan data yang berisi informasi penting maupun bersifat rahasia dan dapat menyesuaikan dengan keterbatasan komputasi dan ruang penyimpanan pada node sensor?
d. Bagaimana membangun metode yang memiliki skalabilitas tinggi terhadap keragaman jumlah pengguna yang akan terhubung dengan JSN?

1.3. Batasan Masalah

Dalam penelitian ini, kami menggunakan beberapa batasan masalah sebagai berikut:

a. Base station atau sink yang bertugas sebagai pengelola terdekat dari node sensor memiliki kemampuan tak terbatas dan aman baik secara fisik maupun aplikasi sehingga tidak dapat diambil alih oleh penyerang.
b. Administrator yang bertugas untuk mengelola komunikasi JSN maupun pengguna memiliki kemampuan tak terbatas dan aman baik secara fisik maupun aplikasi sehingga tidak dapat diambil alih oleh penyerang.
c. Penelitian ini tidak fokus pada protocol broadcast tertentu. Protocol yang digunakan adalah mekanisme *flooding* tanpa ada penambahan komputasi untuk routing.

d. Penelitian ini fokus pada node sensor yang statis.
e. Node sensor memiliki sumber daya yang cukup untuk menjalankan berbagai operasi pada digital signature dan proses enkripsi-dekripsi meskipun jumlahnya terbatas.
f. Layanan *availability* terbatas pada penanganan serangan DoS berbasis PKC.
g. Skalabilitas fokus pada dua hal utama yaitu jumlah pengguna yang beragam dan ketersediaan kunci publik dalam rangka menampung jumlah pengguna yang dapat terhubung ke node sensor.

1.4. Tujuan Penelitian

Adapun tujuan dari penelitian ini adalah :

a. Mengembangkan skema puzzle yang mendampingi signature dalam rangka menjamin autentikasi dan *availability* khususnya terhadap serangan DoS berbasis PKC. Skema ini memiliki kompleksitas verifikasi pada sisi penerima yang rendah, untuk minimalisasi biaya, sehingga dapat diterapkan pada JSN.
b. Membangun skema puzzle dinamis dengan membatasi jumlah iterasi hash pada pengirim sehingga delay dapat dikontrol. Proses ini berdampak pada waktu pemrosesan pengiriman pesan yang rendah dan peningkatan kompleksitas verifikasi yang kecil sehingga dapat diimplementasikan pada node sensor.
c. Mengembangkan skema cipher puzzle dengan nilai kekuatan puzzle yang dinamis pada JSN sehingga data yang ditransmisikan menjadi terenkripsi dan isi datanya tidak dapat diketahui secara langsung oleh peretas. Skema ini dikembangkan agar berdampak minimal pada penurunan performansi terutama pada sisi penerima atau node sensor yang memiliki keterbatasan sumber daya.
d. Mengembangkan metode TinySet pada JSN yang memiliki ketersediaan kunci publik yang besar untuk menghadapi keragaman jumlah pengguna baik dalam proses penambahan maupun pencarian. Selain meningkatkan skalabilitas pengguna yang terhubung pada JSN, metode ini didukung dengan ruang penyimpanan yang rendah sehingga sesuai dengan karakteristik JSN.

1.5. Manfaat Penelitian

Penelitian ini diharapkan dapat bermanfaat pada:

a. Skema puzzle dinamis tanpa enkripsi bisa diimplementasikan untuk aplikasi monitoring pada JSN yang dijalankan secara real time. Jaminan keamanan yang didapatkan bisa diselaraskan dengan waktu pemrosesan yang minim baik disisi pengirim maupun penerima.
b. Skema puzzle dinamis dengan enkripsi bisa diimplementasikan pada diseminasi kode atau data yang dikirimkan oleh administrator kepada node sensor. Proses diseminasi mentransmisikan pesan yang bersifat rahasia sehingga memerlukan proses encoding dengan kebutuhan sumber daya yang bisa dipenuhi oleh node sensor.
c. Integrasi antara JSN dengan aplikasi pendukung IoT lainnya bisa diwujudkan dengan adanya struktur data berbasis peluang untuk menyimpan identitas pengguna. Besarnya jumlah pengguna yang mengakses langsung node sensor tidak lagi menjadi penghalang.
d. Pihak – pihak yang mengakses langsung node sensor bisa dilayani tanpa ada kekhawatiran akan serangan DoS pada JSN. Skema yang diajukan memungkinkan untuk mendeteksi paket palsu sedini mungkin sehingga waktu verifikasi pada setiap paket yang diterima menjadi rendah.

1.6. Kontribusi Penelitian

Kontribusi utama pada penelitian ini adalah:

a. Mengusulkan skema puzzle dinamis sebagai perbaikan skema puzzle eksisting (kekuatan puzzle bersifat statis). Kedinamisan kekuatan puzzle dimaksudkan untuk mengendalikan jumlah iterasi hash sehingga mempersingkat waktu yang dibutuhkan dalam menemukan solusi puzzle dari setiap paket yang akan dikirimkan.
b. Karakterisasi statistik dari kekuatan puzzle untuk membangun fungsi ambang berdasarkan sebaran data. Fungsi ambang ini membatasi jumlah iterasi hash dengan tetap menjamin ditemukannya solusi puzzle.

c. Optimalisasi fungsi ambang dengan metode ekperimental melalui rumus probabilistik yang dijadikan sebagai acuan maksimum iterasi hash pada setiap kekuatan puzzle.
d. Membangun mekanisme tag untuk mengaburkan nilai indeks dan kekuatan puzzle yang ditansmisikan. Kedua parameter ini dikirim secara eksplisit sehingga kompleksitas serangan meningkat terutama dalam memalsukan puzzle palsu yang akan membanjiri JSN.
e. Meningkatkan skalabilitas pada jumlah pengguna yang akan mengakses node sensor dengan pemanfaatan skema keanggotaan TinySet pada JSN.
f. Optimalisasi pemanfaatan skema TinySet pada JSN dengan regularisasi dalam rangka pemanfaatan ruang penyimpanan dan waktu pemrosesan yang lebih baik.
g. Membangun skema cipher puzzle dinamis yang dilengkapi dengan TinySet pada JSN untuk pengguna yang berjumlah besar. Selain memiliki ketahanan terhadap serangan DoS berbasis PKC, skema ini menjamin layanan keamanan autentikasi melalui digital signature dan layanan keamanan confidentiality melalui enkripsi.
h. Membangun library pada Arduino untuk proses enkripsi dan dekripsi menggunakan algoritma RC5.

BAB 2
DASAR TEORI DAN KAJIAN PUSTAKA

Bab ini menjelaskan mengenai landasan teori dari fokus penelitian dan posisi penelitian (*state of the art*) dari autentikasi JSN khususnya untuk menghadapi serangan DoS berbasis PKC. Gambaran umum, penggunaan dan aspek keamanan pada JSN dibahas secara detail pada Bab 2.1. Kemudian, posisi penelitian autentikasi pada JSN dijelaskan pada Bab 2.2. Sedangkan, detail dari metode filter pada autentikasi pada JSN sebagai solusi menghadapi serangan DoS berbasis PKC dibahas pada Bab 2.3.

2.1. Jaringan Sensor Nirkabel (JSN)

Era IoT saat ini memungkinkan semua benda dapat tergabung dan terpantau melalui Internet. Peralatan penghubung yang melekat pada benda tersebut dapat berupa sensor, smart card, RFID, dan perangkat keras berdaya rendah lainnya. Berbeda dengan yang lain, sensor secara khusus digunakan untuk merekam kondisi fisik lingkungan sehingga bisa dipantau dari jarak jauh. Sensor pada umumnya digunakan untuk merekam suhu, kelembaban, tekanan, cahaya, suara, getaran, kadar gas tertentu, gerakan, keberadaan sebuah objek dsb pada daerah yang susah dijangkau. Hal tersebut menyebabkan susahnya dilakukan pemeliharaan peralatan baik kondisi fisik maupun perangkat lunaknya. Selain itu jumlah sensor node sendiri umumnya berjumlah banyak. Semakin luas daerah yang dipantau dan semakin banyak parameter yang direkam maka semakin besar jumlah sensor node yang dibutuhkan. Terlebih lagi kondisi lingkunan yang tidak memungkinkan untuk pemasangan kabel atau perangkat pendukung berdaya tinggi sehingga sensor dihubungkan tanpa melalui kabel. Itulah mengapa jaringan sensor nirkabel dibangun dan berkembang pesat saat ini.

Jaringan sensor nirkabel sendiri diartikan sebagai kumpulan autonomous sensor dan aktuator dengan infrastruktur komunikasi nirkabel, dengan maksud untuk mengawasi dan mengontrol kondisi fisik lingkungan pada lingkungan yang tidak terjangkau dan secara kooperatif meneruskan datanya ke

lokasi utama termasuk juga meneruskan perintah menuju aktuator tujuan melalui jaringan (Yang & Cao 2008). Jaringan sensor nirkabel terdiri dari ratusan atau bahkan ribuan node yang terhubung antara satu dengan yang lain baik secara langsung maupun tidak. Dengan kata lain, sensor node terhubung secara multihop. Sehingga tugas sensor selain merekam data juga meneruskan dan pertukaran data baik dengan jaringan lokal maupun eksternal. Seperti yang dijelaskan sebelumnya, sensor memiliki keterbatasan pada daya sehingga jangkauan jaringan antar sensor pun rendah dengan bandwidth pertukaran data maksimal 250 kBps dan kemampuan komputasi yang rendah. Meskipun istilah rendah dan ringan pada sumber daya sering digunakan akan tetapi tidak ada definisi yang akurat mengenai hal tersebut (Pei et al. 2018). Berbagai proses dan fungsi yang terlibat pada sebuah skema harus menjadi bahan pertimbangan dalam perhitungan komputasi. Semakin rendah jumlah fungsi dan prosedur pada sebuah skema diartikan sebagai penyederhanaan sistem yang berdampak pada rendahnya komputasi. Node sensor didalamnya dilengkapi dengan mikrokontroller, unit pengecap, perangkat penyimpanan, penerima dan berbagai komponen tambahan lainnya. Tidak sedikit pula node sensor hanya dilengkapi dengan baterai sebagai sumber daya karena letaknya yang jauh dari listrik. Oleh karena itu penghematan energi yang berdampak semakin panjang usia node sensor menjadi kebutuhan utama (Al-Riyami et al. 2016; Pei et al. 2018).

Selanjutnya akan dibahas detail pemanfaatan jaringan sensor nirkabel pada dunia nyata, keamanan yang diperlukan dan perkembangan penelitian pada keamanan jaringan sensor nirkabel.

2.1.1. Aplikasi pada diseminasi data atau kode

Jaringan sensor nirkabel dibangun dengan perangkat yang memiliki keterbatasan sumber daya, sehingga tantangan utamanya adalah bagaimana membangun jaringan yang efisien. Hal tersebut diikuti dengan lingkungan sekitar yang dinamis. Berikut karakteristik dari pembangunan jaringan sensor nirkabel (Manjeshwar & Agrawal 2001) :

1. Data Centric. Tidak seperti aplikasi pada umumnya yaitu pengguna mengakses pada node tertentu, aplikasi pada jaringan sensor nirkabel lebih mengutamakan kebutuhan informasi terhadap parameter tertentu.
2. Application-specific. Kebutuhan jaringan sangat bergantung pada aplikasi yang mengakses dan tujuan dibangunnya jaringan sensor nirkabel.
3. Node yang bertetangga sangat dimungkinkan mempunyai informasi yang sama. Lokasi atau penempatan sebuah node menjadi penting. Selain itu, sistem sleep juga bisa digunakan untuk meningkatkan efisiensi perangkat. Dalam rangka menghindari redudansi, akan lebih baik mengirim data yang saling melengkapi atau penggabungan dibandingkan sendiri-sendiri.
4. Pemanfaatan ID sensor menjadi tidak efisien. Hal tersebut dikarenakan jumlah sensor yang besar sehingga memungkinkan ukuran jumlah ID lebih besar dari data yang ditransmisikan. Selain itu, sensor node bersifat data-centric mengakibatkan tidak perlunya mengetahui ID sensor tertentu. Oleh karena itu, keunikan node sensor akan diidentikkan dengan ip address pada IoT.

Dari karakteristik tersebut, jaringan sensor nirkabel telah banyak digunakan pada aplikasi di dunia nyata khususnya untuk tujuan pelacakan sebuah target, deteksi kejadian tertentu, dan monitoring atau pengawasan (Anastasi 2011). Contoh kasus pada proses monitoring antara lain aplikasi pemantauan polusi untuk tata lingkungan kota; *Precision Agriculture Monitor System* (PAMS) yaitu sistem cerdas untuk memantau lingkungan tanaman pertanian dan memberikan layanan kepada petani (Li et al. 2011); struktur dan pembangunan fasilitas kesehatan. Implementasi dari pemanfaatan JSN untuk deteksi kejadian yaitu pencegahan terhadap pencurian, kebakaran, banjir, tsunami maupun bencana alam lainnya (Anastasi 2011). Sedangkan untuk fungsi pelacakan target umumnya digunakan untuk pengawasan target bergerak. Pada berbagai aplikasi tersebut tugas spesifik dari JSN yang saat ini sedang berkembang adalah diseminasi data maupun kode (Hamida & Chelius 2008; Hyun & Ning 2008; Lim 2011; He et al. 2012; Tan et al. 2013; He et al. 2013). Diseminasi data merupakan aktivitas dengan data hasil perekaman lingkungan diteruskan pada perangkat lain untuk diolah pada aplikasi dengan tingkat yang lebih tinggi. Sedangkan pada diseminasi

kode, data yang diteruskan merupakan kode program, perintah ataupun perbaikan bug. Sehingga baik data maupun diseminasi kode, informasi yang diteruskan menjadi penting dan rahasia baik dari maupun menuju node sensor. Selain diseminasi kode maupun data, *data collection* juga banyak dikembangkan sebagai proses komunikasi yang penting antara node sensor dengan base station (pemroses data dari informasi yang didapatkan dari node sensor) (Anastasi 2011). Pada berbagai kasus spesifik tersebut, pemanfaatan JSN berdasarkan mekanisme pelaporan data dibagi menjadi tiga yaitu: *Periodic-sensing, event-driven* dan berbasis query (Hamida & Chelius 2008).

Periodic-sensing merupakan suatu kebutuhan aplikasi JSN dengan node sensor yang selalu aktif memonitor kondisi lingkungan dan secara teratur melaporkan hasilnya pada base station, sebagai contoh aplikasi pengawasan cuaca. Dengan kata lain node sensor bertindak proaktif pada setiap interval waktu tertentu. Pada JSN yang memenuhi *periodic-sensing* permasalahan utama yang ingin ditangani adalah manajemen pemanfaatan node sensor dalam rangka memaksimalkan masa hidupnya dan penghematan *bandwidth*, sehingga berkembanglah berbagai protokol untuk mentransmisikan data secara berkala agar lebih efisien (Ye et al. 2002; Lu et al. 2014).

Event-driven merupakan suatu kebutuhan aplikasi JSN dengan node sensor yang beroperasi untuk memonitor secara diam-diam dan diprogram untuk melaporkan kegiatan berdasarkan aksi tertentu sebagai contoh: keberadaan sebuah objek pada intrusion detection, pelacakan target, atau aplikasi berbasis militer. Dengan kata lain sensor node bertindak reaktif terhadap sebuah kejadian.

Aplikasi berbasis query merupakan suatu kebutuhan aplikasi JSN agar hasil penginderaan dari node sensor yang tersedia di jaringan ditransmisikan menuju base station sebagai reaksi dari permintaan pengguna akan ukuran atau kejadian tertentu. Aplikasi berbasis query biasanya bertujuan untuk monitoring terhadap parameter tertentu dan berguna untuk aplikasi data collection (Chen et al. 2009). Berbeda jika penggunaan berdasarkan query pada aplikasi *real-time*, pengguna membutuhkan akses langsung menuju node sensor. Hal ini dikarenakan, pengguna membutuhkan data faktual sesaat setelah kejadian atau yang saat itu

juga ditangkap oleh node sensor (Kumar et al. 2012). Perbedaan dengan kebutuhan e*vent-driven* yaitu pada apllikasi berbasis query proses pergerakan data diperlukan sedangkan e*vent-driven* bersifat mitigasi resiko.

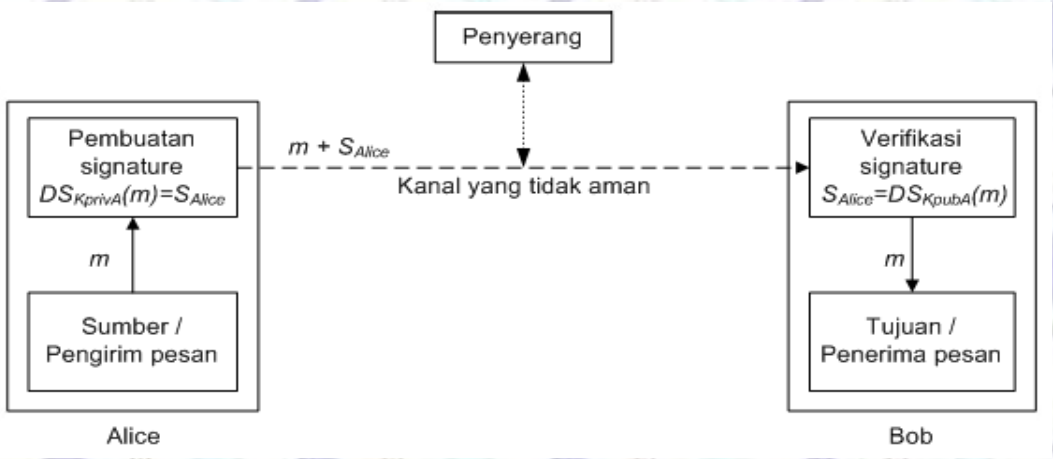
2.1.2. Keamanan dan pertahanan terhadap DoS

Keamanan JSN tidak bisa disamakan dengan perangkat keras lainnya dikarenakan karakteristiknya berupa node bersumber daya rendah namun berskala besar yang tersebar di lingkungan geografis yang luas dan susah dijangkau. Hal tersebut memungkinkan adanya penyusup atau berbagai ancaman keamanan baik fisik maupun non fisik. Berbagai keamanan yang ada tidak bisa langsung diimplementasikan pada node sensor secara langsung. Hal tersebut dikarenakan terbatasnya kemampuan komputasi, penyimpanan, daya, transmisi dan jangkauan dari node sensor. Tantangan tersebut yang menjadi bahan penelitian beberapa tahun terakhir.

Berdasarkan rekomendasi X.800 (ITU-T (CCITT) Recommendation 1991), layanan keamanan terbagi menjadi enam kelompok utama yaitu *authentication*, *access control*, *confidentiality*, *integrity*, *non-repudiation* dan *service availability*. Masing-masing layanan tersebut dirancang untuk menghadapi gangguan dari penyerang yang dirangkum pada (Yang 2013).

Tabel 2.1 Layanan keamanan beserta serangannya

Serangan	Layanan keamanan					
	authentica-tion	*access control*	*confidentiality*	*integrity*	*non-repudiation*	*service availability*
Terkuak-nya isi pesan			√			
Penyang-kalan partisipasi					√	
Penyamar-an	√	√				
Replay				√		
Modifika-si Pesan				√		
DoS						√

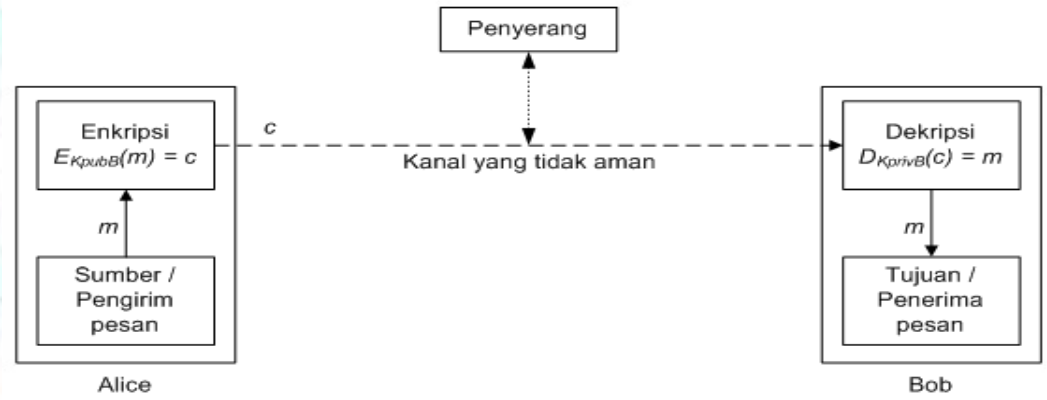


Gambar 2.1 Mekanisme penggunaan digital signature

Authentication merupakan layanan keamanan untuk memastikan identitas dari pengakses jaringan (Yang 2013). Mekanisme ini digunakan untuk memastikan bahwa pengguna yang mengirim atau berinteraksi dengan sistem merupakan pengguna yang berhak. Pihak penyerang tidak bisa menyamar dan menyusup ke dalam sistem. Fokus utama pada penelitian ini adalah autentikasi pada JSN yang memanfaatkan *signature* atau kunci asimetris di mana ilustrasi dari komunikasi antar pengirim (Alice) dan penerima (Bob) diperlihatkan pada Gambar 2.1. Pembuatan *signature* hanya bisa dijalankan oleh pemilik kunci privat sedangkan pihak penerima bisa menyatakan bahwa pesan ini benar milik pengirim jika dan hanya jika penerima memiliki kunci publik dari pengirim (Hogg 2005).

Access control merupakan layanan keamanan yang mengatur hak akses dari sumber daya yang tersedia pada sistem. Hal tersebut dimaksudkan untuk menghindari pemanfaatan sumber daya untuk kegiatan ilegal dengan cara akses yang tidak sah pula. Tujuan utama dari mekanisme *access control* pada JSN adalah meminimalisir node ilegal untuk bergabung pada kumpulan node sensor yang legal sehingga memperpanjang masa hidup node sensor yang legal (Zhou et al. 2007).

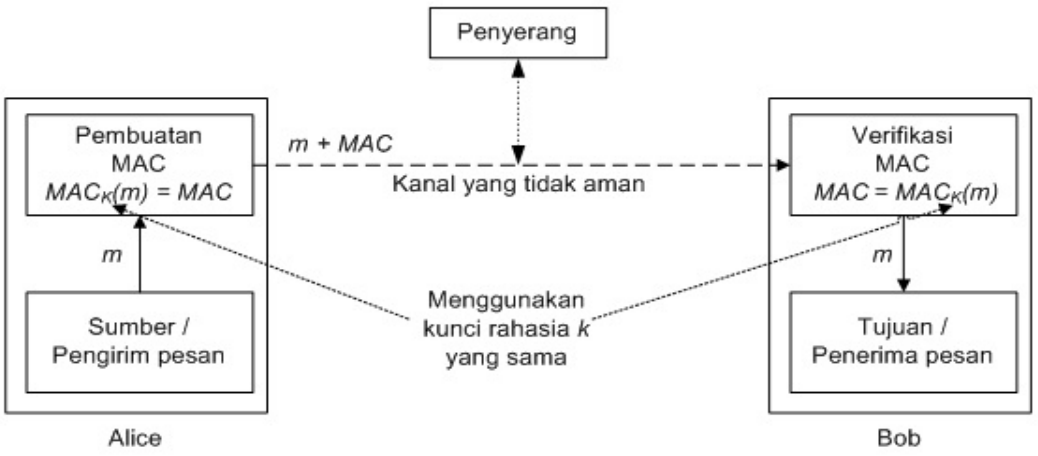
Confidentiality adalah layanan keamanan untuk menjaga keamanan data yang ditransmisikan dari pihak ketiga yang tidak berhak untuk mengetahui isinya melalui komunikasi apapun. Penyerang dapat mengakses secara ilegal dengan cara langsung maupun tidak langsung. Cara langsung yaitu dengan



Gambar 2.2 Mekanisme enkripsi

menghadang pesan yang dikirim kemudian dimodifikasi atau diganti dengan pesan palsu yang telah disiapkan. Sedangkan cara tidak langsung, diawali dengan pengamatan dan analisa pola data yang ditransmisikan oleh jaringan. Hal utama yang dijalankan untuk layanan *confidentiality* yaitu dengan proses enkripsi. Ilustrasi dari proses enkripsi dapat dilihat pada Gambar 2.2.

Integrity adalah layanan keamanan yang menjamin informasi yang diterima asli tanpa modifikasi dari pihak yang tidak berhak. Modifikasi disini dapat berupa penambahan, pengurangan maupun pengiriman ulang informasi tanpa terdeteksi atau sepengetahuan dari pengguna yang sah. Beberapa mekanisme yang memastikan pesan tidak diubah yaitu *Message Authenticate Code* (MAC) yang diilustrasikan pada Gambar 2.3, fungsi hash dan enkripsi (Yang 2013).



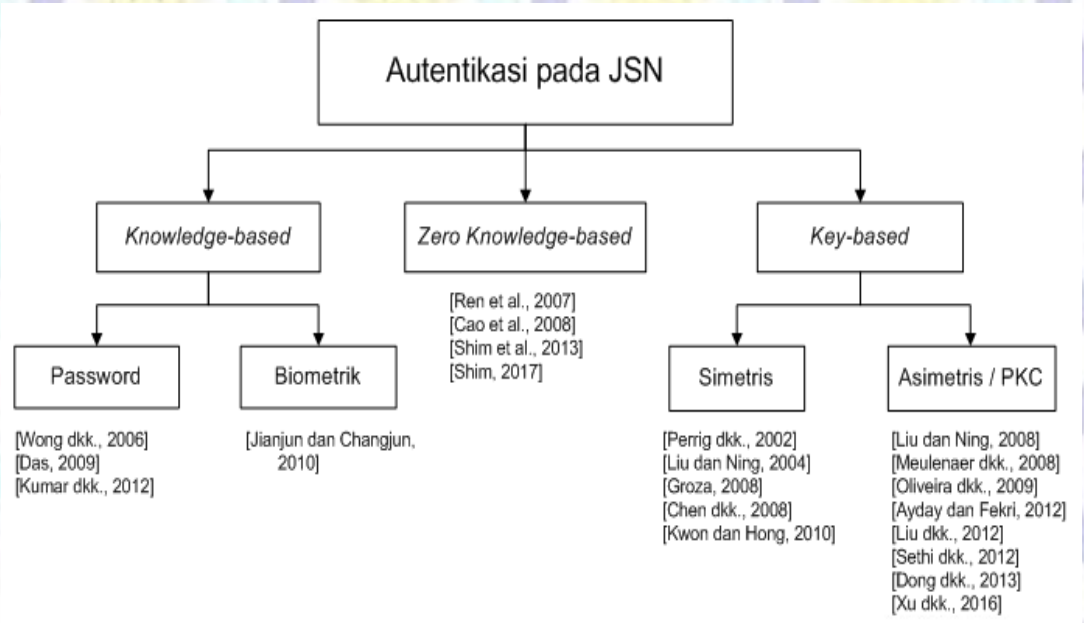
Gambar 2.3 Mekanisme MAC

Non-repudiation merupakan layanan keamanan untuk menghindari penyangkalan pengguna terhadap keikutsertaan dalam berpartisipasi pada sebuah komunikasi. Partisipasi yang dimaksud baik dalam segi pengirim maupun penerima. Tujuan tersebut bisa dipenuhi dengan adanya digital signature (Yang 2013).

Service availability adalah layanan keamanan untuk memastikan ketersediaan informasi kapanpun dibutuhkan dan secara khusus melindungi dari serangan DoS. Layanan ini melindungi dari serangan yang mencoba untuk mengambil alih energi dengan berbagai cara sehingga tidak bisa diakses oleh pihak yang justru berhak (Yang 2013). Metode yang bisa digunakan untuk menghadapi serangan DoS dan menjamin ketersediaan sebuah sistem yaitu metode filter yang menjadi fokus pada penelitian ini.

2.2. Kajian Pustaka

Penelitian ini akan membahas mengenai proses autentikasi pada JSN yang umumnya dimanfaatkan untuk kasus *data collection*, diseminasi kode maupun data. Mekanisme autentikasi broadcast pada jaringan sensor dibagi menjadi tiga kelompok utama yaitu *knowledge-based*, *zero knowledge-based* dan



Gambar 2.4 Taksonomi metode autentikasi pada JSN

key-based (Menezes et al. 1996). Taksonomi dari penelitian yang berkembang pada autentikasi di JSN diperlihatkan pada Gambar 2.4.

Autentikasi *knowledge-based* (berbasis pengetahuan) merupakan autentikasi yang didasarkan pengetahuan pada setiap pengirim yang tersimpan di sisi penerima. Pengetahuan tersebut unik untuk setiap pengirim, dapat disimpan pada saat awal dibangunnya sebuah JSN maupun penambahan anggota baru sebagai anggota kelompok sebuah jaringan. Terdapat beberapa kelompok pengetahuan yang dapat digunakan, yaitu password (Wong et al. 2006; Das 2009; Jianjun & Changjun 2010) dan biometrik (Jianjun & Changjun 2010). Pemanfaatan autentikasi berbasis pengetahuan pada JSN membutuhkan komputasi yang rendah. Hal tersebut dikarenakan kemampuan yang dibutuhkan hanyalah mencocokkan password atau biometrik yang diterima dengan yang disimpan di dalam node sensor. Akan tetapi metode tersebut membutuhkan ruang penyimpanan yang semakin membesar seiring dengan meningkatnya jumlah pengirim. Selain itu, ukuran password maupun biometrik menjadi beban jaringan padahal semakin panjang ukuruannya maka keamanan semakin meningkat. Hal tersebut berbanding terbalik dengan kebutuhan *bandwidth*.

Autentikasi *zero knowledge-based* merupakan autentikasi dengan penerima yang mendapatkan pengetahuan sebuah rahasia dari pengirim tanpa mengungkapkan informasi apapun (di luar apa yang bisa didapatkan penerima sebelum protokol dijalankan) (Menezes et al. 1996). Dengan kata lain, dari pesan yang diterima penerima memverifikasi dengan menjalankan algoritma pembuktian tertentu. Parameter yang digunakan untuk pembuktian bersifat unik dan melekat pada ciri-ciri pengirim. Semakin tinggi tingkat anonimity semakin tinggi kompleksitas komputasi di sisi penerima. Oleh karena itu beberapa algoritma pada *zero knowledge* masih memanfaatkan *digital signature* untuk mengurangi tingkat kompleksitas (Cao et al. 2008). Beberapa skema pada *zero knowledge* membutuhkan tenaga yang cukup tinggi dan waktu pemrosesan verifikasi yang tinggi pula. Sebagai contoh IBS-pairing, yang operasi bi-linear komputasinya hampir setara dengan 4 kali operasi multiplikasi pada *digital signature*. Padahal, multiplikasi adalah operasi dengan kebutuhan energi

tertinggi pada *digital signature*. Contoh lain yang lebih spesifik menggunakan *identity-based signature* yaitu (Ren et al. 2007; Shim et al. 2013; Shim 2017). Dari beberapa contoh pemanfaatan autentikasi sebelumnya, belum ada skema pada *zero knowledge* yang mengatasi DoS. Seperti diketahui, implementasi autentikasi menggunakan *digital signature* membutuhkan komputasi tinggi di sisi penerima dalam proses verifikasi. Hal tersebut bisa dimanfaatkan oleh penyerang dengan membanjiri penerima menggunakan paket palsu. Serangan DoS semacam ini disebut DoS berbasis PKC dan menjadi fokus masalah pada penelitian ini.

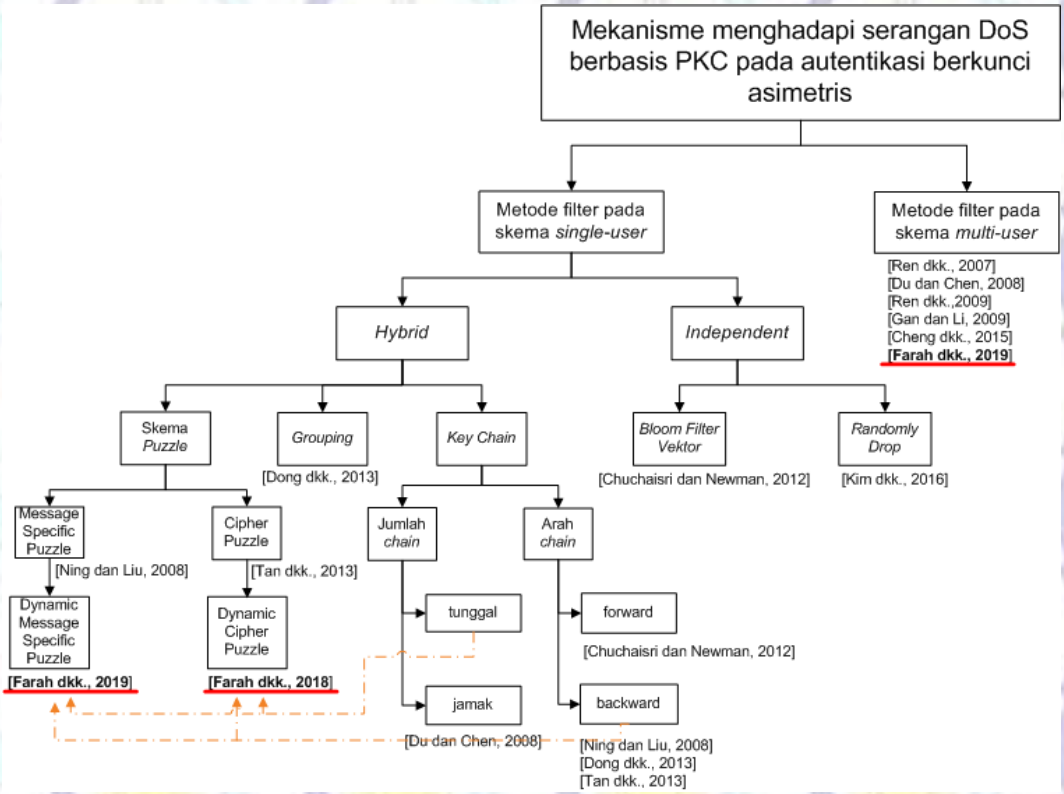
Autentikasi berdasarkan kunci dibagi kembali menjadi dua berdasarkan blok kriptografi yang membangunnya. Yang pertama yaitu simetris karena kunci yang digunakan untuk membangun autentikasi di sisi pengirim sama dengan kunci yang digunakan untuk verifikasi pada sisi penerima. Nama lain dari blok kriptografi tersebut yaitu MAC. Implementasi MAC yang sesuai dengan karakteristik JSN yaitu µTESLA (Perrig et al. 2002), multilevel µTESLA (Liu & Ning 2004), unbounded one-way chains (Groza 2008), X-TESLA (Kwon & Hong 2010), termasuk juga modifikasinya yang menggunakan teknik Bloom Filter (Chen et al. 2008). Blok simetris terkenal akan keefisiensinya. Selain itu, proses komputasi juga kebutuhan penyimpanan yang rendah. Akan tetapi memiliki kelemahan dalam rendahnya keamanan terutama dalam pre-distribusi kunci, kebutuhan akan sinkronisasi waktu dan *delayed authentication*. Hal tersebut dikarenakan tidak dikirimnya MAC dengan pesannya secara bersamaan sehingga mengakibatkan adanya delay yang bisa dimanfaatkan oleh penyerang. Selain itu skalabilitas yang rendah membuat pemanfaatan TESLA menjadi tidak sesuai untuk jaringan sensor berskala besar yang memiliki keragaman dalam besaran jumlah node sensor. Di tahun 2005, terdapat variasi TESLA yang mendukung skalabilitas di sisi penerima yaitu multi sender µTESLA (Liu et al. 2005). Variasi TESLA tersebut memanfaatkan penjadwalan dalam pengiriman pesan yang sudah terdefinisi sebelumnya. Pengiriman pesan harus dijalankan secara bergantian antara pengirim dan tidak dapat dilakukan secara bersamaan.

Hal itulah yang menyebabkan tidak adanya kedinamisan pada multi sender µTESLA (Prasuna & Hemalatha 2013).

Tabel 2.2 Penelitian terkait autentikasi menggunakan kunci pada JSN

No	Penelitian	Kontribusi	Peluang penelitian
1	(Perrig et al. 2002) SPINS: Security Protocols for Sensor Networks	- Perancangan dan pembangunan skema autentikasi broadcast µTESLA untuk perangkat berkomputasi rendah - Kriptografi simetri yang cepat dengan biaya komunikasi yang rendah	lemah terhadap serangan DoS karena *delayed authentication*
2	(Meulenaer et al. 2008) On the Energy Cost of Communication and Cryptography in Wireless Sensor Networks	Investigasi kebutuhan energi pemanfaatan protokol kriptografi kerberos dan ECDSA yang memanfaatkan pertukaran kunci Elliptic Curve-Diffie Hellman (ECDH-ECDSA) pada platform sensor MICAz dan TelosB	Biaya komunikasi dan komputasi relatif tinggi sehingga kurang efisien dalam pengaplikasiannya di JSN
3	(Liu et al. 2012) PKC Based Broadcast Authentication using Signature Amortization for WSNs	Pemanfaatan *Signature Amortization* dengan menggunakan satu signature ECDSA untuk meng-autentikasi semua pesan broadcast dengan overhead yang dibagikan pada semua penerima sehingga keamanan tinggi dengan biaya rendah	Aspek serangan DoS bukan menjadi fokus utama sehingga tidak ada penjelasan mengenai akibat dari serangan tersebut pada metode yang diusulkan
4	(Sethi et al. 2012) End-to-end Security for Sleepy Smart Object Networks	- Constrained Application Protocol (CoAP) yang menjamin integritas dan autentikasi pada perangkat multihop berdaya rendah - Implementasi PKC yang umum tersedia pada perangkat keras dengan mikrokontroller 8 bit	Fokus penelitian pada layer 3 dengan kurangnya investigasi mengenai multicast, komunikasi kelompok, dan pengaturan kongesti
5	(Xu et al. 2016) Improving Efficiency of Authenticated OpenFlow Handsh	Pengintegrasian antara Software Defined Internet of Things (SDIoT) dengan perangkat keras chip ECC dan pembangunan protokol handshake sehingga proses komputasi pada IoT lebih ringan dan memiliki tingkat keamanan yang tinggi	Tidak membahas mengenai proses verifikasi hanya fokus pada pertukaran kunci pada autentikasi asimetri menggunakan ECC

Autentikasi dengan kunci asimetris atau biasa disebut *Public Key Cryptography* (PKC) adalah autentikasi yang kuncinya digunakan untuk pembuatan signature di sisi pengirim (*private key*) berbeda dengan kunci yang digunakan untuk verifikasi signature di sisi penerima (*public key*). Dibandingkan dengan kunci simetris, skema ini memiliki kelebihan dalam waktu autentikasi yang dapat dilakukan segera setelah pesan diterima oleh node sensor. Selain itu, skema ini juga menjamin keamanan dalam poin *non-repudiation*. Implementasi PKC sendiri dapat dibagi menjadi *one-time signature* dan *public key digital signature* (Grover & Lim 2015). *One-time signature* memanfaatkan *one-way key chain* untuk memproduksi *public key*. *Private key* bersifat rahasia pada sisi pengirim dan digunakan sebagai dasar pembuatan *public key*. Wilayah aplikasi yang sesuai menggunakan skema tersebut antara lain pada kasus dengan pesan yang tidak dikirim dengan intensitas yang tinggi pada waktu yang tidak dapat diprediksi dengan kata lain non periodik. Hal tersebut dikarenakan keterbatasan jumlah pesan yang dapat diautentikasi dan ukuran *signature* yang besar. Pemanfaatan *one-time signature* pada JSN yaitu BiBa (Perrig 2001), HORS (Reyzin & Reyzin 2002), dan perbaikan performansi dari HORS (Shiuhpyng et al. 2006). Sedangkan *digital signature* yang menggunakan kunci publik memiliki kelebihan dalam jumlah *signature* yang tidak terbatas. Hal tersebut dikarenakan dipublikasinya *public key* tidak akan bisa menyebabkan terkuaknya *private key*. Skema ini memiliki jaminan keamanan akan tetapi diikuti dengan tingginya komputasi dan komunikasi sehingga bisa memberatkan *bandwidth* jaringan. Meskipun demikian, JSN saat ini sudah bisa mengimplementasikan skema ini dengan berbagai penyesuaian seperti (Oliveira et al. 2009; Ayday & Fekri 2012; Liu et al. 2012; Dong et al. 2013; Cheng et al. 2015). Dari beberapa algoritma *digital signature* ECC sangat direkomendasikan untuk JSN. Meskipun komputasinya terbilang cukup tinggi, akan tetapi untuk sejenis PKC yang lain metode tersebut lebih rendah dengan ukuran *signature* yang relatif kecil (Cao et al. 2008). Bahkan beberapa skema sudah dapat diimplementasikan secara fisik seperti yang terdapat yaitu MICAz 2, TelosB,



Gambar 2.5 Taksonomi autentikasi menggunakan PKC pada JSN

Tmote Sky, dan Imote2 (Liu & Ning 2008) ; MICAz dan TelosB (Meulenaer et al. 2008) ; Arduino (Xu et al. 2016).

Seperti yang dibahas sebelumnya, salah satu tujuan dari penelitian ini adalah menghadapi serangan PKC-based DoS untuk autentikasi berkunci asimetris pada JSN. Fokus dari serangan ini bukan untuk mendapatkan kunci ataupun sertifikat pengguna, melainkan untuk menghabiskan energi dan meningkatkan waktu respon dari node sensor (Gan & Li 2009). Dalam menghadapi serangan tersebut, terdapat dua skema berdasarkan jumlah pengirim yang mengakses jaringan yaitu *single-user* dan *multi-user*. Pada skema *single-user*, pengirim berjumlah tunggal dan metode yang berkembang untuk menghindari PKC-based DoS adalah *filtering* atau bisa juga disebut *pre-authentication filter*. Detail taksonomi dari metode filter baik untuk single-user maupun *multi-user* dapat dilihat pada Gambar 2.5.

Secara umum, metode filter pada *single-user* dapat dikelompokkan ke dalam skema hybrid dan independen. Skema independen memiliki metode utama

yang berdiri sendiri tanpa digabungkan atau modifikasi dengan metode lain. Metode yang berdiri sendiri antara lain menggunakan skema *Bloom Filter Vector* (BFV) yaitu *improved key pool* (Chuchaisri & Newman 2012) dan *Randomly Drop* pada DoS attack resistant scheme (Kim & An 2016). Sedangkan untuk metode hibrid dapat dibagi lagi menjadi *grouping* dan keychain. Contoh pemanfaatan metode *grouping* yaitu *Group based* (Dong et al. 2013) dan *Group based adaptive-regroup* (Dong et al. 2013). Selain itu, metode yang paling banyak digunakan yaitu keychain karena efisien, sederhana tetapi tetap memiliki tingkat keamanan yang tinggi. Pada implementasinya, keychain dapat dikelompokkan berdasarkan jumlah dan arah chain. Pada pembagian jumlah chain terdapat dua jenis keychain yaitu tunggal yang dimanfaatkan oleh key chain (Chuchaisri & Newman 2012; Dong et al. 2013) dan Message Specific Puzzle (MSP) (Ning & Liu 2008). Begitu pula skema yang akan diajukan pada penelitian ini. Sedangkan pada jumlah chain yang jamak dan berarah *backward* digunakan oleh *Pre-Authenticator* (Du & Chen 2008). Metode lain yang memanfaatkan arah *backward* yaitu (Dong et al. 2013) dan MSP (Ning & Liu 2008). Sebaliknya hanya (Chuchaisri & Newman 2012) yang menggunakan arah keychain *forward*.

Terdapat beberapa metode menghadapi serangan DoS berbasis PKC khususnya skema multi-user. Beberapa yang cukup terkenal dan banyak digunakan yaitu BAS, DAS, CAS, HAS dan MAS (Ren et al. 2007; Ren et al. 2009). Pada implementasinya autentikasi yang digunakan pada berbagai mekanisme tersebut adalah ECC. Pada tahun 2015, mekanisme menyerupai RSA dibangun untuk autentikasi pengguna (Cheng et al. 2015). Meskipun memiliki komputasi yang rendah, akan tetapi biaya komunikasi yang dibutuhkan sangat tinggi yaitu dua kali lipat daripada penggunaan ECC. Berbagai metode tersebut fokus pada mekanisme pembangunan sertifikat untuk autentikasi pengguna. Sedangkan RRAS merupakan metode autentikasi multi-user yang fokus pada pembagian beban verifikasi pada node sensor (Gan & Li 2009). Penelitian-penelitian terkait pertahanan terhadap serangan DoS pada JSN dirangkum pada Tabel 2.3.

Tabel 2.3 Penelitian terkait metode filter dalam menghadapi DoS pada JSN

No	Penelitian	Kontribusi	Peluang penelitian
1	(Aura et al. 2000) DOS-resistant Authentication with Client Puzzles	- Pemanfaatan skema puzzle untuk menghadapi serangan DoS pada autentikasi client-server	Memungkinkan adanya serangan yang memanfaatkan puzzle palsu selain itu delay pada saat pengiriman akan meningkat seiring dengan peningkatan tingkat keamanan
2	(Ning & Liu 2008) Mitigating DoS Attacks against Broadcast Authentication in Wireless Sensor Networks	- Pemanfaatan ECDSA dalam autentikasi JSN sehingga meningkatkan kompleksitas serangan - *weak authenticator*, message spesific puzzle (MSP) yang memanfaatkan skema puzzle dan kunci sesi dengan keychain dalam menghadapi serangan DoS baik untuk autentikasi simetri maupun asimetri	Efisien dalam verifikasi akan tetapi butuh pengirim yang powerful dikarenakan delay dalam pembangunan solusi puzzle
3	(Chuchaisri & Newman 2012) Fast Response PKC-Based Broadcast Authentication in Wireless Sensor Networks	- Skema broadcast dengan tujuan kecepatan komunikasi dan keamanan juga overhead seminimal mungkin pada perangkat yang bertenaga rendah, dengan cara mininalisasi delay dan menghindari DoS pada autentikasi broadcast - Pemanfaatan skema key pool dan key chain yang didalamnya memanfaatkan digital signature, bloom filter dan distribusi secret key antar sensor	- Implementasi hanya pada tahap simulasi karena digital signature masih dianggap terlalu mahal untuk JSN - Fokus pada manejemen kunci - Memiliki beberapa level sinkronisasi karena sistem broadcast yang dijalankan secata periodik

Lanjutan Tabel 2.3

No	Penelitian	Kontribusi	Peluang penelitian
4	(Dong et al. 2013) Providing DoS resistance for signature-based broadcast authentication in sensor networks	filter yang dijalankan sebelum autentikasi sehingga pesan palsu sebelum verifikasi signature dapat minimalisir sehingga metode ini bisa diterapkan pada JSN multihop	Analisa dijalankan menggunakan simulasi dan belum diimplementasikan secara real pada sensor. Komputasi filter lebih rendah dari signature akan tetapi lebih tinggi dibandingkan fungsi hash
5	(Tan et al. 2013) A confidential and DoS-resistant multi-hop code dissemination protocol for wireless sensor networks	Pembangunan skema yang menjamin autentikasi dan confidentiality pada diseminasi kode atau data di JSN yang multi hop. Pembangkitan signature dengan menggunakan Tiny ECC dan cipher puzzle digunakan untuk ketahanan terhadap serangan DoS	Penggunaan skema puzzle yang masih rentan terhadap delay jika tingkat keamanannya tinggi.
6	(Ren et al. 2009) Multi-User Broadcast Authentication in Wireless Sensor Networks	- Skema autentikasi broadcast berbasis signature pada multi-user JSN tahan terhadap DoS - Memanfaaatkan metode sertifikat (CAS), *Direct-storage* (DAS), bloom filter (BAS), Hybrid antara bloom filter dan merkle hash tree (HAS)	Metode yang diajukan memiliki keterbatasan akan biaya komunikasi (CAS) dan ruang penyimpanan (DAS, BAS maupun HAS) untuk diimplementasikan pada JSN yang memiliki keterbasan sumber daya
7	(Du & Chen 2008) Defending DoS Attacks on Broadcast Authentication in Wireless Sensor Networks	Autentikasi broadcast pada JSN memanfaatkan keychain dalam jumlah yang sesuai dengan jumlah pengguna	storage overhead yang tinggi karena menyimpan semua identitas pengguna beserta kunci komitmen dari setiap pengguna tersebut

Lanjutan Tabel 2.3

No	Penelitian	Kontribusi	Peluang penelitian
8	(Gan & Li 2009) A Multi-user DoS-containment broadcast authentication scheme for wireless sensor networks	Multi user autentikasi broadcast menggunakan signature dengan memanfaatkan bloom filter dan skema autentikasi random berbasis reputasi (RRAS) untuk menghadapi serangan DoS	Jumlah pengguna terbatas dikarenakan tingginya ruang penyimpanan yang dibutuhkan untuk menyimpan informasi reputasi dari setiap node
9	(Cheng et al. 2015) Scheme for Multiuser Broadcast Authentication in Wireless Sensor Networks	verifikasi validitas tanpa bufer dari tiap query yang di broadcast oleh user dari perangkat bergerak yang memanfaatkan RSA sebagai signature	Membutuhkan komputasi dari pihak ketiga sebagai *certificate-authority*

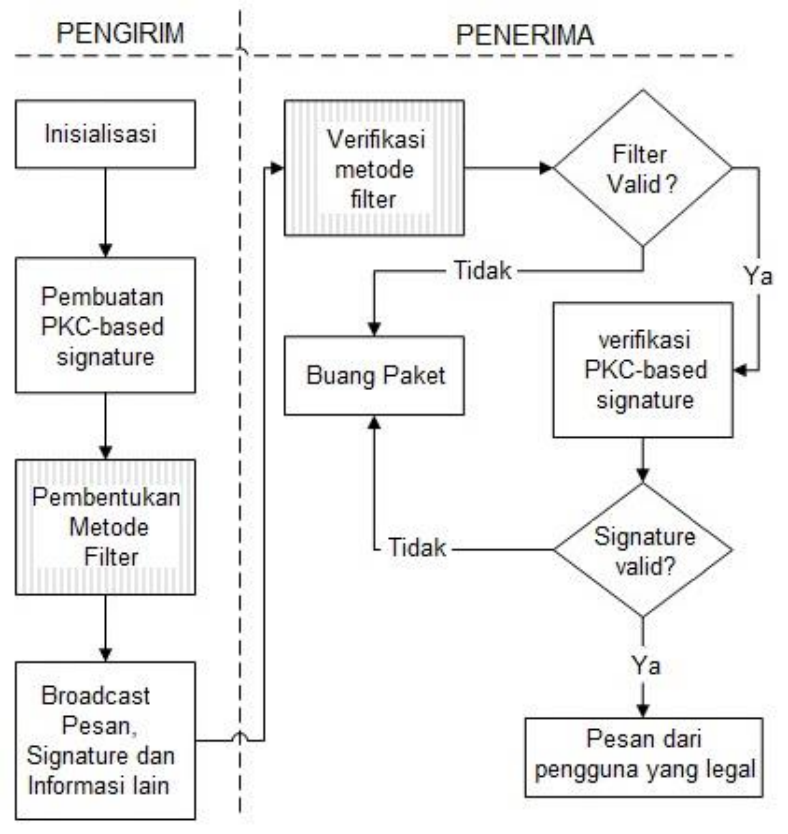
2.3. Metode filter pada JSN

Komunikasi *broadcast* sangat rentan terhadap serangan yang membanjiri dengan paket palsu. Oleh karena hal itu, penggunaannya harus didampingi dengan autentikasi untuk memastikan identitas dari pengirim. Penelitian ini fokus pada autentikasi *broadcast* berkunci asimetris dalam rangka menghadapi serangan DoS berbasis PKC. Meskipun autentikasi menggunakan kunci asimetris memiliki komputasi yang cukup tinggi, akan tetapi tingkat keamanan terjamin dan terbukti dapat diimplementasikan pada perangkat keras berdaya rendah seperti node sensor.

Dari berbagai pilihan PKC, penelitian ini memanfaatkan Elliptic Curve Digital Signature Algorithm (ECDSA) (Johnson et al. 2001) sebagai mekanisme pembangkit signature. Algoritma ini cukup dikenal karena keefektifannya. Hal ini disebabkan, dengan tingkat keamanan yang sama dengan algoritma digital signature yang lain, ukuran kunci ECDSA lebih kecil. Berikut beberapa alasan mengapa ECDSA digunakan :

1. ECDSA diketahui sebagai algoritma signature yang paling aman dan efisien untuk perangkat berdaya terbatas (Hyun & Ning 2008) .
2. Seluge sebagai protocol untuk diseminasi kode pada MICAz-Tiny OS menggunakan ECDSA sebagai digital signature (Hyun & Ning 2008).
3. Dari 7 penelitian yang membahas permasalahan DoS pada digital signature, terdapat 5 penelitian yang menggunakan ECDSA (Ning & Liu 2008; Du & Chen 2008; Chuchaisri & Newman 2012; Kim et al. 2016) sedangkan 2 sisanya tidak spesifik terhadap algoritma tertentu (independen)

Detail dari implementasi ECDSA dipaparkan pada Lampiran D. Verfikasi dari ECDSA membutuhkan sumber daya yang tinggi khususnya pada proses multiplikasi bit. Hal ini dimanfaatkan oleh penyerang untuk mengirimkan *signature* palsu dalam jumlah besar. Metode filter atau bisa juga disebut *pre-authentication* merupakan proses tambahan yang rendah komputasi dan bertujuan untuk mencegah terjadinya serangan DoS berbasis PKC. Proses ini diperlukan



Gambar 2.6 Blok diagram penggunaan metode filter

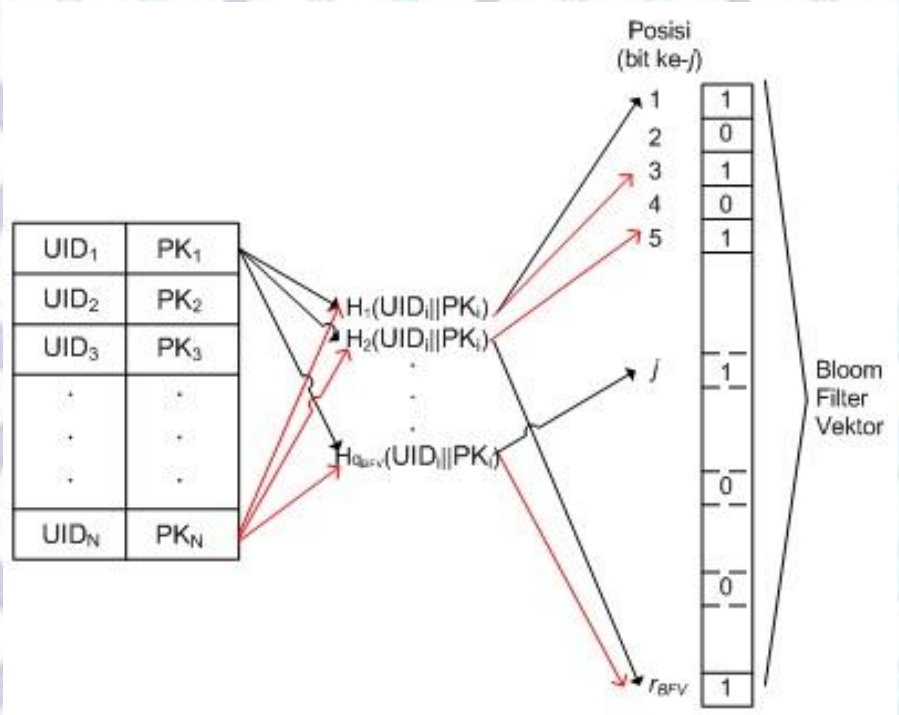
sebagai saringan awal agar tugas node sensor tidak berat. Metode filtering diilustrasikan pada Gambar 2.6.

Berdasarkan mekanisme dan penggunaan algoritma, metode filter dikelompokkan menjadi dua bagian yaitu independen dan hybrid. Independen dimaksudkan bahwa metode filter yang digunakan bersifat berdiri sendiri dan tidak digabungkan dengan metode lainnya dalam pemanfaatannya. Pada pengelompokan tersebut, terdapat dua algoritma yang dapat dijadikan contoh yaitu *Bloom filter* dan *Randomly drop*. Metode ini memiliki kelemahan dengan masih adanya kemungkinan pesan palsu yang tidak terjaring dikarenakan baik dikarenakan proses random maupun kesamaan nilai hash. Masing-masing metode filter tersebut akan dibahas secara detail pada Bab 2.3.1 dan 2.3.2.

Metode filter dengan mekanisme hybrid bersifat dapat digabungkan dengan metode atau konsep lainnya sehingga diharapkan bisa menyaring berbagai macam pesan palsu dengan probabilitas lebih tinggi daripada mekanisme independen. Contoh algoritma pada mekanisme ini yaitu *grouping* dan skema puzzle dan *key chain* yang dibahas secara detail pada Bab 2.3.3, 2.3.4 dan 2.3.5, secara berurutan.

2.3.1. Bloom Filter

Bloom filter merupakan metode yang cukup popular dan digunakan untuk verifikasi keanggotaan. Metode ini dikenalkan oleh Burton Bloom di tahun 1970 sebagai struktur data yang sederhana, efisien dan probabilistik untuk merepresentasikan kelompok dalam mendukung pencarian anggota (Broder & Mitzenmacher 2004) Teknik ini dibangun dengan sebuah vektor berukuran r_{BFV} bit yang selanjutnya disebut Bloom Filter Vector (BFV) dan berisi informasi lojik benar (anggota) atau salah (bukan). Anggota direpresentasikan dengan bit 1 sedangkan bukan anggota dengan bit 0. Ilustrasi dari mekanisme bloom filter dapat dilihat pada Gambar 2.7. Garis petunjuk berwarna hitam merupakan proses penempatan pasangan UID_1 dan PK_1 sedangkan garis petunjuk berwarna merah



Gambar 2.7 Mekanisme bloom filter

merupakan proses penempatan pasangan UID_N dan PK_N pada BFV.

Proses pembangunan BFV diawali dengan pendefinisian ruang penyimpanan menggunakan vektor berdimensi 1 dengan ukuran $1 \times r_{BFV}$ yang berisi nilai 0. Di lain sisi, operasi hash dijalankan pada pasangan identitas node sensor dengan kunci publik. Operasi tersebut dijalankan sebanyak q_{BFV} kali dengan fungsi hash yang berbeda-beda. Luaran dari operasi hash tersebut dipetakan pada index yang merupakan posisi dari bit ke-j pada r_{BFV}. Pemetaan ini bisa memanfaatkan ukuran r_{BFV} dan fungsi modulo. Posisi yang didapatkan dari hasil pemetaan tersebut diubah nilainya menjadi 1. Proses ini dijalankan hingga semua identitas node sensor dan kunci publik dipetakan pada BFV. Dipilihnya teknik BFV sebagai struktur data dikarenakan efisien dalam komputasi dan ruang penyimpanan. Namun, hal tersebut diikuti dengan peluang false positif yang masih mungkin terjadi dan dirumuskan pada

$$FPR = \left(1 - e^{-q_{BFV}N/r_{BFV}}\right)^{q_{BFV}}, \tag{2.1}$$

dengan FPR adalah peluang false positif. Semakin banyak operasi hash yang digunakan, maka semakin lama dalam membangun BFV dan semakin cepat penuh isi vektornya menjadi bernilai 1. Sebaliknya, jika jumlah operasi hash terlalu sedikit maka peluang false positif semakin besar. Oleh karena itu jumlah optimum dari banyaknya fungsi hash dirumuskan sebagai (Tarkoma et al. 2012)

$$q_{BFV,opt} = \frac{r_{BFV}}{\mathrm{N}} \ln 2. \tag{2.2}$$

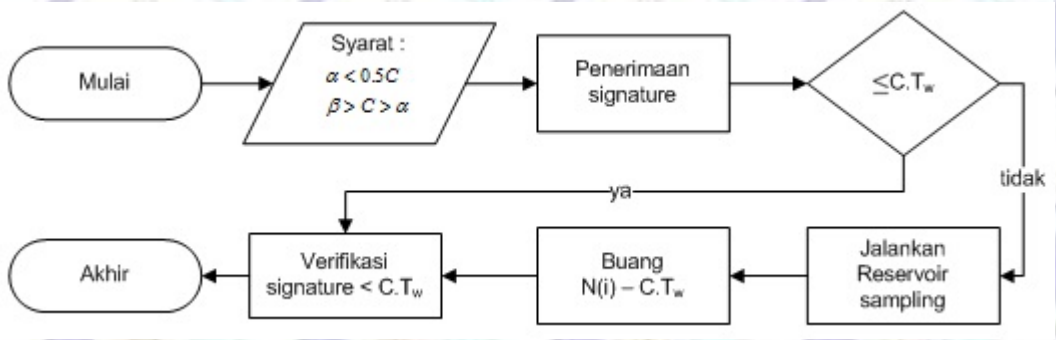
False negatif tidak mungkin dialami sehingga kondisi ini dapat diartikan bahwa paket palsu masih dapat diterima oleh sistem. Ukuran dari BFV didefinisikan pada fase inisialisasi dan harus memenuhi persyaratan berikut (Tarkoma et al. 2012)

$$r_{BFV} = \frac{-N \,.\ln(FPR)}{0.4805}. \tag{2.3}$$

Semakin tinggi jumlah anggota maka semakin besar kemungkinan paket palsu yang diterima atau probabilitas false positif semakin tinggi (Chuchaisri & Newman 2012).

2.3.2. Randomly drop

Metode randomly drop akan mengefisienkan paket yang diterima oleh sistem dengan cara membuangnya secara acak sesuai dengan kemampuan atau energi yang dimiliki oleh node sensor. Tujuan utama dari metode ini adalah penghematan energi sehingga usia node sensor dapat diperpanjang. Mekanisme ini hanya bisa dijalankan pada kondisi yaitu signature resmi yang diterima separuh dari jumlah signature yang bisa dijalankan pada node sensor. Akan tetapi jumlah signature yang tidak resmi bisa berjumlah lebih besar dari kemampuan yang dimiliki node sensor. Gambaran umum proses yang dijalankan pada skema ini ditunjukkan pada Gambar 2 8, dengan α, β, C, T_w dan $N(i)$ adalah total PKC valid yang dikirim oleh node sensor tetangga, total PKC invalid yang dikirim oleh peretas, kemampuan node sensor untuk memproses PKC, interval waktu pada time window W dan jumlah signature yang diterima node sensor pada saat T_w.



Gambar 2 8 Skema randomly drop

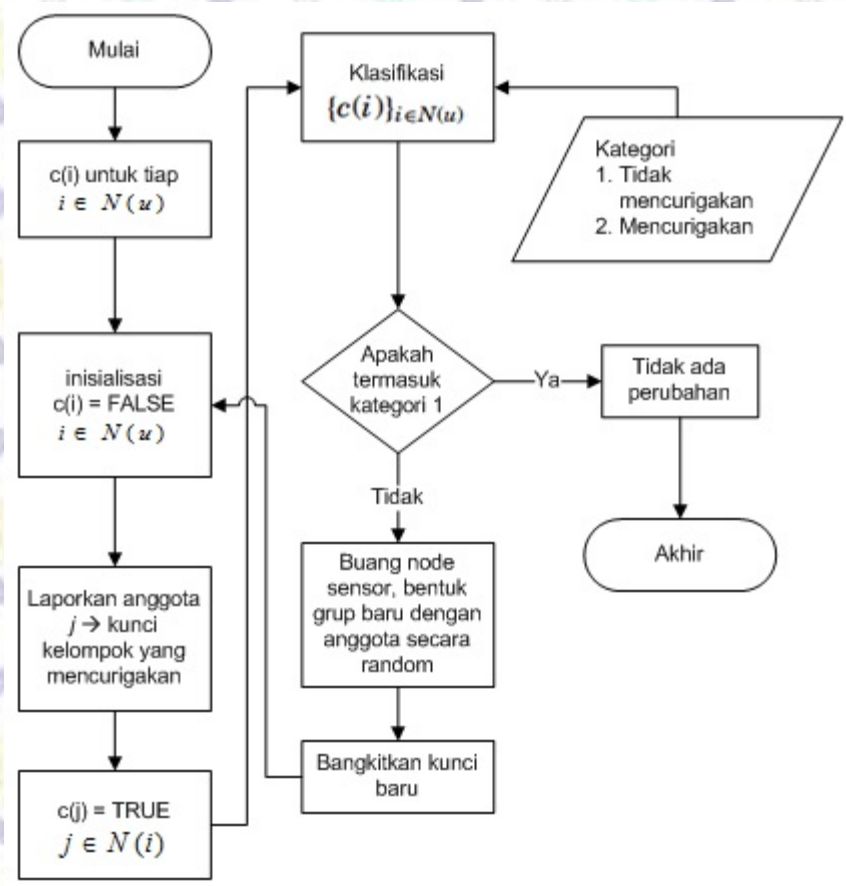
Pada saat signature diterima pada skema randomly drop (Kim & An 2016), pengecekan apakah verifikasi signature yang datang lebih dari kemampuan node sensor akan dijalankan. Jika ya, maka penerima harus mengimplementasikan proses pembuangan secara random menggunakan teknik *reservoir sampling* tertentu. Metode tersebut akan memilih sejumlah sample dari signature yang diterima. Kelebihan teknik *reservoir sampling* yaitu efisien dalam penggunaan ruang penyimpanan. Sedangkan, kelemahan metode ini adalah pesan asli dari pengirim terautentikasi harus kurang dari setengah energi yang dimiliki oleh node sensor. Selain itu juga masih terdapat peluang terjadinya false positif dan negatif karena paket yang dibuang bersifat random dan sangat bergantung pada algoritma yang digunakan untuk mengacaknya.

2.3.3. Grouping

Pada metode ini, penerima dibagi menjadi beberapa kelompok dengan jumlah anggota sama besar. Masing-masing kelompok memiliki kunci kelompok yang unik antara satu kelompok dengan kelompok lainnya. Pada (Dong et al. 2013), digunakan *q*-bit dari nilai komitmen W_{group} sebagai tambahan signature yang ikut serta dikirim bersamaan dengan pesan. Nilai tersebut didapatkan dari sejumlah *m* kelompok di JSN dengan

$$W_{group} = \frac{q}{m} \text{ bit pertama dari } H(K_i||M). \quad (2.4)$$

Nilai komitmen berisi hasil fungsi hash dari penggabungan antara kunci tiap kelompok dan pesan.



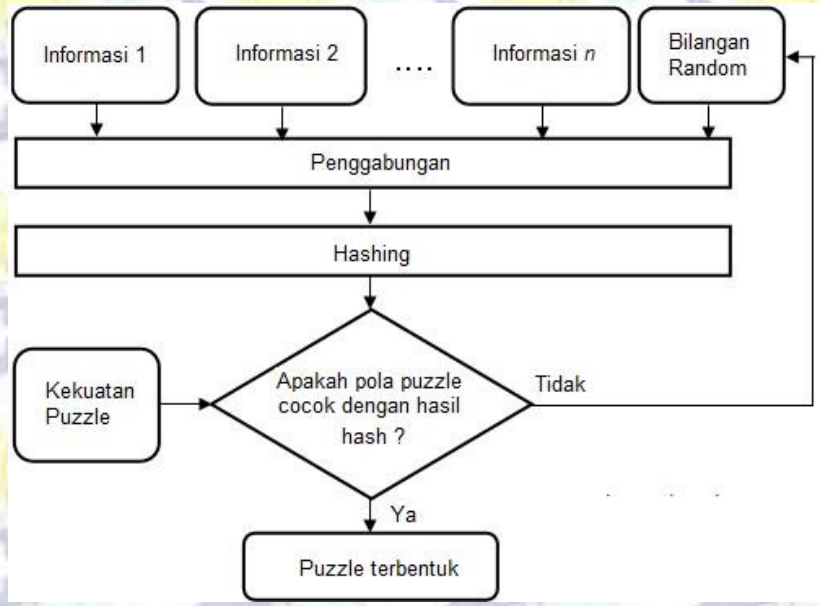
Gambar 2.9 Skema re-grouping adaptif

Keamanan pada metode grouping sangat terjaga, karena penyerang harus mengetahui semua kunci dari setiap kelompok yang ada di JSN. Metode ini memiliki kekurangan dalam meningkatnya ukuran paket yang dipertukarkan atau tingginya biaya komunikasi antar pengirim dan penerima. Selain itu kedinamisan dalam penambahan dan pengurangan anggota menambah beban jaringan. Pengurangan anggota diperlukan jika terdapat kelompok yang ditengarai tidak aman. Terdapat pengembangan metode untuk mengatasi adanya penyerang pada sebuah kelompok dengan mekanisme seperti terlihat pada Gambar 2.9 (Dong et al. 2013). Skema re-grouping adaptif menawarkan kedinamisan dalam mendeteksi adanya penyerang di dalam sebuah kelompok. Jika pengirim menerima laporan adanya kunci kelompok *j* yang tidak aman maka boolean variabel *c(j)* akan bernilai TRUE dan dilanjutkan dengan proses analisa untuk klasifikasi apakah benar terdapat penyerang atau tidak. Klasifikasi tersebut didasarkan pada

frekuensi laporan akan kegagalan node sensor tersebut dalam memverifikasi pesan yang datang. Jika jumlah kegagalan verifikasi melebihi threshold yang ditentukan, maka node sensor tersebut mencurigakan dan masuk pada kelompok 2 yang selanjutnya akan dibuang. Karena telah terjadi kebocoran informasi akan nilai kunci kelompok maka perlu adanya kelompok baru dengan pembangkitan kunci kelompok dan anggota yang baru.

2.3.4. Puzzle

Skema ini bersifat independen dapat dimanfaatkan terpisah maupun diintegrasikan dengan metode lain. Secara umum mekanisme dari skema puzzle terlihat pada Gambar 2.10. Informasi akan dikirimkan oleh pengguna digabungkan dengan bilangan random yang selanjutnya disebut solusi puzzle. Proses hash dari penggabungan tersebut akan disesuaikan dengan pola tertentu. Jika *l* bit pertama cocok maka solusi puzzle ditemukan, sebaliknya proses harus diulang dari awal. Pada setiap percobaan untuk pencarian solusi puzzle, hanya terdapat dua kemungkinan yang mungkin terjadi yaitu sukses (puzzle ditemukan) atau gagal (kandidat puzzle tidak sesuai dengan pola). Oleh karena itu, ketidakpastian dari pencarian puzzle bisa didekati dengan Bernoulli trial (Menezes et al. 1996). Jika peluang ditemukannya solusi dinotasikan dengan *p*, banyaknya



Gambar 2.10 Skema puzzle

kesuksesan dalam pencarian solusi dinotasikan dengan *k* pada jumlah percobaan sebanyak *n* kali, maka bernoulli trial untuk kejadian *X* dirumuskan dengan

$$P(X = k) = \binom{n}{k}.p^k.(1-p)^{n-k}, \tag{2.5}$$

dengan *X* diartikan sebagai kejadian ditemukannya solusi. Oleh karena itu, Persamaan (2.5) diartikan sebagai peluang ditemukannya solusi sebanyak *k* pada *n* percobaan. Proses menemukan solusi puzzle penuh dengan ketidakpastian akan tetapi dapat dihitung peluang ditemukannya minimal satu solusi pada *n* kali percobaan dapat dirumuskan pada penurunan rumus berikut

$$\begin{aligned} P(X \geq 1) &= 1 - P(X = 0) \\ &= 1 - nC_0.\ p^0.(1-p)^{n-0} \\ &= 1 - 1.1.(1-p)^n \\ &= 1 - (1-p)^n \\ &= 1 - \left(1 - \left(\frac{1}{2}\right)^l\right)^n. \end{aligned} \tag{2.6}$$

Peluang sukses dari skema puzzle sendiri dirumuakan sebagai $p = (1/2)^l$. Rata-rata iterasi hash yang dijalankan pada setiap kekuatan puzzle *l* adalah 2^l (Du & Chen 2008). Tujuan dari penelitian ini adalah menurunkan delay akan tetapi tetap mempertahankan keamanan dengan menggunakan fungsi ambang pada daerah interkuartil bukan rata-rata.

2.3.4.1. Client Puzzle

Client puzzle dikembangkan dengan tujuan menghadapi serangan DoS yang bisa berakibat fatal pada sumber daya server (Aura et al. 2000). Penggunaan kunci publik saja tidak cukup, dikarenakan pemanfaatannya membutuhkan sumber daya baik komputasi maupun ruang penyimpanan yang mahal. Penggunaan puzzle mengawali proses autentikasi sebelum pengirim memanfaatkan sumber daya yang lebih besar. Pada pembuatan puzzle, server mambangkitkan bilangan random N_S secara periodik yang dikirim ke client. Selain itu pada tahap inisialisasi server menentukan tingkat keamanan *l* yang juga dikirim bersamaan dengan bilangan random *Ns.* Dengan diterimanya kedua parameter tersebut, client bisa langsung berkomunikasi dengan server melalui

proses pencarian solusi puzzle terlebih dahulu dengan rumus

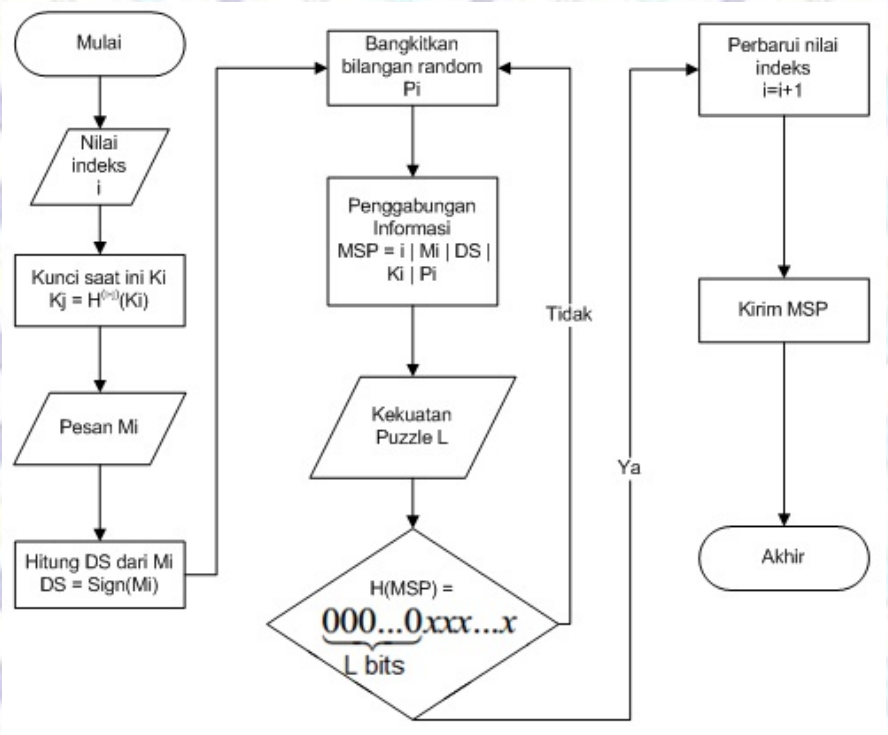
$$H(Uid, Ns, Nc, Pidx) = \underbrace{000 \ldots 000}_{l \text{ bit pertama dari fungsi hash}} \overbrace{XXX}^{\text{sisa bit dari fungsi hash}}, \quad (2.7)$$

dengan *Nc* adalah bilangan random yang meningkatkan kompleksitas serangan dan informasinya hanya dimiliki client dengan panjang 24 bit. Pihak server harus menyimpan informasi *Nc* dan *Pidx* yang telah terverifikasi dalam rangka mengindari penggunaan kembali puzzle yang telah digunakan. Komputasi pencarian solusi puzzle pada penerima dibatasi dengan waktu tertentu dengan batas waktu ini pulalah pihak server mengirimkan kembali nilai *Ns*. Meskipun dibatasi oleh waktu, metode ini memiliki kelemahan dalam hal banyaknya interaksi client-server yang dibutuhkan sebelum proses pembuatan puzzle (Ning & Liu 2008), sedangkan komunikasi antara pengguna dengan node sensor yang berjumlah banyak tidak memungkinkan interaksi tersebut. Selain itu tidak adanya kunci sesi membuat sistem lemah terhadap modifikasi pesan pada proses transmisi.

2.3.4.2. Message Specific Puzzle

Tahun 2008, (Ning & Liu 2008) mengembangkan *Message Specific Puzzle* (MSP) yang digunakan sebagai metode filter dengan memanfaatkan key chain arah *backward* sebagai kunci sesi. MSP hanya membutuhkan satu operasi fungsi hash pada saat verifikasi. Sedangkan pembuatan solusi puzzle di sisi penerima membutuhkan sejumlah fungsi hash yang jumlahnya tidak dapat ditentukan. Proses tersebut diawali dengan pembangkitan kunci menggunakan metode key chain. Nilai komitmen yang didapat harus dikirimkan pada penerima melalui jalur yang aman. Setelah mendapat kunci terbaru dan proses pembangkitan signature utama, pengirim mulai dapat membangkitkan bilangan random dalam rangka mencoba mencari apakah solusi ditemukan. Solusi dari MSP merupakan hasil dari hashing atas nilai indeks, pesan, signature dan bilangan random yang telah digabungkan sebelumnya seperti dirumuskan pada

$$H(idx, M, Sig, Kidx, Pidx) = \underbrace{000 \ldots 000}_{l \text{ bit pertama dari fungsi hash}} \overbrace{XXX}^{\text{sisa bit dari fungsi hash}}. \quad (2.8)$$



Gambar 2.11 Skema MSP

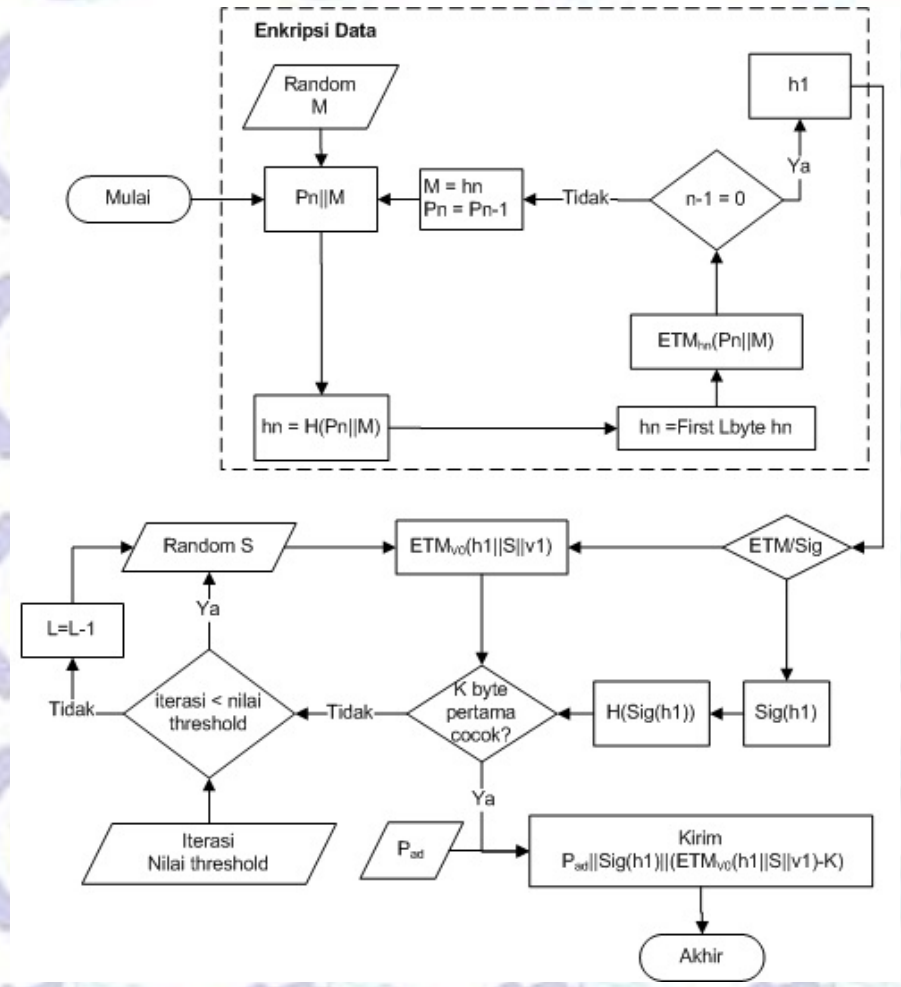
Signature berfungsi sebagai autentikator broadcast yang dibangkitkan oleh pengguna. Solusi ditemukan dan dapat dikirimkan jika memenuhi pola nilai nol yang berurutan sejumlah *l* bit. Nilai tersebut disebut dengan kekuatan puzzle. Semakin tinggi nilai *l* maka semakin tinggi tingkat keamanan dari MSP. Akan tetapi hal tersebut diikuti dengan semakin banyaknya iterasi fungsi hash yang dibutuhkan untuk mencoba pencarian solusi. Flowchart dari skema MSP dapat dilihat pada Gambar 2.11.

Skema puzzle ini cukup efisien dan mudah diimplementasikan serta memiliki komputasi yang rendah khususnya pada sisi penerima. Akan tetapi kekurangan terbesarnya adalah delay pada sisi pengirim yang tinggi dan tidak dapat diprediksi.

2.3.4.3. Cipher Puzzle

Pemanfaatan key chain tunggal berarah *backward* lain yang diintegrasikan dengan puzzle yaitu cipher puzzle (Tan et al. 2013). Skema ini merupakan pengembangan dari MSP. Cipher puzzle menambahkan karakteristik

keamanan *confidential* dengan cara mengenkripsi pesan yang dikirimkan. Hal ini dikarenakan, skema ini digunakan pada kasus diseminasi kode sehingga data yang ditransmisikan bersifat penting dan rahasia. Selain itu cipher puzzle digunakan secara spesifik pada protocol Deluge (Hui & Culler 2004). Program image dipecah menjadi beberapa bagian. Setiap potongan yang akan dikirimkan harus melalui metode puzzle. Berbeda dengan MSP yang hanya memanfaatkan fungsi hash, cipher puzzle menggunakan pengamanan berlipat yaitu dengan metode *Encryption Then MAC* (ETM). Metode ini dimanfaatkan pada data gabungan antara pecahan kode program, bilangan random dan kunci sesi terbaru. Hasil dari ETM akan dibandingkan dengan hasil dari fungsi hash pada *signature* kode program jika memenuhi kesamaan bit sebanyak *l* maka solusi ditemukan.



Gambar 2.12 Skema cipher puzzle

Proses pembuatan puzzle ini dirumuskan pada

$$ETM_{v0}(h1|S|v1) = H(Sig(h1)) , \quad (2.9)$$

dengan *v0* merupakan kunci sesi pada indeks ke-0 sedangkan *v1* merupakan kunci sesi pada indeks ke-1 dan *S* adalah solusi puzzle. Panjang kesamaan bit pada Persamaan (2.9) inilah yang merupakan kekuatan puzzle pada skema cipher puzzle. Detail pembuatan skema ini pada sisi penerima diilustrasikan pada Gambar 2.12. Keamanan pada skema ini cukup tinngi, bahkan penulis mengklaim sebagai peneliti pertama yang menjamin parameter keamanan autentikasi dan *confidentiality* pada JSN multihop. Kekurangan utama pada MSP masih juga dialami oleh cipher puzzle. Semakin tinggi kekuatan puzzle, maka semakin tinggi delay pada sisi penerima. Satuan kekuatan puzzle pada cipher puzzle adalah byte sehingga kedinamisannya rendah. Hal tersebut dikarenakan rata-rata waktu yang dibutuhkan untuk mengirim pesan dengan kekuatan tertentu dirumuskan sebagai 2^l. Jika *l* bernilai 3, maka rata-rata dibutuhkan jumlah hash sebanyak 2^{24}. Nilai tersebut cukup tinggi dan bisa digunakan sebagai maksimum kekuatan puzzle sehingga pilihan penggunaan parameter tersebut sangat tidak variatif.

2.3.5. Key chain

Keychain singkatan dari one-way key chain banyak digunakan pada berbagai skenario untuk autentikasi yang efisien (Ning & Liu 2008). Komputasinya cukup rendah, hanya membutuhkan satu fungsi hash yang dijalankan sebanyak jumlah kunci yang ingin dihasilkan. Langkah pertama yang dijalankan adalah membangkitkan sebuah bilangan random yang dijadikan sebagai kunci pada posisi terakhir pada key chain *backward* dan kunci pada posisi pertama pada key chain *forward*. Kemudian berturut-turut nilai tersebut di hash sejumlah kunci yang dibutuhkan. Secara matematis keseluruhan kunci dengan arah *backward* yang dihasilkan bisa dirumuskan dalam (Ning & Liu 2008).

$$K_{i,backward} = H(K_{i+1}), 0 \leq i \leq n-2 , \quad (2.10)$$

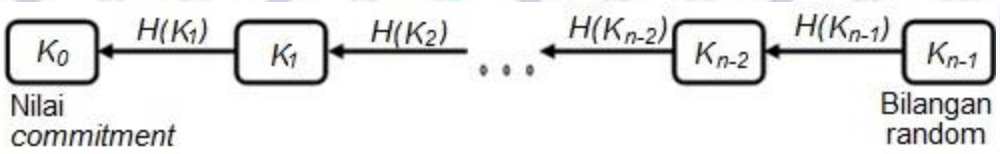
sedangkan kunci yang dihasilkan oleh key chain arah *forward* dirumuskan pada

$$K_{i,forward} = H(K_{i-1}), 1 \leq i \leq n-1 \, . \quad (2.11)$$

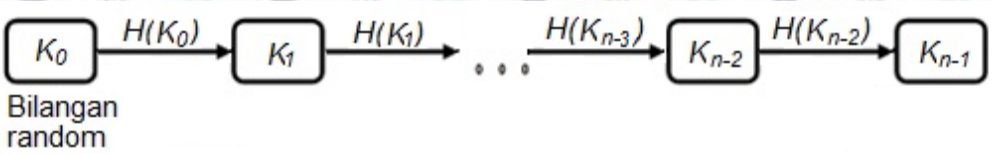
Kunci pada posisi pertama disebut sebagai nilai komitmen. Nilai inilah yang akan dikirimkan ke penerima sebagai dasar untuk verifikasi metode filter. Pembangkitan key chain dengan arah *backward* diilustrasikan pada Gambar 2.13. Sedangkan dengan key chain arah *forward* berlaku dengan arah panah sebaliknya seperti yang ditunjukkan pada Gambar 2.14. Pada implementasinya, pemanfaatan key chain merupakan gabungan dari jumlah key chain dengan arah penggunaan fungsi hash tertentu. Hingga saat ini ada tiga variasi pemanfaatan key chain yang bertujuan untuk mitigasi serangan DoS pada JSN yaitu tunggal dengan arah *backward* (Ning & Liu 2008; Dong et al. 2013) dan *forward* (Chuchaisri & Newman 2012) juga jamak dengan arah *backward* (Du & Chen 2008). Kelebihan dan kekurangan arah key chain akan dibahas pada Tabel 2.4. Masing-masing metoda memiliki kelebihan dan kekurangan. Penilitian ini fokus pada key chain dengan arah *backward*.

Tabel 2.4 Perbandingan antar key chain dengan arah yang berbeda

No	Keterangan	Arah penggunaan fungsi hash	
		Backward	*Forward*
1	Pemanfaatan bilangan random	Sebagai kunci paling akhir (K_{n-1})	Sebagai kunci pertama (K_0)
2	Pendistribusian kunci	Indeks dan kunci berikutnya turut dikirimkan pada pesan	Hanya perlu mengirimkan indeks pada pesan
3	Jumlah kunci yang tersedia	Terbatas dan ditetapkan pada tahap inisialisasi	Tak terbatas
4	Pembangkitan ulang key chain	Setelah mencapai indeks akhir	Tidak diperlukan selama key chain tidak terkuak (aman)
5	Kunci yang dikirim diketahui oleh penyerang	Key chain tetap aman karena kunci selanjutnya tidak dapat diketahui penyerang	Keseluruhan kunci pada key chain aktif akan diketahui oleh penyerang
6	Sumber daya pengirim	Membangkitkan dan menyimpan seluruh nilai pada key chain aktif	Hanya perlu menyimpan kunci pertama dan indeks terbaru yang digunakan



Gambar 2.13 Key chain arah *backward*



Gambar 2.14 Key chain arah *forward*

Tipe forward key chain lebih hemat dalam hal komunikasi, karena kunci tidak perlu dikirimkan. Pengguna hanya perlu mengirimkan indeks dari kunci. Selain itu, jumlah kunci yang tersedia berjumlah tak terbatas. Akan tetapi, tipe tersebut memiliki keterbatasan dalam hal keamanan. Jika node sensor diambil alih oleh penyerang (*node compromised*) maka key chain tersebut sudah tidak dapat digunakan kembali. Pengguna harus membangkitkan bilangan random baru dan mengirimkannya kepada seluruh node sensor sebagai pertanda bahwa pengguna telah membentuk key chain yang baru.

Sedangkan untuk tipe backward, tingkat keamanannya lebih terjamin. Hal tersebut dikarenakan node sensor memiliki pertahanan jika node sensor tetangganya diambil alih. Hal tersebut tidak mempengaruhi key chain dari pengirim. Penyerang tidak akan bisa mengetahui kunci yang selanjutnya meskipun kunci yang tersimpan di dalam node sensor diketahui. Alasannya dikarenakan sifat hash yang *irreversible*. Keterbatasan pada tipe ini yaitu jumlah kunci yang terbatas karena jumlah kunci yang tersedia harus didefinisikan pada awal pembentukannya dan meningkatnya ukuran paket yang ditransmisikan. Lebih lanjut, biaya komunikasi bertambah karena diperlukan ruang untuk mengirim kunci yang akan digunakan untuk verifikasi. Penelitian saat ini lebih banyak menggunakan tipe *backward* karena keamanannya, sedangkan tipe forward dijumpai pada (Chuchaisri & Newman 2012) dengan memanfaatkan multi key chain sehingga bisa digunakan untuk multi-user dan (Tan et al. 2013) yang memanfaatkan single key chain. Pemanfaatan tipe forward

diimplementasikan dengan alasan keterbatasan ukuran paket, sedangkan pada skema tersebut (Chuchaisri & Newman 2012) diharuskan mengirim BFV untuk setiap pertukaran pesan.

2.4. JSN Multiuser

Penggunaan JSN saat ini telah berkembang dan diharapkan mampu mengolah, menyimpan dan menyediakan data hasil penginderaan langsung kepada pengguna sesuai dengan permintaan atau kebutuhan (Ren et al. 2009). Hal tersebut sangat penting khususnya pada aplikasi berdasarkan query. Pengguna jaringan bisa seorang administrator yang mengatur jaringan dari jarak jauh ataupun beragam aplikasi yang memanfaatkan data yang diperoleh dari penginderaan sensor. Skema multi-user pada kriptografi PKC yang tengah berkembang yaitu penggunaan sertifikat, ruang penyimpanan penerima, bloom filter dan hybrid yang akan dibahas mendetail pada Sub Bab berikut. Pada semua skema tersebut node sensor tidak dilengkapi dengan pasangan kunci publik dan privat. Sebaliknya yang memilikinya hanya pengguna dan base station.

2.4.1. Certificate based Authentication Scheme (CAS)

Pada skema ini, base station bertindak sebagai otoritas sertifikasi dan node sensor hanya perlu menyimpan informasi kunci publik dari base station. Dengan kata lain, base station meneruskan informasi tambahan pada paket yang diterima dari pengguna JSN yang beragam. Informasi tambahan tersebut dirumuskan pada (Ren et al. 2009)

$$Cert_{Uid} = Uid|\, PK_{Uid}|ExpT|Sig_{SK_{BS}}\{H(U_{id}|ExpT|PK_{Uid})\}\,, \qquad (2.12)$$

dengan *ExpT* adalah waktu kadaluwarsa dari sertifikat yang dibuat, SK_{BS} adalah kunci privat dari base station. Pesan broadcast yang akan diterima oleh node sensor secara keseluruhan ditunjukkan pada

$$\langle M|\, tt \mid Sig_{SK_{Uid}}\{H(Uid|tt|M)\} \mid Cert_{Uid}\rangle\,. \qquad (2.13)$$

Node sensor harus menjalankan dua kali proses verifikasi. Pertama memverifikasi sertifikat pengguna yang merupakan informasi tambahan dari base station. Kedua, dari verifikasi pertama node sensor mendapatkan informasi mengenai kunci publik dari pengguna yang bisa digunakan untuk memverifikasi signature dari pesan yang diterima.

Kelemahan penggunaan CAS adalah tidak efisien karena sertifikat harus dikirim bersamaan dengan pesan hingga membebani jaringan terutama pada JSN multihop. Redundansi yang terjadi akan menghabiskan bandwidth yang tersedia. Selain itu, untuk memverifikasi pesan dibutuhkan minimal dua kali operasi verifikasi signature. Seperti diketahui, kegiatan tersebut rentan sekali dengan serangan DoS berbasis PKC.

2.4.2. Direct-storage based Authentication Scheme (DAS)

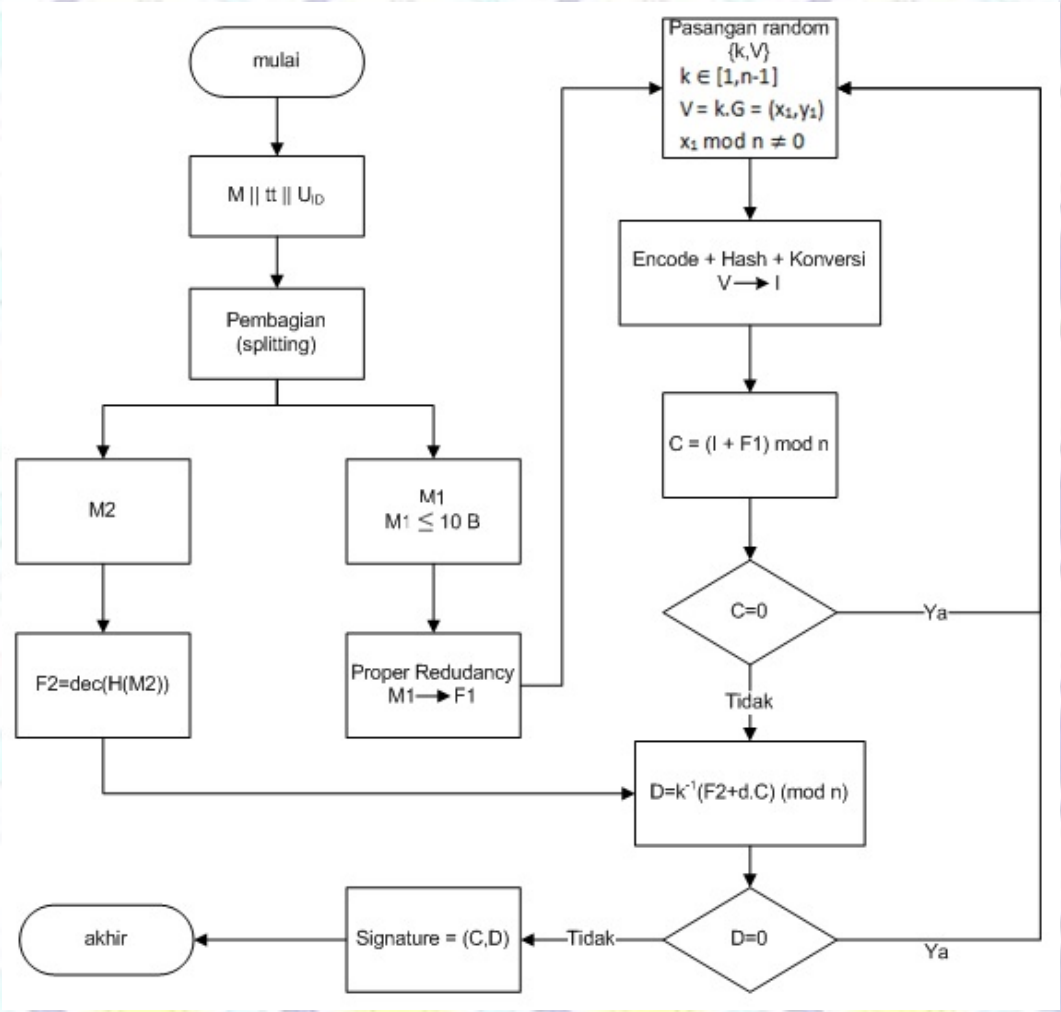
Dalam rangka mengurangi beban komunikasi yang dialami oleh CAS, metode DAS menyimpan semua public key beserta ID dari setiap pengguna yang akan mengakses jaringan sensor di sisi penerima. Dengan demikian pesan yang diterima dan dipertukarkan pada jaringan sensor menjadi (Ren et al. 2009)

$$\langle M | tt | Sig_{SK_{Uid}}\{H(Uid|tt|M)\} | Uid | PK_{Uid}\rangle . \quad (2.14)$$

Pendekatan ini sederhana, rendah komputasi dan tidak membebani jaringan dengan mengirimkan sertifikasi pengguna. Akan tetapi masalah timbul pada ruang penyimpanan pada node sensor yang akan membengkak seiring dengan perkembangan jumlah pengguna. Pada node sensor berkapasitas 5 kB maksimal jumlah pengguna yang dapat disimpan sebesar 232, sedangkan untuk 19.5 kB metode ini dapat manampung hingga 1000 pengguna. Padahal di saat yang sama, metode CAS bisa mendukung 2560 pengguna secara bersamaan (Ren et al. 2009).

2.4.3. Bloom filter based authentication scheme (BAS)

CAS dan DAS merupakan metode dasar dengan pendekatan sederhana yang masih memiliki banyak kekurangan jika diimplementasikan pada dunia nyata. BAS merupakan perbaikan skema DAS dengan tidak menyimpan informasi



Gambar 2.15 Skema pembentukan signature pada BAS

pengguna secara keseluruhan (ID dan kunci publik) akan tetapi hanya hashing dari pemetaan antara keduanya menggunakan metode Bloom Filter. Secara garis besar metode BAS sama seperti varian ECDSA yang dimanfaatkan untuk multi-user (Ren et al. 2009).

Pada tahap inisialisasi, base station membangkitkan pasangan ID dan kunci publik sejumlah *N*. Jika panjang PK_{Uid} adalah l byte dan Uid 2 byte, maka ukuran BFV harus memenuhi (Gan & Li 2009)

$$q_{BFV}N < r_{BFV} < N(l+2). \tag{2.15}$$

Tujuan dari nilai r_{BFV} tersebut adalah untuk menurunkan ukuran filter dan menurunkan peluang false positif pada BFV.

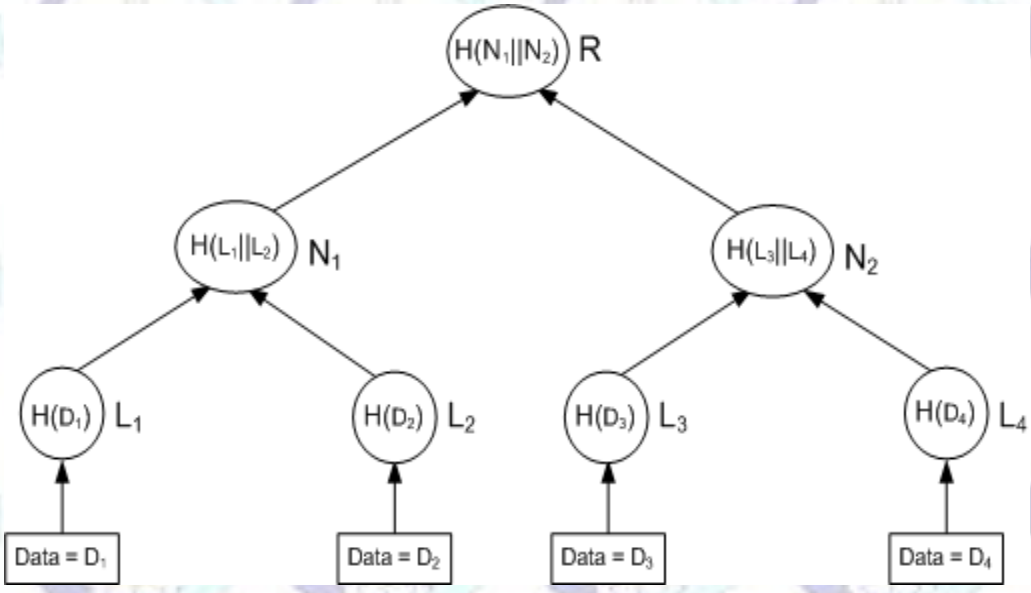
Metode tambahan yang digunakan untuk menurunkan peluang kesalahan statistik adalah Counting Bloom Filter (CBF). Metode ini digunakan untuk membantu dalam proses penambahan dan pengurangan jumlah pengguna. Akan tetapi CBF memerlukan tambahan ruang penyimpanan yang bertindak sebagai counter yang nilainya akan bertambah 1 jika terdapat penambahan anggota dan akan berkurang 1 jika terdapat anggota yang keluar atau dihapus. Ukuran counter yang optimum adalah 4 bit untuk setiap vektor BFV (Ren et al. 2009) sehingga ukuran CBF 4 kali lebih besar dari BFV. Setelah BFV atau CBF terbentuk, pembentukan signature dijalankan dengan menjalankan variansi dari ECDSA seperti tampak pada Gambar 2.15. Perbedaan antara varian ECDSA yang dijalankan BAS yaitu pembagian pesan menjadi dua bagian dan nilai *C* yang merupakan hasil konversi, hash juga encode dari perkalian antara variabel random dengan generator.

Kekurangan dari DAS dapat ditutupi dengan rendahnya ruang penyimpanan yang dibutuhkan untuk menyimpan informasi tersebut. Akan tetapi hal ini, tidak menutupi kekurangan Bloom Filter akan false positif. Hal tersebut memungkinkan adanya pesan palsu untuk dapat diterima sehingga bisa mengacaukan JSN. Dengan kata lain peluang false positif pada BAS berbanding terbalik dengan jumlah pengguna yang dapat diakomodasi oleh JSN jika ruang penyimpanan pada node sensor tetap (Ren et al. 2009).

2.4.4. Hybrid based Authentication Scheme (HAS)

HAS dibangun untuk mengatasi kekurangan BAS dalam menampung jumlah pengguna yang bisa tergabung dalam JSN. Skema ini memanfaatkan *Markle Hash Tree* (MHT) yang dibangun oleh base station. Mekanisme ini berisi informasi pengirim yang dibangun dengan sebuah tree dengan root yang akan menghasilkan nilai yang digunakan untuk verifikasi anggota (Ren et al. 2009).

Posisi daun MHT ditempati oleh operasi hash antara penggabungan ID dan kunci publik dari pengguna dan dinotasikan dengan L_1 hingga L_4 pada contoh kasus Gambar 2.16,. Sedangkan perhitungan nilai dari node internal atau non-daun didapatkan dari operasi hash dan penggabungan nilai dari node dibawahnya



Gambar 2.16 Contoh kasus *Merkle Hash Tree*

(Grover & Lim 2015). Node internal dinotasikan dengan N_1 dan N_2 pada contoh kasus Gambar 2.16. Sedangkan, perhitungan nilai root dari contoh tersebut adalah $R = H(N_1||N_2) = H(H(H(D_1)|H(D_2))|H\big(H(D_3)\big|H(D_4)\big))$, dengan posisi daun yang terdiri dari 4 node dengan data D_1 hingga D_4.

Pada tahap inisialisasi, base station harus menentukan jumlah pengguna yang dapat ditampung berkaitan dengan peluang false positif dan keterbatasan ruang penyimpanan. Dari parameter tersebut, base station mengumpulkan seluruh kunci publik dari pengguna dan membaginya menjadi m kelompok dengan jumlah anggota m_n yang sama besar. Kemudian, MHT dibangun pada tiap kelompok yang terbentuk. Sehingga jumlah root yang terbentuk $h_r^i, i = 1, \dots, |\boldsymbol{S}|$, dengan $\boldsymbol{S}$ adalah jumlah maksimum pengguna yang mampu ditampung oleh bloom filter. Sehingga jumlah pengguna yang bisa ditampung oleh skema HAS menjadi $S \times m \times m_n$.

Dengan adanya skema ini, jumlah pengguna yang tergabung dalam JSN bisa ditingkatkan dan terhindar dari adanya serangan menggunakan paket palsu akan tetapi kelebihan tersebut diikuti dengan meningkatnya biaya komunikasi sebesar 20 byte atau bahkan 40 byte untuk tiap pesannya (Ren et al. 2009).

2.5. Skema keanggotaan TinySet

TinySet merupakan skema keanggotaan berupa struktur data yang merupakan gabungan antara *compact hash table* dan *blocked bloom filter* (Einziger & Friedman 2017). Blocked bloom filter merupakan partisi dari bloom filter yang dibagi menjadi blok-blok dengan ukuran tetap dan bersifat independen antara satu blok dengan blok yang lain (Putze et al. 2007; Qiao et al. 2011). Bloom filter jenis ini memiliki kelebihan dalam keefisienan pemanfaatan ruang penyimpanan, memiliki throughput yang tinggi dan kebutuhan sumber daya yang rendah (Kanizo et al. 2013). Akan tetapi jumlah pengguna yang menempati setiap blok memiliki ketimpangan sehingga akan efisien pada satu blok dan tidak pada blok yang lain. Jika dibandingkan dengan bloom filter skema ini menjadi tidak efisien. Hash table merupakan alternatif lain dari pemanfaatan bloom filter. Metode ini bermula dari ide pemanfaatan *perfect hash function*. Fungi ini bisa menghasilkan nilai hash yang unik dan tanpa *collision* pada setiap anggota. Nilai

Tabel 2.5 Perbandingan TinySet dengan bloom filter

Aspek	TinySet	Bloom filter
Ruang penyimpanan	Ukuran blok array tetap dengan pambagian tiap blok teratur sehingga efisien	Ukuran array tetap dengan posisi keanggotaan yang tidak teratur sehingga pemanfaatannya menjadi kurang efiisien
Kompleksitas algoritma	Selain fungsi hash, dibutuhkan metode menajemen penempatan fingerprint pada block array	Hanya membutuhkan sejumlah *k* operasi fungsi hash
Operasi query	Posisi lebih detail dan unik sehingga false positif lebih rendah	Posisi tidak unik sehingga memungkinkan terjadinya *collision* dan false positif lebih tinggi
Penghapusan anggota	Mendukung dengan beberapa pilihan konfigurasi	Tidak mendukung akan tetapi bisa menambahkan counter yang perlu menambah ruang penyimpanan

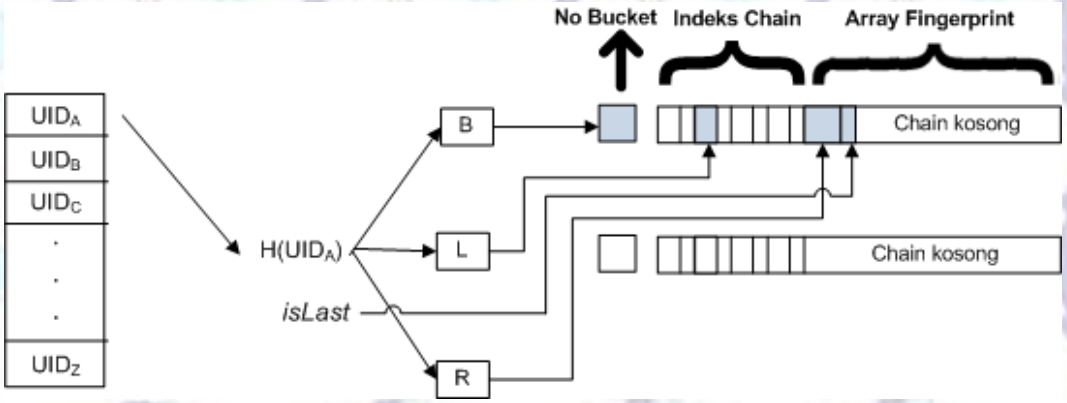
unik ini dinamakan fingerprint dan disimpan ke dalam array yang sederhana (Broder & Mitzenmacher 2004). Perbandingan antara TinySet dengan konsep bloom filter pada beberapa aspek dipaparkan pada Tabel 2.5.

TinySet hanya membutuhkan sebuah fungsi hash yang digunakan diawal iterasi untuk menentukan posisi dari anggota sesuai dengan

$$\mathrm{H} : X \rightarrow B \times L \times R, \tag{2.16}$$

dengan H, *X*, *B*, *L* dan *R* merupakan fungsi hash, nilai masukan, nilai dari bucket, indeks dari chain dan fingerprint yang posisinya pada TinySet diilustrasikan pada Gambar 2.17. Nilai masukan merupakan nilai unik yang melekat pada anggota pada kasus tersebut direpresentasikan dengan identitas pengguna (*UID*).

Pada setiap bucket (*B*), terdapat sejumlah bit indeks chain (*L*) dan pada setiap indeks terdapat sejumlah fingerprint (*R*) yang menandakan nilai unik dari anggota. Indeks chain direpresentasikan dengan deretan bit yang diinisialisasi dengan nilai 0 dan menandakan tidak adanya fingerprint pada indeks chain tersebut. Jika ada fingerprint yang menempati posisi indeks chain tertentu maka nilai indeks itu diberikan nilai 1. Ukuran tiap fingerprint perlu ditambahkan satu bit diakhir posisi yang diberi nama *isLast* untuk menandakan apakah fingerprint tersebut merupakan anggota terakhir pada indeks chain tempat fingerprint itu berada. Nilai *isLast* menjadi 1 jika fingerprint merupakan anggota terakhir pada indeks chain yang ditempatinya sebaliknya akan bernilai 0 jika bukan.



Gambar 2.17 Skema TinySet

TinySet mendukung 3 operasi utama yaitu penambahan, query dan pengurangan pengguna. Memanfaatkan kelebihan bloom filter, query negatif akan yang menunjukkan bukan anggota selalu benar sebaliknya query dengan hasil positif yang menunjukkan anggota memiliki kemungkinan tingkat kesalahan yang disebut false positif. Nilai kesalahan ini bergantung pada ukuran fingerprint yang ditetapkan pada tahap inisialisasi dan dirumuskan pada (Einziger & Friedman 2017)

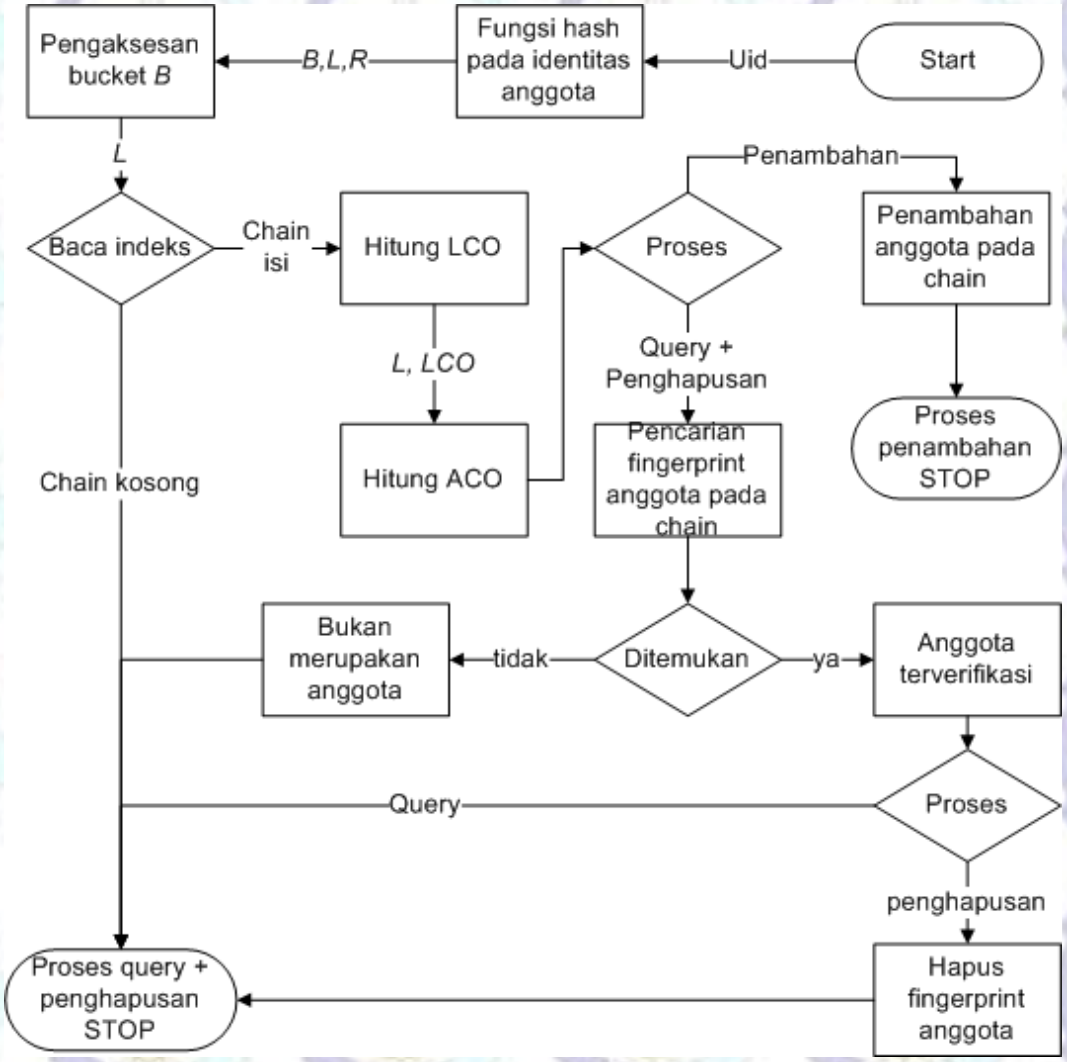
$$cell = \left\lceil \log_2 \left(\frac{\lambda}{FPR} \right) \right\rceil \tag{2.17}$$

$$FPR = \frac{\lambda}{2^{cell}},$$

$$\lambda = \frac{N}{B.L}, \tag{2.18}$$

dengan *cell* adalah ukuran fingerprint, λ merupakan rata-rata jumlah fingerprint pada setiap chain, *FPR* adalah peluang false positif, *N* adalah jumlah pengguna atau pengirim, *B* adalah jumlah bucket dan *L* adalah ukuran chain . Parameter ini direkomendasikan bernilai dibawah 1 jadi isi dari chain antara 1 atau 0 sehingga dapat dijalankan operasi single rank atau linear scan nya pendek (Einziger & Friedman 2017).

Pada implementasinya, ukuran fingerprint (*cell*) dibatasi maksimum 31 bit dengan nilai fingerprint sehingga didapatkan nilai *FPR* minimum 2^{-31}. Detail dari ketiga proses pada TinySet ditunjukkan pada flowchart pada Gambar 2.18. Ketiga proses tersebut diawali dengan menjalankan fungsi hash pada paramter unik yang melekat pada anggota yaitu identitas pengguna. Hasil hash yang didapatkan dapat diekstraksi menjadi 3 variabel unik posisi dari pengguna pada TinySet. Pada bucket yang dituju, ketiga proses tersebut menelusuri indeks dari chain sehingga diketahui apakah indeks tersebut sudah terisi sebelumnya (set) atau masih kosong (unset). Proses query dan penghapusan pengguna akan keluar jika indeks masih kosong karena fingerprint yang dicari pasti tidak ada atau menandakan bukan anggota. Sedangkan pada proses penambahan pengguna, jika indeks masih kosong maka nilai indeks akan diubah menjadi set atau terisi dengan anggota baru yang akan dimasukkan pada TinySet. Setelah posisi indeks terbaca



Gambar 2.18 Flowchart TinySet

dan ditentukan, proses dilanjutkan dengan menghitung *Logical Chain Offset* (LCO) yang diartikan sebagai banyaknya chain yang terisi dan disimpan sebelum chain tujuan. Parameter ini diperlukan karena array pada TinySet menyimpan fingerprint secara terurut berdasarkan nilai indeks pada chain. Sebagai contoh untuk perhitungan LCO dari indeks nomer 5, sedangkan indeks 0 hingga 4 (indeks sebelum nomer 5) merupakan chain yang sebagian sudah terisi misal indeks 1,3 dan 4 sehingga nilai LCO dari indeks 5 adalah 3. Angka 3 ini merupakan penjumlahan chain yang nilainya tidak kosong (set). Pada realitanya, setiap chain bisa berisi lebih dari satu anggota atau fingerprint oleh karena itulah dilakukan perhituangan *Actual Chain Offset* (ACO) yang diartikan sebagai offset pada array

untuk mencari chain tujuan. Parameter ini bertugas untuk menelusuri fingerprint dari chain pertama hingga chain tujuan sesuai dengan LCO dengan memperhatikan parameter lain yaitu isLast untuk menentukan apakah suatu indeks pada chain telah berganti atau tidak. Proses penelusuran ini akan berhenti jika islast yang ditelusuri berjumlah sama dengan LCO. Nilai ACO dipastikan lebih besar atau sama dengan nilai LCO. Kedua parameter tersebut akan bernilai sama jika chain yang set memiliki tepat satu anggota. Setelah posisi ACO ditentukan proses akan dilanjutkan dengan penambahan fingerprint. Sedangkan pada proses query dan penghapusan, proses dilanjutkan dengan mencocokkan fingerprint dari anggota yang dicari dengan yang tersimpan pada array TinySet. Jika ditemukan maka anggota tersebut valid dan pada proses penghapusan anggota tersebut dihapus dengan menggeser fingerprint kekiri dan jika fingerprint tersebut merupakan isLast maka indeks pada chain harus diubah menjadi unset. Meskipun TinySet memanfaatkan array tetapi penggunaan pointer dihindari karena tingginya komputasi. TinySet hanya memanfaatkan teknik *rank indexing*.

2.6. Evaluasi kinerja

Evaluasi dari metode yang diusulkan dihitung berdasarkan performansi maupun biaya dari implementasinya ketika digunakan baik pada simulasi maupun test bed pada perangkat JSN.

Pada tahap simulasi parameter yang digunakan untuk mengukur keberhasilan metode yang diajukan yaitu pengukuran delay atau waktu pemrosesan di sisi penerima. Hal ini menjadi penting mengingat tingginya delay pada proses pembuatan puzzle dengan waktu yang tidak dapat diprediksi sebelumnya. Parameter yang dimanfaatkan untuk mengamati proses pengiriman ini adalah jumlah dan rata iterasi hash baik yang didapatan dari percobaaan sebanyak *n* sample. Nilai ini kemudian dimodelkan ke dalam persamaan matematika. Kedekatan antara nilai ambang aktual dengan pendekatan nilai ambang dari kandidat fungsi hasil *curve fitting* diukur menggunakan R-squared dengan rumus

$$\boldsymbol{R^2 \equiv 1 - \frac{\sum_{i=1}^{n}\left(Y_{[i]} - X_{[i]}\right)^2}{\sum_{i=1}^{n}\left(Y_{[i]} - \bar{Y}\right)^2}}, \quad (2.19)$$

dan RMSE dengan rumus

$$RM\boldsymbol{SE} = \sqrt{\frac{\sum_{i=1}^{n}\left(X_{[i]} - Y_{[i]}\right)^2}{\boldsymbol{n}}}, \quad (2.20)$$

dengan $X_{[i]}$, $Y_{[i]}$, $\bar{Y}$, dan n adalah nilai prediksi dari data ke-i, nilai aktual dari data ke-i, rata-rata nilai aktual dan banyaknya data.

Keamanan dan performansi berbanding terbalik, semakin tinggi keamanan dibutuhkan kompleksitas yang tinggi sehingga performansi menurun. Karena skema yang diajukan bersifat dinamis, parameter yang digunakan untuk mengukur jarak antara kekuatan puzzle yang digunakan dengan nilai maksimumnya adalah *Mean Absolute Deviation* (MAD). Parameter tersebut digunakan untuk menghitung nilai rata-rata dari jarak sebuah titik dengan titik pusatnya. Titik pusat yang digunakan pada penelitian ini adalah kekuatan puzzle maksimum yang bisa dijalankan oleh pengirim sehingga disebut MAD_l. Perhitungan dari MAD ditunjukaan pada

$$MAD_L = \frac{\sum_{i=1}^{n}|l_i - l_{maks}|}{n}. \quad (2.21)$$

Pada Bernoulli trial, nilai ini dapat diukur berdasarkan nilai ekspektasi. Standar deviasi dari kekuatan puzzle tertuang pada (Moore et al. 2009).

$$\sigma_l = \sqrt{E(l^2) - [E(l)]^2}$$

$$= \sqrt{\left[\sum_{i=0}^{l_{maks}}\left(l_i^2 . PS_{[success,i]}\right)\right] - \left[\sum_{i=0}^{l_{maks}}\left(l_i . PS_{[success,i]}\right)\right]^2}. \quad (2.22)$$

Persamaan ini digunakan karena skema puzzle merupakan bagian dari kasus variabel random diskret sehingga nilai sebaran dapat diketahui melalui peluang keberhasilan atau nilai ekspektasi pada setiap iterasi.

Skema yang diajukan harus memiliki tingkat keamanan yang tinggi akan tetapi dengan tetap memperhatikan sumber daya yang dibutuhkan mengingat

JSN memiliki keterbatasan. Parameter yang digunakan sebagai acuan dalam menjaga keseimbangan antara keamanan dan biaya komputasi ada peluang ditemukannya solusi. Pada setiap iterasi, bilangan random dibangkitkan sebagai kandidat solusi. Peluang ditemukannya solusi untuk setiap iterasi yang dijalankan tercantum pada Persamaan (2.6).

Selain performansi, hal lain yang perlu dipertimbangkan adalah biaya implementasi dari metode yang diajukan. Biaya implementasi terdiri dari komunikasi, ruang penyimpanan dan kompleksitas. Biaya komunikasi diukur dari seberapa besar ukuran pesan yang dikirim atau dipertukarkan antara pengirim dan penerima. Ruang penyimpanan bergantung terhadap parameter apa yang perlu disimpan dan besarnya kode untuk membangun metode yang diajukan. Pada pengguna dengan jumlah yang terbatas, ruang penyimpanan bergantung pada seberapa banyak identitas pengguna yang mampu ditampung atau dibatasi di sisi penerima atau node sensor. Pada pengguna jamak atau multi-user besarnya ruang penyimpanan bergantung pada ukuran TinySet yang dibentuk pada tahap inisialisasi. Ruang penyimpanan pada TinySet dirumuskan pada

$$TS_{size} = B \times \left(\log_2 B + L + \left(B_{cap} \times (cell + 1)\right)\right). \quad (2.23)$$

Sedangkan B_{cap} merupakan kapasitas bucket yang dirumuskan pada

$$B_{cap} = \lceil L \times \lambda \rceil . \quad (2.24)$$

Parameter ini bertujuan untuk menunjukkan kapasitas maksimum dari item atau anggota pada setiap bucket. Item tersebut bisa tersebar pada indeks yang berbeda akan tetapi harus berada pada bucket yang sama.

Pada pemanfaatan perangkat keras atau test bed dibutuhkan pengukuran untuk menghitung besaran sumber daya yang dibutuhkan. Penelitian ini memanfaatkan parameter energi yang pengukurannya dibantu dengan alat multimeter. Energi yang dibutuhkan pada perangkat keras diukur berdasarkan

$$E = V \times I \times t . \quad (2.25)$$

Tegangan yang dilambangkan dengan simbol *V* pada perangkat mikrokontroller ATMega 2560 dalam hal ini Arduino beroperasi pada nilai 5 Volt. Sedangkan arus mengalir dan waktu yang dibutuhkan untuk menjalankan sebuah metode diukur pada saat proses tersebut dijalankan.

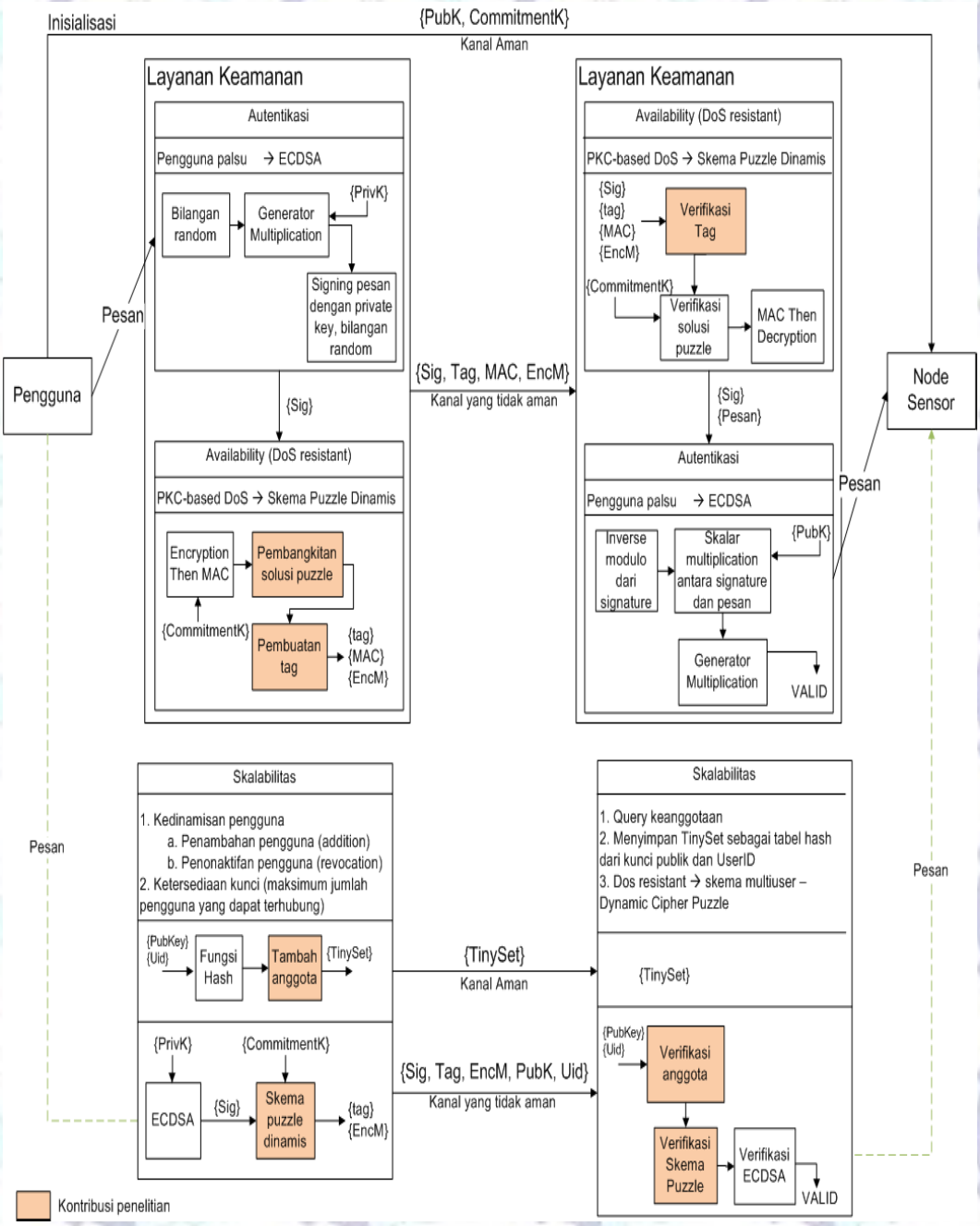
Parameter – parameter ini akan dimanfaatkan pada tahap simulasi maupun testbed pada kasus JSN. Tujuan utamanya adalah mengamati komunikasi antara pengirim atau pengguna menuju penerima atau node sensor. Berbagai pengukuran pada parameter tersebut akan dimanfaatkan untuk bahan perbandingan antara skema atau metode yang diajukan dengan metode saat ini (eksisting) yang secara garis besar dikelompokkan menjadi tiga bagian utama dan dijelaskan secara detail pada Bab 3.

BAB 3
METODE PENELITIAN

Bab ini menjelaskan tentang usulan skema yang dibangun dalam rangka menjawab permasalahan sehingga tujuan dari penelitian ini tercapai. Gambaran umum dari kontribusi penelitian dibahas pada Bab 3.1. Penjelasan detail dari skema yang dibangun dibagi menjadi 2 bagian utama yaitu skema Puzzle Dinamis dan keanggotaan menggunakan TinySet yang dijelaskan pada Bab 3.2 dan 3.3.

3.1. Skema penelitian yang diajukan

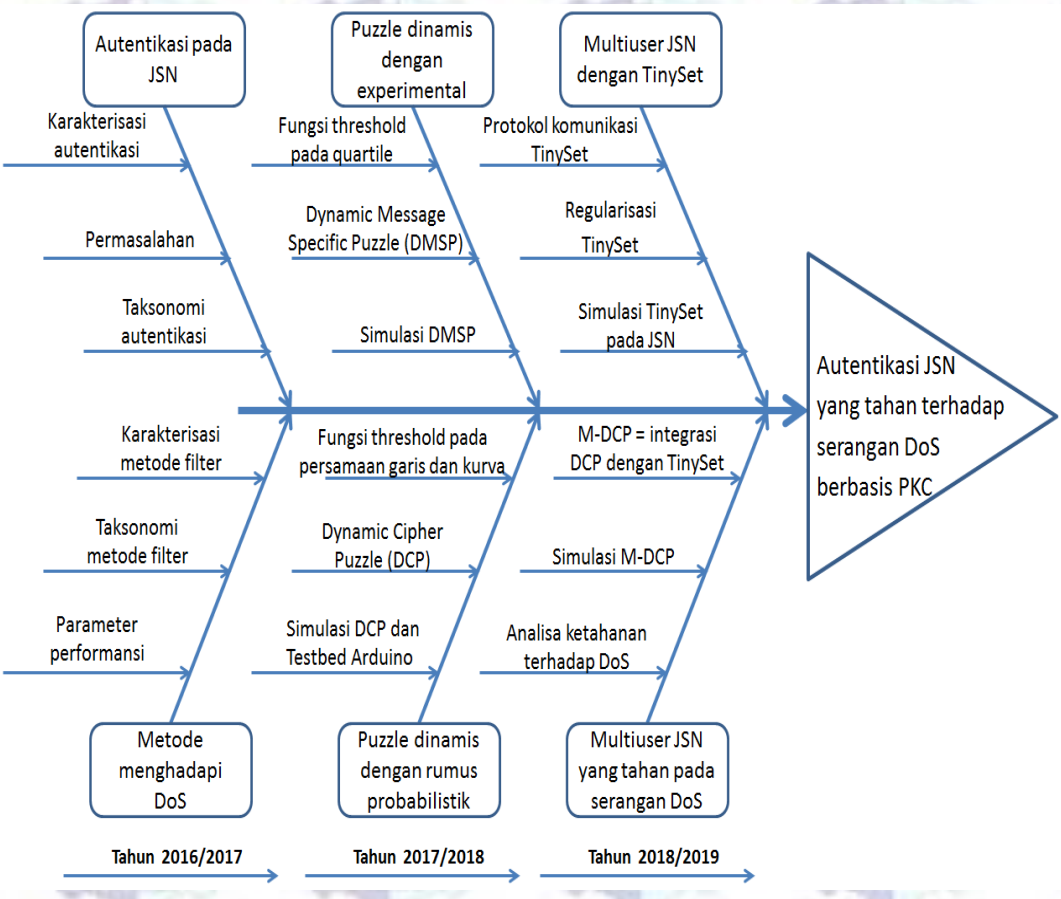
Penilitian ini bertujuan utama untuk membangun sistem yang mampu menghadapi serangan DoS berbasis PKC. Skema yang diajukan harus dapat menyesuaikan dengan kemampuan JSN yang memiliki keterbatasan sumber daya. Komunikasi antara pengirim dan penerima secara garis besar diilustrasikan pada Gambar 3.1. Skema ini dibangun berdasarkan dua fokus utama yaitu ketahanan terhadap serangan terutama DoS dan rendahnya kompleksitas khususnya di sisi penerima. Oleh karena telah memenuhi kedua persyaratan tersebut, skema puzzle dipilih sebagai metode filter dalam autentikasi pesan pengguna. Skema puzzle ini digunakan untuk mendampingi algoritma signature utama yaitu ECDSA. Perbedaan dari skema yang dibangun dengan skema puzzle sebelumnya (Aura et al. 2000; Ning & Liu 2008; Tan et al. 2013) terletak pada proses pembuatan puzzle dan pengiriman indeks yang bersifat implisit. Selain bertujuan untuk menurunkan delay pada sisi pengirim, metode ini dapat mengaburkan nilai kekuatan puzzle pada pesan yang ditransmisikan. Dengan adanya tambahan mekanisme tag, proses verifikasi di sisi penerima meningkat. Detail mengenai dampak dan proses pembangunan tag akan dijelaskan lebih lanjut pada Bab 3.2.3.



Gambar 3.1 Usulan blok diagram autentikasi JSN multiuser

Kontribusi selanjutnya dibangun dalam rangka menghadapi jumlah pengguna atau pengirim yang meningkat. Skema keanggotaan yang rendah sumber daya diperlukan karena mekanisme verifikasi pengirim berada pada perangkat yang memiliki keterbatasan sumber daya. TinySet diajukan sebagai struktur data berbasis probabilitas yang memanfaatkan tabel hash sehingga hemat

ruang penyimpanan. Perbedaan skema yang dibangun dengan TinySet sebelumnya (Einziger & Friedman 2017) adalah adanya optimasi pada tahap inisialisasi dengan metode regularisasi yang dibahas detil pada Bab 3.3.1. Selain itu, pertahanan terhadap DoS untuk autentikasi multiuser pada JSN memerlukan perubahan pada skema puzzle yang sudah diajukan sebelumnya. Hal ini dikarenakan parameter yang dikirimkan oleh pengguna berbeda dengan kondisi pada saat kontribusi pertama dibangun. Pada kontribusi pertama jumlah pengguna masih terbatas, sehingga JSN dapat menyimpan kunci publik dan identitas pengguna secara utuh. Jika jumlah pengguna meningkat dan sumber daya pada JSN tidak mencukupi maka diperlukanlah metode penyimpanan pengguna yang lebih kompres dan hemat ruang penyimpanan.



Gambar 3.2 Diagram tulang ikan autentikasi JSN dalam menghadapi serangan DoS berbasis PKC

Pembagian detil kontribusi dapat dilihat melalui diagram tulang ikan pada Gambar 3.2. Selain kontribusi, diagram ini juga menunjukkan pembagian tiap kontribusi pada semester sehingga total periode penelitian dapat diketahui yaitu 6 semester atau 3 tahun. Diagram ini diawali dengan memfokuskan area penelitian yaitu autentikasi pada JSN. Pengenalan karakteristik, identifikasi permasalahan dan taksonomi pada JSN lebih didalami untuk mengetahui *state of the art* dan tujuan yang akan dicapai pada semester berikutnya. Pada semester 2, penelitian ini sudah lebih fokus untuk membahas permasalahan serangan DoS berbasis PKC pada autentikasi JSN. Pemahaman mengenai bagaimana cara menghadapi serangan pada setiap metode filter yang ada, pembangunan taksonomi dan parameter apa yang dijadikan tolak ukur keberhasilan. Hasil dari investigasi ini dituangkan pada karya ilmiah survei mengenai ketahanan terhadap serangan DoS pada JSN. Pada semester 3, penelitian difokuskan pada salah satu metode filter yang dianggap paling sesuai dengan karakteristik JSN yaitu skema puzzle. Percobaan mulai dijalankan dan disimulasikan, sehingga dicetuskanlah ide kekuatan puzzle dinamis menggunakan fungsi ambang. Pada semester 4, ide ini dioptimalisasi untuk mendapatkan fungsi ambang yang lebih baik. Fungsi ambang yang dibangun, didekati menggunakan persamaan garis dan eksponensial pada diagram kartesian. Selain itu, tingkat keamanan ditingkatkan dengan adanya aspek *confidentiality* sehingga pesan yang dikirimkan tidak lagi pesan asli melainkan yang sudah terenkripsi. Pada semester 5, ruang lingkup JSN dikembangkan dengan melibatkan pengirim yang jumlahnya besar sehingga ruang penyimpanan JSN tidak mampu menampung jika harus menyimpan kunci publik dan identitas pengguna secara utuh. Pemanfaatan, implementasi dan optimalisasi TinySet pada JSN menjadi fokus utama pada semester ini. Regularisasi dibangun untuk memudahkan proses inisialisasi pada saat pendaftaran pengguna. Pada semester akhir, pemanfaatan TinySet yang didampingi dengan skema puzzle dinamis dibangun dan dianalisa sebagai tujuan utama dari penelitian ini. Detail dari kegiatan pada semester 1 dan 2 telah dibahas sebelumnya yaitu pada Bab 2.3, 2.4, dan 2.5. Sedangkan kegiatan semester 4 hingga 6 akan dibahas pada Bab 3.2 dan 3.3.

3.2. Skema Puzzle Dinamis

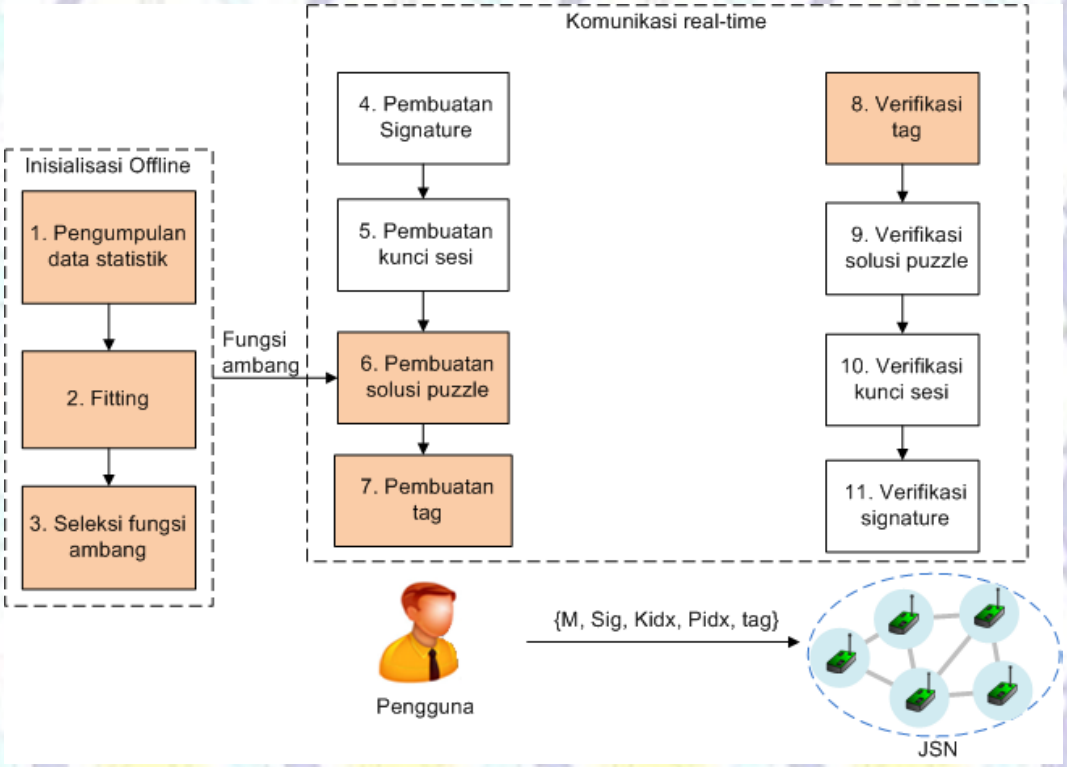
Skeme puzzle dinamis merupakan skema puzzle yang dibangun dengan kekuatan puzzle yang tidak tetap dengan tujuan delay pada sisi pengirim yang rendah dan terkontrol. Selain itu, skema ini juga bersifat independen sehingga tidak bergantung pada jenis digital signature tertentu. Dalam mencapai tujuan tersebut, dibangunlah fungsi ambang yang akan dijadikan acuan untuk kedinamisan kekuatan puzzle. Fungsi ini membatasi jumlah iterasi hash pada setiap tingkatan kekuatan puzzle. Inisialisasi parameter hanya digunakan untuk menentukan maksimum nilai kekuatan puzzle sedangkan kekuatan puzzle yang akan dikirim tidak dapat diketahui tetapi jumlah iterasi hashnya dibatasi dengan jaminan ditemukannya solusi melalui parameter tertentu. Fungsi ambang ini bersifat independen dan dapat diimplementasikan pada semua skema puzzle.

Penelitian mengenai skema puzzle dinamis diawali dengan pembentukan fungsi ambang menggunakan cara eksperimental. Metode ini dijadikan sebagai awalan untuk implementasi dan memperdalam pengetahuin mengenai karakteristik skema puzzle. Fungsi ambang ini diterapkan pada skema puzzle yang ringan dan sudah banyak digunakan yaitu MSP (Ning & Liu 2008). Pemanfaatan fungsi ambang dengan cara eksperimental pada MSP dijelaskan pada Bab 3.2.1.

Penelitian dilanjutkan dengan optimasi fungsi ambang dengan memperhatikan aspek rumus probabilistik. Metode ini diterapkan pada skema puzzle yang digunakan untuk diseminasi data atau kode pada JSN yaitu cipher puzzle (Tan et al. 2013). Diawali dengan tingginya komputasi pada cipher puzzle dibanding dengan MSP, maka fungsi ambang ini dibangun untuk menurunkan delay seoptimal mungkin. Pemanfaatan fungsi ambang dengan rumus probabilistik pada cipher puzzle dijelaskan pada Bab 3.2.2.

3.2.1. Fungsi ambang dengan eksperimental

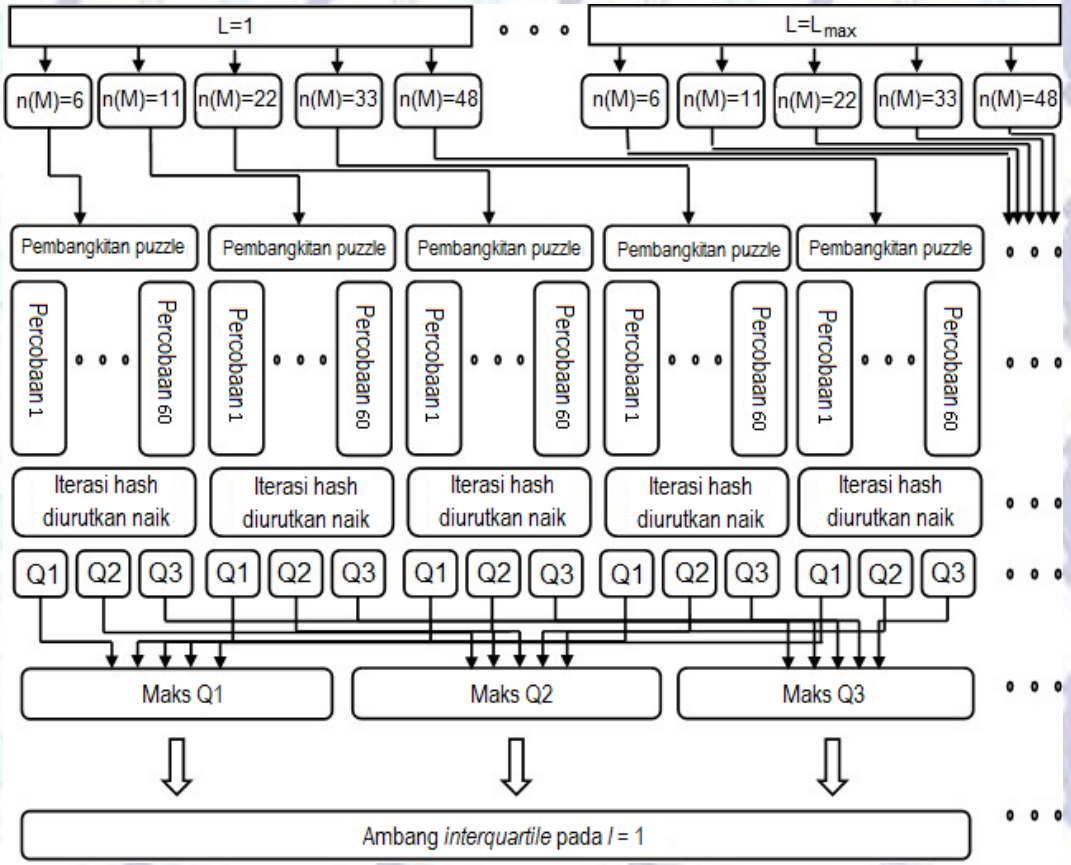
Skema ini merupakan pengembangan dari MSP sehingga diberi nama *Dynamic Message Specific Puzzle* (DMSP). Prosedur komunikasi skema DMSP dapat dilihat pada Gambar 3.3.



Gambar 3.3 Diagram komunikasi skema DMSP

Pembangunan skema DMSP diawali dengan pembangkitan fungsi ambang secara offline. Proses ini berguna untuk mencari fungsi yang optimal dan digunakan sebagai batas atas jumlah iterasi hash yang bisa dijalankan pada setiap tingkatan kekuatan puzzle. Berdasarkan kandidat fungsi ambang yang didapat, akan dijalankan pemilihan fungsi ambang yang optimal. Pada pembentukan fungsi ambang eksperimental, proses yang dijalankan dibagi menjadi 2 proses utama yaitu tahap pengumpulan data statistik dan proses fitting.

Pengumpulan data statistik dijalankan pada pembangkitan puzzle yang memenuhi segala kemungkinan skenario. Tujuan utama dari mekanisme ini adalah untuk mencari nilai perwakilan dari tiap kekuatan puzzle *l* melalui metode statistik persebaran data. Pertama proses ini dijalankan dengan cara diulang dan hasil percobaan sampling diamati untuk mewakili populasi tertentu. Nilai akhir yang didapat dapat digunakan sebagai acuan kuartil dari sebaran data untuk jumlah iterasi fungsi hash. Ilustrasi dari metode ini dapat dilihat pada Gambar 3.4.



Gambar 3.4 Mekanisme pengumpulan data statistik dalam pembentukan fungsi ambang

Percobaan dijalankan hingga memenuhi minimum sample untuk mengetahui distribusi sebuah populasi yaitu tepatnya 60 perulangan. Alasan lainnya, rentang nilai iterasi hash antar percobaan yang dilakukan untuk setiap varian panjang pesan tersebut sudah dekat (jarak antar satu percobaan dengan yang lain setelah hasilnya diurutkan cukup rendah). Fungsi hash yang digunakan pada percobaan ini adalah SHA1 karena merupakan salah satu fungsi hash kriptografi dengan panjang digest atau luaran hash yang paling pendek dengan tingkat keamanan yang tinggi dibandingkan dengan fungsi hash non-kriptografi. Fungsi hash kriptografi adalah kelas khusus dari fungsi hash yang memenuhi properti tertentu sehingga bisa dimanfaatkan pada kasus-kasus di bidang kriptografi. Properti yang harus dipenuhi untuk membentuk fungsi hash kriptografi terdiri dari 3 tingkatan, yaitu (Katz & Lindell 2008):

a. *Collision resistance* yang diartikan sebagai tidak ditemukannya penyerang yang efisien yang bisa menemukan collision kecuali dengan peluang yang sangat kecil. Collision secara matematis dapat dituliskan sebagai

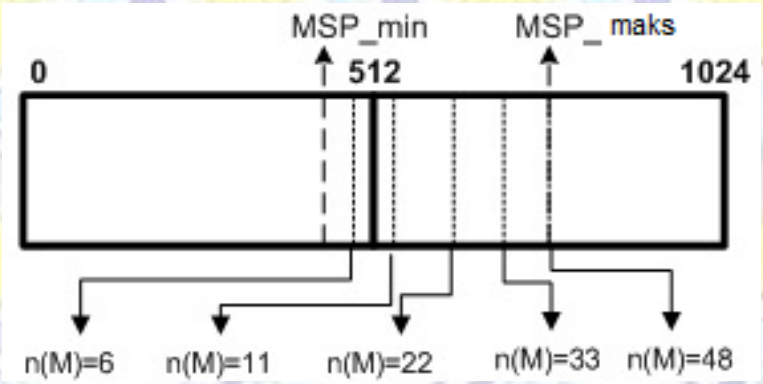
$$H_1(x) = H_1(x'), \text{ dengan } x \neq x'. \quad (3.1)$$

b. Second pre-image resistance yang dipenuhi jika fungsi hash bisa menemukan $H_1(x') = H_1(x)$, dengan $x \neq x'$ dalam waktu polinomial jika diketahui fungsi hash $H_1(\ )$ dan nilai inputan atau pesan *x*.

c. Pre-image resistance yang dipenuhi jika tidak ditemukannya penyerang yang bisa mendapatan nilai *x* jika diketahui fungsi hash $H_1(\ )$ dan nilai digest *s*, dengan $H_1(x) = s$ dengan waktu polinomial. Tidak mungkin diketahui isi pesan dari nilai hashnya kecuali diujicobakan satu persatu dari setiap kemungkinan (brute force).

Serangan yang bisa merusak SHA1 baru ditemukan tahun 2017 itupun dengan syarat terdapat sebuah instan collision yang diketahui sebelumnya dengan kebutuhan 2^{63} iterasi hash (Stevens et al. 2017). SHA1 banyak digunakan karena memiliki komputasi rendah dan diikuti dengan waktu pemrosesan yang cepat. Oleh karena itu, SHA1 juga dipilih sebagai fungsi hash utama pada skema puzzle yang dikembangkan sebelumnya (Ning & Liu 2008; Tan et al. 2013).

Pemilihan lima varian panjang pesan dijalankan sesuai blok atau pengelompokan pesan dari fungsi hash SHA1 karena inputan pengguna akan diberikan padding jika tidak memenuhi satu blok (per blok = 512 bit). Ilustrasi

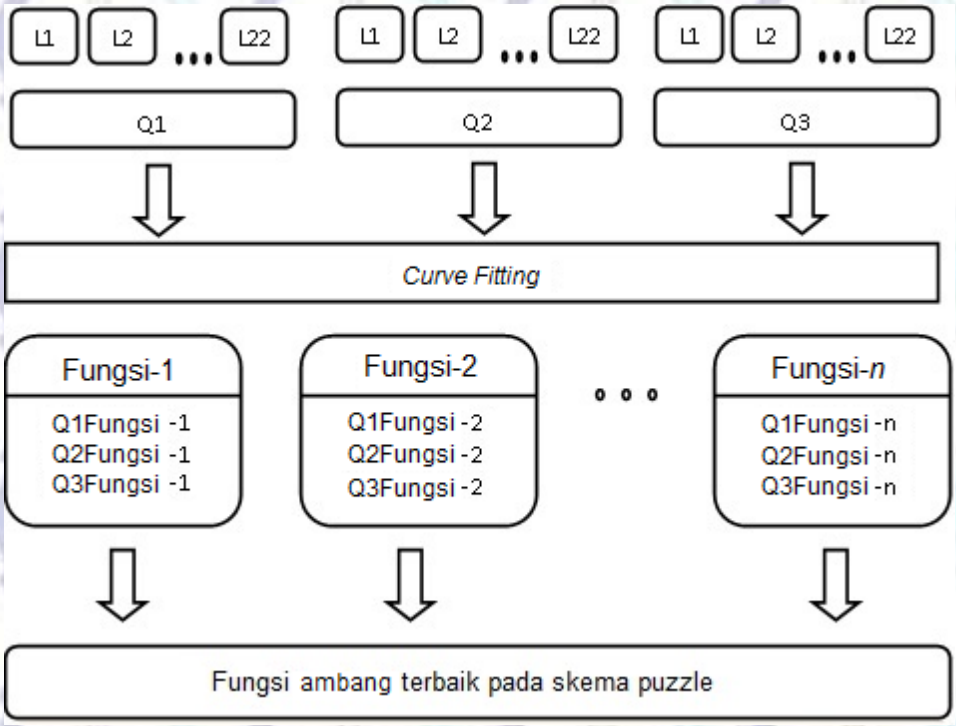


Gambar 3.5 Proses padding pada SHA1

blok input pengguna pada SHA1 dapat dilihat pada Gambar 3.5. Panjang pesan pengguna pada MSP berkisar antara 1 hingga 48 byte sehingga total pesan yang dikirim panjangnya 5 hingga 102 byte atau 440 hingga 806 bit. Blok SHA1 yang memungkinkan untuk terisi hanya 1 hingga 2 blok saja. Pada sebuah percobaan dalam pengambilan sampel, isi pesan pada setiap varian panjang pesan bernilai tetap. Hal ini didasarkan pada filosofi pembangkitan puzzle dengan nilai random berasal dari kandidat solusi puzzle pada setiap iterasi hash sedangkan informasi yang akan dikirim bernilai tetap sesuai ilustrasi skema puzzle pada Gambar 2.10. Kandidat solusi puzzle berukuran 4 byte sehingga banyaknya kemungkinan bilangan random sebesar 2^{32}.

Hasil percobaan yang didapatkan dirutkan secara menaik kemudian diambil nilai Q1, Q2 dan Q3 dari masing-masing varian panjang pesan. Nilai Q1, Q2 dan Q3 disebut sebagai kuartil pertama, kedua dan ketiga yang didapat dengan mengambil nilai ke- 25%, 50% dan 75% dari data. Hasil akhir yang diambil untuk mewakili kuartil dari kekuatan puzzle tertentu merupakan nilai tertinggi dari kandidat yang ada, sehingga didapatkan angka yang benar-benar dapat mewakili distribusi data.

Langkah selanjutnya adalah fitting. Mekanisme fitting berfungsi untuk mencari model matematika dari nilai ambang yang didapat pada proses pengambilan data statistik. Nilai tersebut terdiri dari kuartil 1 hingga 3 yang mewakili setiap kekuatan puzzle paling rendah yaitu 1 hingga kekuatan puzzle maksimum. Setiap kuartil memiliki nilai kandidat fungsi ambang yang didekati (*fitted*) ke dalam kurva menggunakan bantuan aplikasi MATLAB. Detail proses ini diilustrasikan pada Gambar 3.6. Dari fungsi yang didapat, akan ditentukan persamaan mana yang paling merepresentasikan nilai ambang dan digunakan sebagai kandidat fungsi ambang. Seleksi fungsi ambang dilanjutkan kembali untuk memilih fungsi yang paling optimum ketika diimplementasikan pada DMSP. Parameter-parameter yang digunakan untuk menentukan dan menyeleksi fungsi ambang yaitu : rata-rata, maksimum iterasi hash, *Mean Absolute Deviation* (MAD) dan peluang tidak ditemukannya solusi puzzle.



Gambar 3.6 Mekanisme fitting dalam pembentukan fungsi ambang

Fungsi ambang yang optimal dan sesuai dengan karakteristik JSN selanjutnya dimanfaatkan pada komunikasi skema DMSP. Berdasarkan Gambar 3.3 dan Algoritma 3.1 baris ke-1, langkah pertama yang dilakukan oleh pengirim atau pengguna JSN adalah pembentukan signature dari pesan yang akan dikirim. Algoritma signature yang digunakan adalah ECDSA (Johnson et al. 2001) sesuai dengan alasan mengapa MSP memanfaatkannya yang telah dibahas pada Bab 2.3. Detail dari algoritma ini dipaparkan pada Lampiran D. Proses ini dilanjutkan dengan pembangkitan kunci sesi diikuti dengan pencarian solusi puzzle yang ditunjukkan pada Algoritma 3.1 baris ke-7 hingga 18. Informasi yang akan dikirimkan seperti pesan, signature, kunci sesi, indeks dan bilangan random digabung kemudian dihitung nilai hash-nya. Nilai random yang didapat merupakan kandidat solusi puzzle jika hasil operasi hash sesuai dengan pola angka 0 berurutan sepanjang *l*. Jika tidak sesuai dengan pola, maka bangkitkan ulang bilangan random. Jumlah iterasi hash akan bertambah seiring dengan proses tidak ditemukannya solusi. Proses DMSP dilanjutkan dengan perhitungan tag

Algoritma 3.1 Pembangkitan Dynamic Message Specific Puzzle pada pengirim

Input: Kunci sesi K_{idx}, *index* dari proses inisialisasi sedangkan M masukan dari pengguna
Output: Pengiriman paket P_{DMP} pada sensor node atau solusi puzzle tidak ditemukan

```
1  Sig ⇐ ECDSA_sign(M);
2  index++;
3  L, L_prev ⇐ Lmax;
4  iteration ⇐ 0;
5  threshold ⇐ Threshold_Function(L);
6  finding ⇐ false;
7  while (not(finding)) and (L >0) do
8      if iteration < threshold then
9          iteration ++;
10         P_idx ⇐ genRandom();
11         if SHA1(index||M||Sig||K_idx||P_idx).substr(0,L) == zeros(0,L) then
12             finding ⇐ true;
13         end
14     else
15         L − −;
16         threshold ⇐ threshold + Threshold_Function(L);
17     end
18 end
19 tag ⇐ SHA1(index + L_prev + L).substr(0,L);
20 L_prev ⇐ L;
21 P_DMP ⇐ (tag||M||Sig||K_idx||P_idx) ;
22 Send (P_DMP)
```

yang dibahas pada Bab 3.2.3. Setelah tag terbentuk, semua informasi yang didapat Algoritma 3.1 baris ke-21 akan dikirimkan sesuai dengan formasi pada Gambar 3.7. Total panjang paket yang dikirim adalah 102 byte. Tag menggantikan nilai indeks yang berukuran sebesar 2 byte. Sedangkan signature dan pesan berturut-turut sebesar 40 dan 48 byte. Kunci sesi yang dihitung menggunakan keychain memiliki panjang 8 byte. Paket ini ditutup dengan bilangan random yang dikirim sebagai solusi puzzle sebesar 4 byte. Sedangkan rangkaian proses pada sisi penerima dijelaskan secara mendetail pada Algoritma 3.2.

Gambar 3.7 Rincian isi paket skema DMSP

Algoritma 3.2 Verifikasi Dynamic Message Specific Puzzle pada penerima

Input: Kunci commitment K_0, $index$, L_{prev} telah tersimpan pada node sensor dan menerima paket P_{DMP}
Output: Memastikan paket P_{DMP} terautentikasi dan resistan terhadap serangan DoS berbasis PKC

```
1  Receive(P_DMP);
2  LenMtag ⇐ Length(P_DMP) − (320 + 64 + 32);
3  Mtag ⇐ P_DMP.substr(0,LenMtag);
4  L ⇐ L_max;
5  Vertag ⇐ false
6  while (L > 0) and (not(Vertag)) do
7  |  if (Mtag.substr(0,L) == SHA1((index + 1) + L_prev + L).substr(0,L) then
8  |  |  Vertag ⇐ true;
9  |  else
10 |  |  L − −;
11 |  end
12 end
13 if Ver_tag then
14 |  MSigKP_idx ⇐ P_DMP.substr(L,Length(P_DMP) − L)
15 |  if SHA1((index + 1)||MSigKP_idx).substr(0,L) == zeros(0,L) then
16 |  |  K_rec ⇐ P_DMP.substr(LenMtag+320,64);
17 |  |  if SHA1^(index+1)(K_rec) == K_0 then
18 |  |  |  M ⇐ Mtag.substr(L,LenMtag − L);
19 |  |  |  Sig ⇐ P_DMP.substr(LenMtag,320)
20 |  |  |  if ECDSA_ver(Sig) then
21 |  |  |  |  index++;
22 |  |  |  |  L_prev ⇐ L ;
23 |  |  |  else
24 |  |  |  |  Drop_Message_BadSignature
25 |  |  |  end
26 |  |  else
27 |  |  |  Drop_Message_sessionKey_invalid
28 |  |  end
29 |  else
30 |  |  Drop_Message_PuzzleSolution_invalid
31 |  end
32 else
33 |  Drop_Message_tagging_invalid
34 end
```

Proses pada sisi penerima diawali dengan verifikasi tag yang akan dibahas secara detail pada Bab 3.2.3. Kemudian diikuti dengan verifikasi solusi puzzle pada Algoritma 3.2 baris ke-15. Jika hash dari paket yang diterima sama dengan 0 sejumlah *l* bits maka puzzle valid dan dilanjutkan dengan verifikasi

kunci sesi pada Algoritma 3.2 baris ke-17. Fungsi hash dijalankkan sebanyak *indeks* kali pada kunci sesi yang diterima kemudian dibandingkan dengan kunci *commitment*. Jika hasilnya sama, maka kunci sesi valid dan dilanjutkan dengan proses yang paling akhir yaitu verifikasi signature yang dijelaskan pada Algoritma 3.2 baris ke-20. Kunci publik dari pengirim sudah tersimpan di sisi penerima. Sisi penerima melakukan pencocokan ID pengirim dengan kunci publik kemudian dijalankan pembangkitan signature dari kunci publik tersebut. Hasil yang diperoleh dibandingkan dengan signature yang menempel pada pesan yang diterima. Jika hasilnya sama maka proses verifikasi signature dinyatakan valid. Jika sebaliknya maka pesan yang diterima bukan dari pengirim yang tervalidasi dan indeks maupun parameter penting lainnya dari pesan yang diterima bisa diabaikan.

3.2.2. Fungsi ambang dengan rumus probabilistik

Skema ini merupakan pengembangan dari cipher puzzle (Tan et al. 2013) yang telah dibahas secara rinci pada Bab 2.3.4.3 sehingga diberi nama Dynamic Cipher Puzzle (DCP). Berbeda dengan pendekatan eksperimental, fungsi ambang pada skema ini didapatkan melalui pendekatan rumus probabilistik dari Persamaan (2.6). Prosedur komunikasi skema DCP dapat dilihat pada Gambar 3.8.

Proses DCP pada sisi pengirim diawali dengan penentuan fungsi ambang. Penentuan fungsi ambang ini hanya dijalankan satu kali di awal pembangunan sistem komunikasi antara pengguna dan JSN. Tujuan utama dari fungsi ambang adalah mengontrol jumlah iterasi hash sehingga delay pemrosesan pesan pada sisi pengirim dapat ditekan seminimum mungkin. Informasi mengenai seberapa banyak perulangan iterasi hash dibutuhkan dalam pembangunan fungsi ambang jika diketahui nilai tertentu sebagai peluang sukses atau ditemukannya solusi puzzle. Peluang ditemukannya solusi pada skema puzzle telah ditentukan (Ning & Liu 2008; Tan et al. 2013). Jika dibalik, iterasi hash yang diperlukan untuk memenuhi nilai peluang ditemukannya solusi puzzle tertentu diekspresikan pada

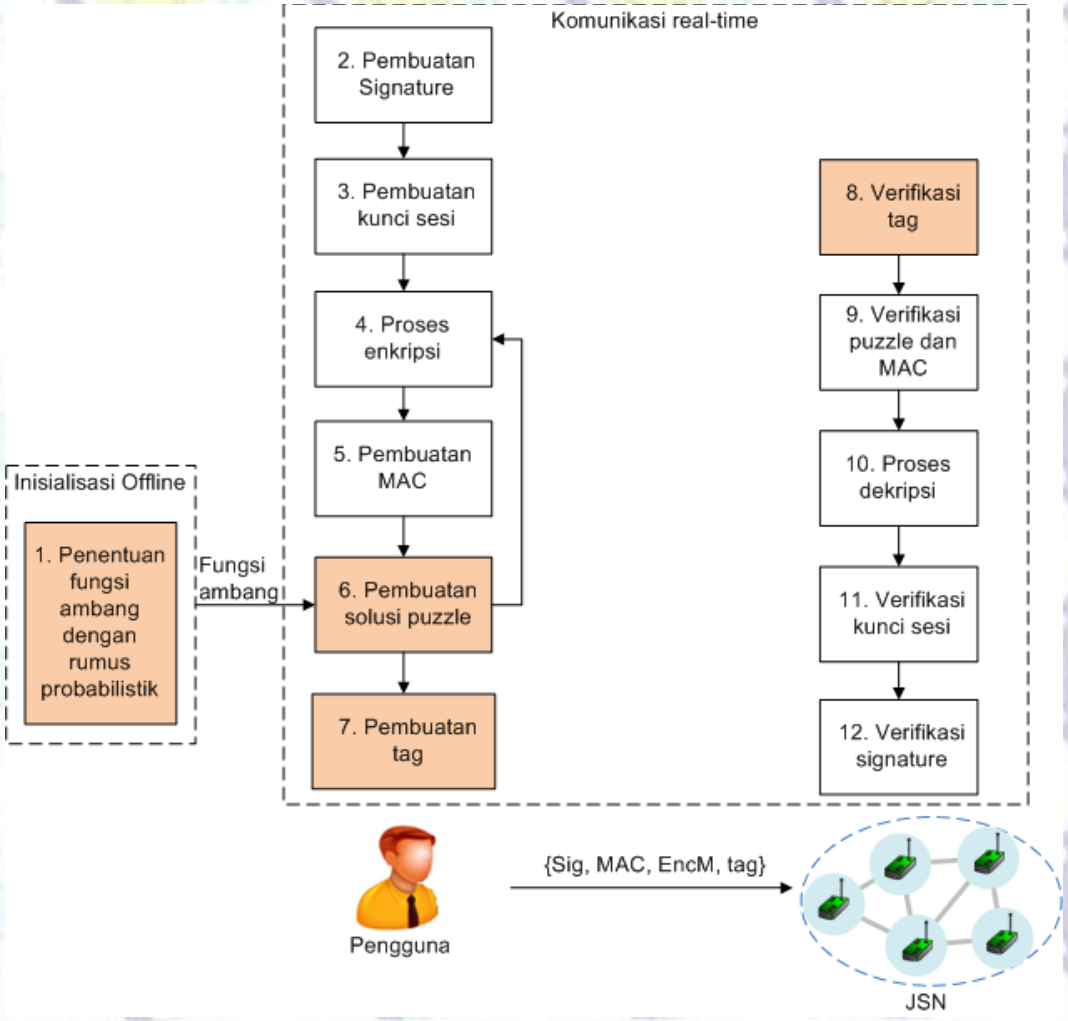
$$P_{success} = 1 - \left\{1 - \left(\frac{1}{2}\right)^{l}\right\}^{n}$$

$$\left\{1 - \left(\frac{1}{2}\right)^{l}\right\}^{n} = 1 - P_{success}$$

$$log_{10}\left\{1 - \left(\frac{1}{2}\right)^{l}\right\}^{n} = log_{10}(1 - P_{success})$$

$$n.\, log_{10}\left\{1 - \left(\frac{1}{2}\right)^{l}\right\} = log_{10}(1 - P_{success})$$

$$n = \frac{log_{10}(1 - P_{success})}{log_{10}\left\{1 - \left(\frac{1}{2}\right)^{l}\right\}} \quad . \qquad (3.2)$$



Gambar 3.8 Diagram komunikasi skema DCP

DCP yang diusulkan memiliki kekuatan puzzle yang berkisar antara l_{maks} hingga 1 sehingga maksimum iterasi hash pada skema DCP diformulasikan dengan

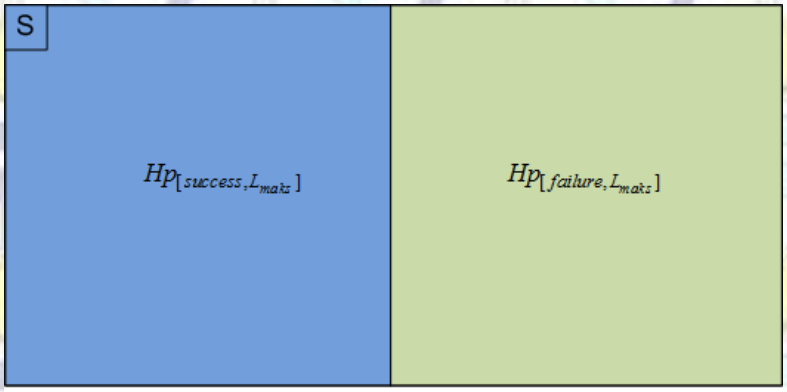
$$n_{DCP} = \sum_{i=0}^{l_{maks}-1} \frac{log_{10}\left(\mathbf{1} - \boldsymbol{P}_{[\boldsymbol{success}, \boldsymbol{l_{maks}-i}]}\right)}{log_{10}\left\{\mathbf{1} - \left(\frac{\mathbf{1}}{\mathbf{2}}\right)^{(\boldsymbol{l_{maks}-i})}\right\}} \,. \tag{3.3}$$

Peluang sukses atau peluang ditemukannya solusi menentukan peluang solusi kosong atau tidak adanya solusi puzzle dan peluang sistem secara keseluruhan. Semakin tinggi peluang kesuksesan pada suatu kekuatan puzzle maka semakin tinggi jumlah iterasi yang diperlukan. Demikian sebaliknya, semakin rendah peluang kesuksesan ditemukannya solusi maka semakin rendah nilai iterasi hash yang dibutuhkan. Pada skema puzzle dinamis dengan nilai kekuatan puzzle yang bersifat dinamis maka peluang kesuksesan ini pun bisa dimanfaatkan untuk mengontrol jumlah iterasi hash. Oleh karena itu, karakteristik dari parameter ini perlu dianalisa. Sehingga, peluang sukses pada skema puzzle dinamis didekati dengan teori himpunan tepatnya konsep irisan dan negasi. Himpunan semesta dari skema puzzle secara umum dirumuskan pada

$$S = Hp_{success} \cup \overline{Hp_{success}}$$

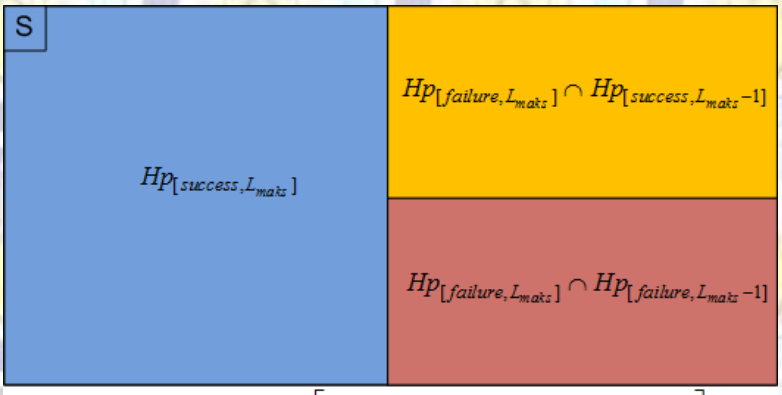
$$S = Hp_{success} \cup Hp_{failure} \,. \tag{3.4}$$

Skema DCP memulai percobaan dengan kekuatan puzzle tertinggi hingga terendah yaitu 1 bit. Pada setiap kekuatan puzzle, hasil dari pencarian solusi memiliki 2 kemungkinan yaitu sukses atau gagal. Jika gagal maka percobaan akan dimulai lagi dengan nilai kekuatan puzzle yang lebih rendah dan memiliki peluang sukses yang berbeda pula. Siklus ini akan berhenti ketika solusi puzzle ditemukan pada tingkatan kekuatan puzzle tertentu atau gagal ditemukannya solusi jika kekuatan puzzle bernilai 0.



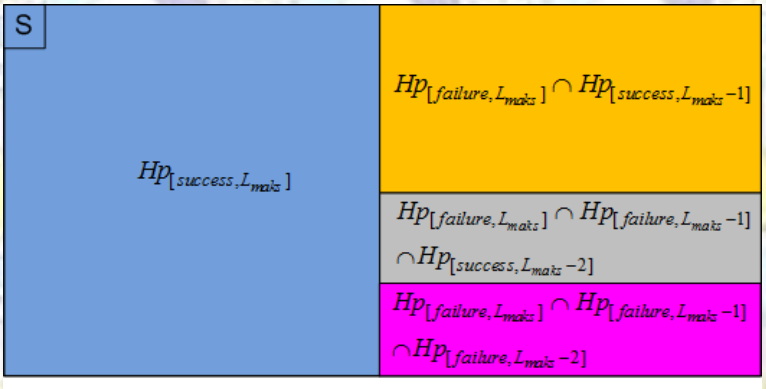
$$S = Hp_{[success,L_{maks}]} \cup Hp_{[failure,L_{maks}]}$$

(a) Peluang sukses pada $l = l_{maks}$



$$S = Hp_{[success,L_{maks}]} \cup \left[Hp_{[failure,L_{maks}]} \cap Hp_{[success,L_{maks}-1]} \right] \cup \left[Hp_{[failure,L_{maks}]} \cap Hp_{[failure,L_{maks}-1]} \right]$$

(b) Peluang sukses pada $l = l_{maks} - 1$



$$S = Hp_{[success,L_{maks}]} \cup \left[Hp_{[failure,L_{maks}]} \cap Hp_{[success,L_{maks}-1]} \right] \cup \left[Hp_{[failure,L_{maks}]} \cap Hp_{[failure,L_{maks}-1]} \cap Hp_{[success,L_{maks}-2]} \right] \cup \left[Hp_{[failure,L_{maks}]} \cap Hp_{[failure,L_{maks}-1]} \cap Hp_{[failure,L_{maks}-2]} \right]$$

(c) Peluang sukses pada $l = l_{maks} - 2$

Gambar 3.9 Diagram Venn pada peluang sukses skema DCP

Peluang sukses ketika direpresentasikan ke dalam teori himpunan untuk nilai $l = l_{maks}$ hingga $l = (l_{maks} - 2)$ diilustrasikan pada Gambar 3.9. Berdasarkan diagram Venn pada Gambar 3.9, peluang tidak ditemukannya solusi puzzle (*zero solution probability*) pada skema DCP dirumuskan dengan

$$P_{ZS} = \prod_{i=1}^{l_{maks}} P_{[failure,\ i]}$$

$$P_{ZS} = \prod_{i=1}^{l_{maks}} \left(1 - P_{[success,\ i]}\right). \tag{3.5}$$

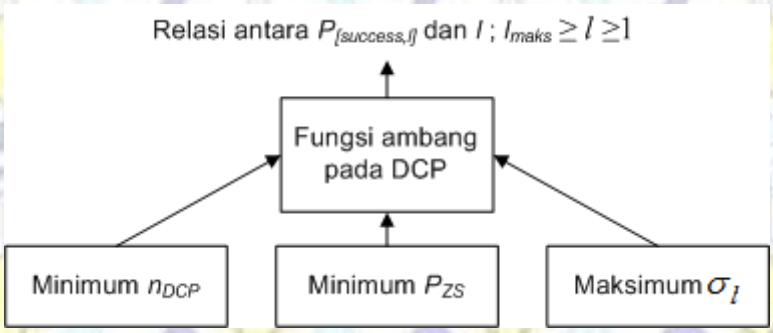
Jika diagram Venn pada Gambar 3.9 diteruskan hingga kekuatan puzzle bernilai 0 maka himpunan semesta dari skema DCP bisa dimodelkan pada

$$S = \bigcup_{i=1}^{l_{maks}} Hps_{[success,\ i]} \cup \bigcap_{i=1}^{l_{maks}} Hp_{[failure,\ i]}$$

$$Hps_{[success,\ i]} = \begin{cases} Hp_{[success,\ i]} & , i = l_{maks} \\ Hp_{[success,\ i]} \bigcap_{j=1}^{n=l_{maks}-1} Hp_{[failure,\ i+j]} & , i < l_{maks} \end{cases} \tag{3.6}$$

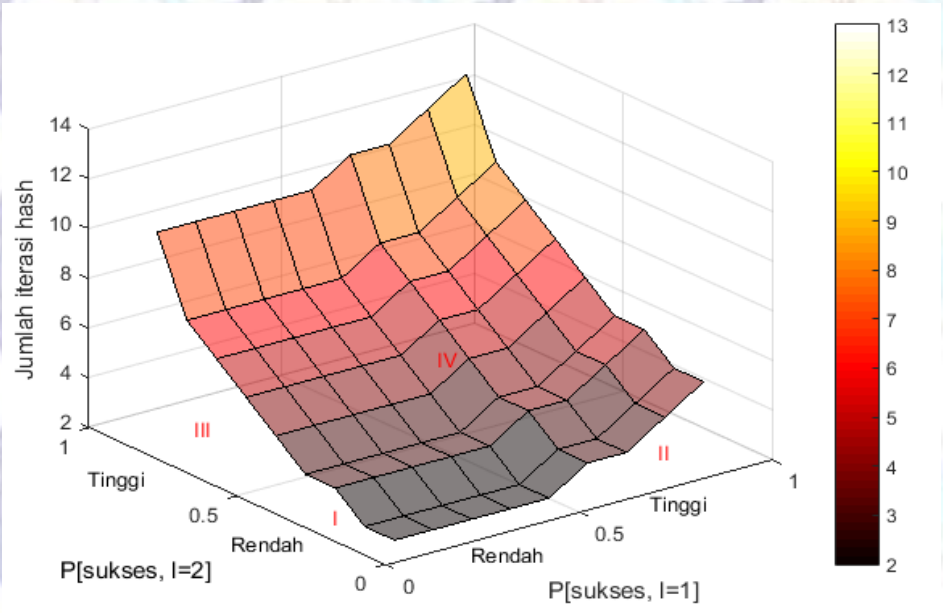
Parameter lain yang menentukan fungsi ambang yang optimum adalah rata-rata iterasi hash dan sebaran datanya. Rata-rata iterasi hash diperlukan dalam menentukan perkiraan delay minimum yang diperlukan untuk memproses pesan pada saat pengiriman. Parameter terakhir adalah standar deviasi yang diukur untuk mengetahui sebaran data dan kedekatannya dengan nilai rata-rata yang didapatkan dari Persamaan (2.22).

Tujuan utama dari pembentukan fungsi ambang adalah mencari relasi antara peluang sukses dari setiap kekutan puzzle dan nilai kekuatan puzzle itu sendiri. Ide ini terinspirasi dari Persamaan (3.3). Relasi ini dibangun karena jika nilai kekuatan puzzle dinamis maka peluang kesuksesan pada setiap kekuatan puzzle bersifat dinamis juga. Ilustrasi dari ide pembentukan fungsi ambang pada



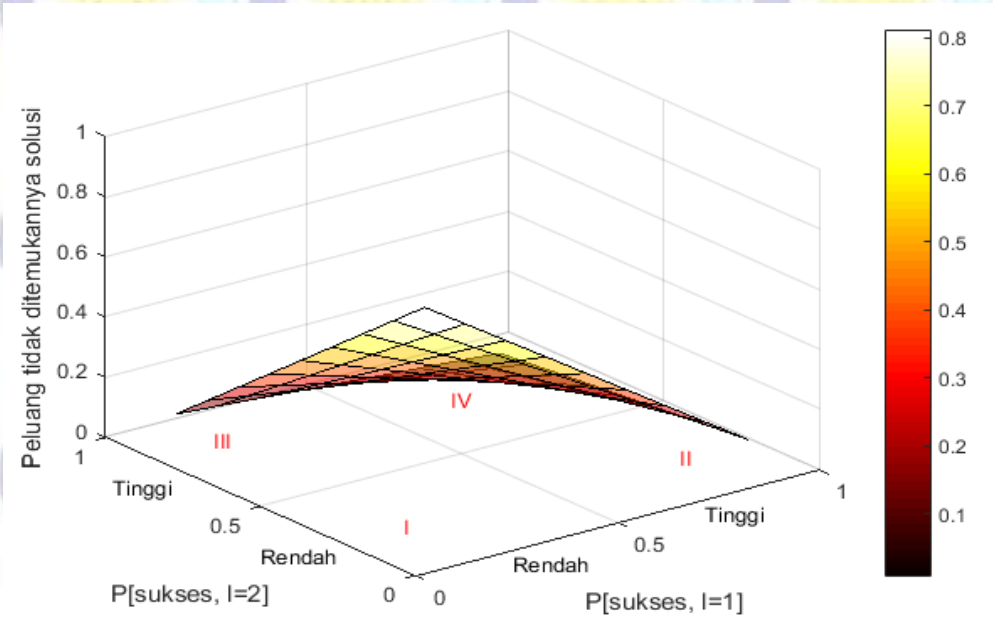
Gambar 3.10 Proses penentuan fungsi ambang yang optimum

skema DCP dapat dilihat pada Gambar 3.10. Parameter penentu fungsi ambang yang optimum adalah jumlah iterasi hash yang paling minimal, peluang tidak ditemukannya solusi dengan nilai paling minimal dan standar deviasi dari kekuatan puzzle dengan nilai yang paling tinggi. Minimumnya jumlah iterasi hash dimaksudkan untuk mendapatkan delay yang serendah mungkin di sisi pengirim, sedangkan peluang tidak ditemukannya solusi yang minimal ditujukan untuk memastikan bahwa dengan nilai iterasi hash yang ditekan, solusi puzzle tetap dapat ditemukan. Maksimumnya nilai standar deviasi dimaksudkan untuk mendapatkan nilai kekuatan puzzle yang bervariasi antara satu iterasi dengan iterasi yang lainnya sehingga susah ditebak oleh peretas.



Gambar 3.11 Jumlah iterasi hash pada DCP jika l_{maks} = 2

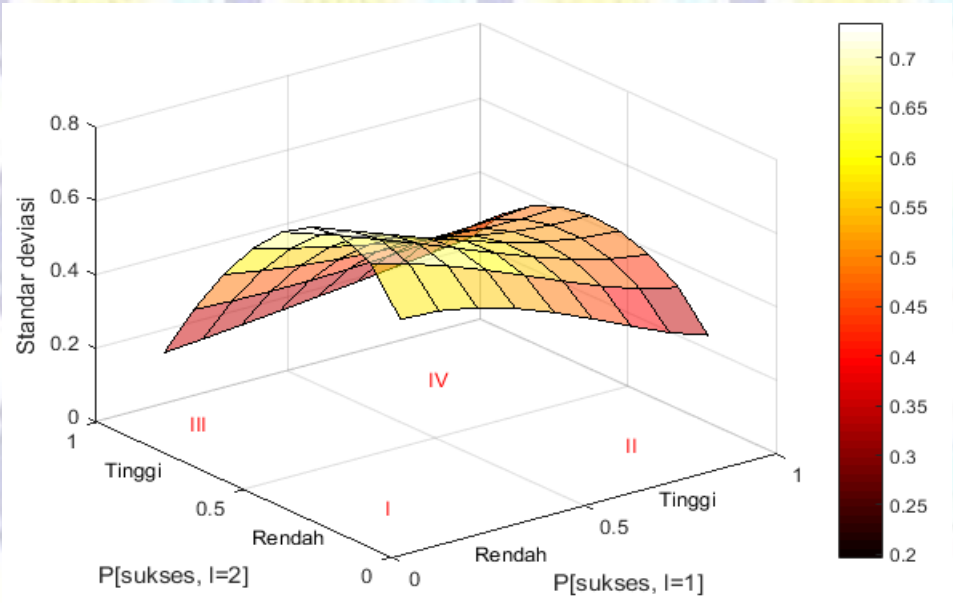
Parameter pertama yang dianalisa adalah jumlah iterasi hash dengan cara perhitungannya yang telah diperlihatkan pada Persamaan (3.3). Plot 3D jumlah iterasi hash dari relasi antara peluang sukses pada setiap kekuatan puzzle dan nilai kekuatan puzzle itu sendiri jika nilai l_{maks} =2 ditunjukkan pada Gambar 3.11. Nilai peluang sukses baik pada kekuatan puzzle 1 atau 2 bit dibagi menjadi 2 daerah yaitu 'Rendah' dan 'Tinggi'. Area dikatakan 'Rendah' jika nilai peluang sukses dibawah 0.5 sedangkan area dikatakan 'Tinggi' jika nilai peluang diatas 0.5 dan tentunya dibawah 1. Oleh karena itu plot 3D tersebut terbagi menjadi 4 area yaitu I, II, III dan IV. Area I ditujukan untuk daerah yang peluang suksesnya 'Rendah' baik pada kekuatan puzzle 1 ataupun 2 bit. Area II menunjukkan daerah yang peluang sukses pada kekuatan puzzle 1 bit bernilai 'Tinggi' sedangkan peluang sukses pada kekuatan puzzle 2 bit 'Rendah'. Sebaliknya, area III ditujukan untuk daerah dengan peluang sukses pada kekuatan puzzle 1 bit 'Rendah' sedangkan peluang pada kekuatan puzzle 2 bit 'Tinggi'. Area IV menunjukkan daerah yang peluang suksesnya 'Tinggi' baik pada kekuatan puzzle 1 maupun 2 bit. Jumlah iterasi hash tertinggi pada DCP terjadi di area IV, sedangkan jumlah iterasi hash terendah terletak di area I. Jumlah iterasi hash di area II lebih kecil daripada area III.



Gambar 3.12 Peluang tidak ditemukannya solusi jika l_{maks} = 2

Parameter kedua yang dianalisa adalah peluang tidak ditemukannya solusi puzzle yang cara perhitungannya telah diperlihatkan pada Persamaan (3.5). Plot 3D peluang tidak ditemukannya solusi dari relasi antara peluang sukses pada setiap kekuatan puzzle dan kekuatan puzzle itu sendiri jika nilai l_{maks} =2 ditunjukkan pada Gambar 3.12. Pembagian area pada jumlah iterasi hash DCP berlaku juga untuk peluang tidak ditemukannya solusi. Peluang tidak ditemukannya solusi pada DCP paling rendah terjadi di area IV, sedangkan yang tertinggi berada di area I. Peluang tidak ditemukannya solusi pada area II dan III bernilai sama dan lebih tinggi dari peluang di area IV akan tetapi lebih rendah dari area I.

Parameter ketiga yang dianalisa adalah standar deviasi cara perhitungannya telah diperlihatkan Persamaan (2.22). Plot 3D standar deviasi dari relasi antara peluang sukses pada setiap kekuatan puzzle dan kekuatan puzzle itu sendiri jika nilai l_{maks} =2 ditunjukkan pada Gambar 3.13. Pembagian area pada jumlah iterasi hash DCP berlaku juga untuk standar deviasi. Standar deviasi paling rendah berada di area IV, sedangkan yang tertinggi terjadi di area I. Standar deviasi pada area II dan III memiliki kisaran nilai yang hampir sama dan lebih rendah dibandingkan area I.



Gambar 3.13 Standar deviasi pada DCP jika l_{maks} = 2

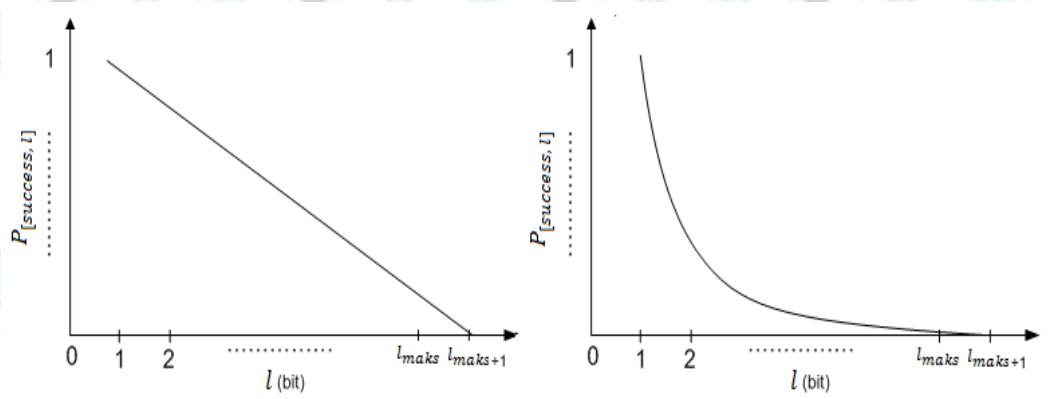
Tabel 3.1 Ringkasan kondisi parameter penentu pada fungsi ambang DCP

Area	Peluang sukses pada kekuatan puzzle		Kondisi optimum parameter	
	l=1	l=2	Terbaik	Terburuk
I	'Rendah'	'Rendah'	n_{DCP}, σ_l	P_{ZS}
II	'Tinggi'	'Rendah'		
III	'Rendah'	'Tinggi'		
IV	'Tinggi'	'Tinggi'	P_{ZS}	n_{DCP}, σ_l

Ringkasan parameter penentu fungsi ambang yang optimum ditunjukkan pada Tabel 3.1. Pada area I, jumlah iterasi hash dan standar deviasi kekuatan puzzle optimum akan tetapi peluang tidak ditemukannya solusi menempati kondisi yang terburuk. Jika memanfaatkan area ini maka fungsi ambang yang terbentuk akan memiliki jumlah iterasi yang rendah dengan nilai kekuatan puzzle yang beragam akan tetapi kemungkinan untuk memiliki kekuatan puzzle 0 bit juga sangat tinggi sehingga area I tidak direkomendasikan. Pada area IV, peluang tidak ditemukannya solusi mencapai nilai optimum akan tetapi jumlah iterasi hash dan standar deviasi dari kekuatan puzzle menempati posisi yang terburuk. Jika memanfaatkan area ini maka fungsi ambang yang terbentuk memiliki kepastian akan solusi puzzle akan tetapi jumlah iterasi tinggi dan standar deviasi kekuatan puzzle menjadi rendah. Area ini menunjukkan kondisi jika kekuatan puzzle bersifat statis dan cenderung mendekati nilai maksimum dari kekuatan puzzle sehingga fungsi ambang menjadi tidak bermanfaat dan area ini menjadi pilihan yang terburuk. Selanjutnya, tersisa dua pilihan yaitu area II dan area III. Nilai dari peluang tidak ditemukannya solusi dan standar deviasi kekuatan puzzle mendekati sama pada kedua area tersebut. Jumlah iterasi hash pada area II lebih rendah dibandingkan area III sehingga fungsi ambang yang akan dibangun akan memenuhi kondisi di area II yang merupakan kondisi ideal antara persilangan ketiga parameter penentu performansi dari DCP. Area II menggambarkan kondisi jika semakin rendah nilai kekuatan puzzle maka semakin tinggi peluang sukses dalam menemukan solusi puzzle. Hipotesa ini didukung dengan hasil percobaan dari penelitian sebelumnya (Afianti et al. 2019). Fungsi ambang dengan cara eksperimental pada penelitian tersebut berasal dari

persamaan eksponensial dengan tingkat kemiringan positif pada relasi antara jumlah iterasi hash dan kekuatan puzzle yaitu semakin kecil kekuatan puzzle semakin besar peluang sukses untuk mendapatkan solusi puzzle. Kondisi tersebut sama dengan hipotesa dari plot 3D untuk l_{maks}=2 yang dibahas sebelumnya. Selain itu, detail nilai yang didapat dari hasil kombinasi peluang sukses ditemukannya solusi pada kekuatan puzzle 1, 2 dan 3 bit untuk ketiga parameter optimasi dapat dilihat pada Lampiran A, B dan C. Semakin tinggi peluang sukses dari kekuatan puzzle 3 bit dengan peluang sukses pada kekuatan puzzle 1 dan 2 bit yang bernilai tetap maka semakin tinggi jumlah iterasi hash dan semakin rendah peluang tidak ditemukan solusi juga standar deviasinya. Jika kombinasi kekuatan puzzlenya diganti (misal: kekuatan puzzle 1 dan 3 bit tetap dan kekuatan puzzle 2 bit semakin meningkat) maka luaran dari parameter optimasi tetap sama yaitu semakin tinggi jumlah iterasi hash dan semakin rendah peluang tidak ditemukan solusi juga standar deviasinya. Kondisi tersebut menunjukkan nilai optimum bagi peluang tidak ditemukannya solusi akan tetapi merupakan kondisi terburuk bagi jumlah iterasi hash dan standar deviasinya. Sebaliknya, semakin rendah peluang sukses dari kekuatan puzzle 3 bit dengan peluang sukses pada kekuatan puzzle 1 dan 2 bit yang bernilai tetap maka semakin rendah jumlah iterasi hash dan semakin tinggi standar deviasi dan peluang tidak ditemukannya solusi (kondisi optimum bagi jumlah iterasi dan standar deviasi namun kondisi terburuk bagi peluang tidak ditemukannya solusi). Kondisi akan tetap sama meskipun kombinasi kekuatan puzzlenya diganti. Relasi yang di dapat masih sama dengan hipotesa yang dibangun berdasarkan hasil plot yang dirangkum pada Tabel 3.1 dan menunjukkan bahwa parameter n_{DCP} dan σ_l berbanding lurus namun berbanding terbalik dengan P_{ZS}. Oleh karena itu, pada penelitian ini akan dibangun persamaan garis dan eksponensial untuk jumlah iterasi hash yang berada di area II pada relasi antara peluang sukses pada setiap kekuatan puzzle dan kekuatan puzzle itu sendiri.

Koordinat titik yang dilewati oleh relasi peluang sukses pada setiap kekuatan puzzle dan kekuatan puzzle harus ditentukan. Gambar 3.14 adalah ilustrasi dari relasi yang akan dibangun dengan sumbu *x* menunjukkan kekuatan



(a) Persamaan garis (b) Persamaan eksponensial

Gambar 3.14 Ilustrasi relasi antara peluang sukses dalam menemukan solusi dan kekuatan puzzle

puzzle yang berkisar antara 0 hingga l_{maks} dan sumbu *y* menunjukkan peluang sukses pada kekuatan puzzle yang berkisar antara 0 hingga 1. Koordinat titik pertama yang dilewati adalah *(x1,y1)* = (l_{maks} +1,0). Titik ini diartikan sebagai tidak mungkinnya kekuatan puzzle l_{maks} +1 untuk berpeluang menghasilkan solusi puzzle. Titik kedua adalah (*x2*,*y2*) = (1,ε) dengan ϵ>0 dan nilainya mendekati 1. Parameter ini digunakan untuk menentukan peluang tertinggi yang mendekati pasti terjadi pada kekuatan puzzle terendah yaitu 1 bit. Penelitian ini menetapkan konstanta $\varepsilon = 1 - 10^{-10}$. Relasi linear dari peluang sukses dengan kekuatan puzzle yang melalui 2 koordinat titik diturunkan pada

$$\frac{y - y1}{y2 - y1} = \frac{x - x1}{x2 - x1}$$

$$\frac{P_{[success,\ l]} - P_{[success,\ l_{maks}+1]}}{P_{[success,\ l_1]} - P_{[success,\ l_{maks}+1]}} = \frac{l - (l_{maks} + 1)}{l_1 - (l_{maks} + 1)}$$

$$\frac{P_{[success,\ l]} - 0}{(1 - 10^{-10}) - 0} = \frac{l - (l_{maks} + 1)}{1 - (l_{maks} + 1)}$$

$$P_{[success,\ l]} = (1 - 10^{-10})^{\frac{[l-(l_{maks}+1)]}{(-l_{maks})}}. \qquad (3.7)$$

relasi eksponensial dari peluang sukses dengan kekuatan puzzle yang diturunkan dari persamaan dasar fungsi eksponensial melalui titik (*x1*,0) dan melalui titik (*x2*,*y2*) = (1,ε) ditunjukkan pada

$$y = a(x - x1)^2$$

$$P_{[success,\ l]} = a\left(l - (l_{maks} + 1)\right)^2$$

$$\left(l_1, P_{[success,\ l]}\right) = (x2, y2) \rightarrow a = \frac{(1 - 10^{-10})}{\left(1 - (l_{maks} + 1)\right)^2}$$

$$a = \frac{(1 - 10^{-10})}{{l_{maks}}^2}$$

$$P_{[success,\ l]} = \frac{(1 - 10^{-10})}{{l_{maks}}^2}\left(l - (l_{maks} + 1)\right)^2. \quad (3.8)$$

Fungsi ambang dengan cara rumus probabilistik merupakan jumlah iterasi hash pada Persamaan (3.2) dengan penyesuaian pada relasi antara peluang sukses dan kekuatan puzzle agar memenuhi Persamaan (3.7) dan (3.8). Oleh karena itu terdapat dua kandidat fungsi ambang dengan rumus probabilistik. Pertama fungsi ambang linear yang ditunjukkan pada

$$n_{linear,l} = \frac{log_{10}\left\{\mathbf{1} - \left[(1 - 10^{-10})\frac{[l-(l_{maks}+1)]}{(-l_{maks})}\right]\right\}}{log_{10}\left\{\mathbf{1} - \left(\frac{\mathbf{1}}{\mathbf{2}}\right)^{\boldsymbol{l}}\right\}}, \quad (3.9)$$

sedangkan fungsi ambang eksponensial ditunjukkan pada

$$n_{kuadratik,l} = \frac{log_{10}\left\{\mathbf{1} - \left[\frac{(1-10^{-10})}{{l_{maks}}^2}\left(l - (l_{maks} + 1)\right)^2\right]\right\}}{log_{10}\left\{\mathbf{1} - \left(\frac{\mathbf{1}}{\mathbf{2}}\right)^{\boldsymbol{l}}\right\}}. \quad (3.10)$$

Proses DCP dilanjutkan dengan komunikasi real-time setelah kandidat fungsi ambang telah didapatkan. Proses ini diawali dengan inisialisasi parameter yang disepakati antara pengirim dan penerima yang terdiri dari maksimum kekuatan puzzle (l_{maks}) dan banyaknya kunci sesi (n_{key}) yang merujuk pada jumlah indeks yang disediakan. Tidak ada nilai khusus pada penentuan maksimum kekuatan puzzle akan tetapi MSP merekomendasikan 22 bit sebagai angka yang sesuai nilai yang paling wajar (Ning & Liu 2008). Penelitian ini meningkatkan nilai rekomendasi tersebut menjadi 24 bit dikarenakan delay sudah dapat dikontrol

Algoritma 3.3 Pembangkitan DCP pada pengirim

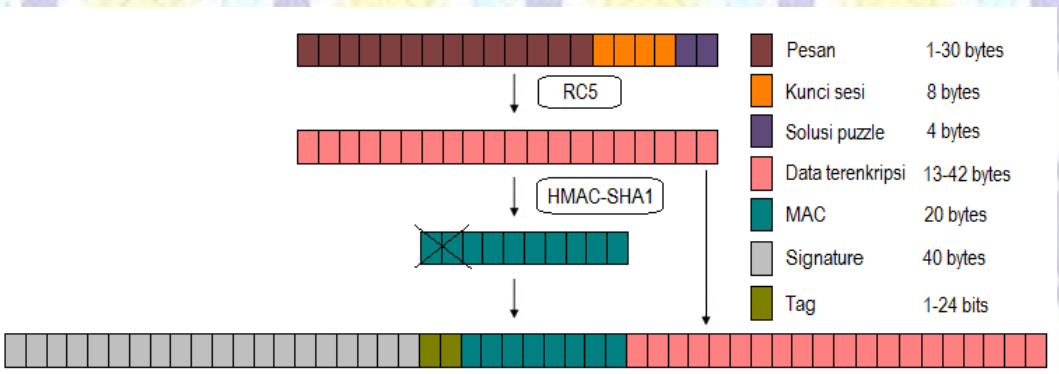
Input: Session key K_{idx}, $index$ dan kekuatan puzzle maksimum L_{max} dari proses inisialisasi sedangkan M masukan dari pengguna.
Output: Pengiriman paket P_{DCP} pada sensor node atau solusi puzzle tidak ditemukan

```
1  Sig ⇐ ECDSA_sign(M)
2  L, L_prev ⇐ L_max
3  iteration ⇐ 0
4  threshold ⇐ Threshold_Function(L)
5  finding ⇐ False
6  while (not(finding)) and (L > 0) do
7      if iteration < threshold then
8          iteration++
9          srand ⇐ genRandom()
10         EncM ⇐ Enc_RC5 (M||srand||K_idx)
11         MAC ⇐ HMAC_SHA1(EncM)
12         if Hash(Sig).substr(0,L) == MAC.substr(0,L) then
13             finding ⇐ True
14         end
15     else
16         L--
17         threshold ⇐ threshold+ Threshold_Function(L)
18     end
19 end
20 if finding then
21     tag ⇐ Hash(index+L_prev+L).substr(0,L)
22     MAC ⇐ MAC.substr(L,Length(MAC)−L)
23     index++
24     Lprev ⇐ L
25     P_DCP ⇐ Sig||tag||MAC||EncM
26     Send ⇐ P_DCP
27 else
28     Zero_Solution
29 end
```

dan maksimum jumlah iterasi hash dapat diminimalisasi. Setelah proses inisialisasi di awal komunikasi, pengirim dapat memproses pesan yang detail pelaksanaannya terlihat pada Algoritma 3.3. Pembangkitkan signature dari pesan yang akan dikirim menjadi pembuka pemrosesan pesan pengguna yang ditunjukkan pada Algoritma 3.3 baris ke 1. Kemudian proses pembuatan solusi puzzle dari pesan, signature dan kunci sesi ditunjukkan pada Algoritma 3.3 baris ke 2 hingga 19. Pada proses ini, kandidat solusi puzzle yang merupakan karakter random digabungkan dengan pesan dan kunci sesi. Hasil penggabungan ini

dienkripsi pada variabel *EncM*. Kemudian dibangkitkanlah *Message Authenticate Code* (MAC). Penelitian ini, menggunakan *Hash Message Authenticate Code* (MAC) yang merupakan bagian spesifik dari MAC dan melibatkan fungsi kriptografi hash dan sebuah kunci sesi (Krawczyk et al. 1997). Proses ini digunakan untuk menjamin integrity dan autentikasi dari pesan. Hasil MAC yang didapat dibandingkan dengan penggalan pola yang didapat dari signature. Proses ini akan diulang hingga salah satu kondisi terpenuhi yaitu ditemukannya solusi puzzle pada kekuatan puzzle tertentu atau tidak ada solusi puzzle yang ditandai dengan kekuatan puzzle yang bernilai 0. Jika solusi puzzle, ditemukan maka dilanjutkan dengan pembentukan tag yang akan dibahas pada Bab 3.2.3. Setelah tag terbentuk, semua informasi yang didapat pada Algoritma 3.3 baris ke 25 akan dikirimkan sesuai dengan formasi pada Gambar 3.15. Payload dari paket yang akan ditransmisikan adalah 102 byte. Tag mengambil bagian atau porsi dari MAC sebanyak *l* bit. Parameter utama yang isinya dilindungi adalah pesan yang panjangnya bervariasi antara 1 hingga 30 byte, kunci sesi 8 byte dan solusi puzzle 4 byte. Hal tersebut berdampak pada pesan yang terenkripsi bervariasi antara 13 hingga 42 byte.

Pada sisi penerima, proses verifikasi pesan yang diterima diawali dengan pemeriksaan nilai tag yang ditunjukkan pada Algoritma 3.4 baris ke 5-11. Detail dari proses ini akan dijelaskan pada Bab 3.2.3. Jika tag telah terverifikasi maka MAC yang diterima harus dibandingkan dengan hasil perhitungan HMAC-SHA1 dari pesan yang diterima dengan kunci



Gambar 3.15 Rincian isi paket skema DCP

Algoritma 3.4 Verifikasi DCP pada penerima

Input: Kunci *commitment* K_0, *index*, L_{prev} telah tersimpan pada node sensor dan menerima paket P_{DCP}

Output: Memastikan paket P_{DCP} terautentikasi dan resistan terhadap serangan DoS berbasis PKC

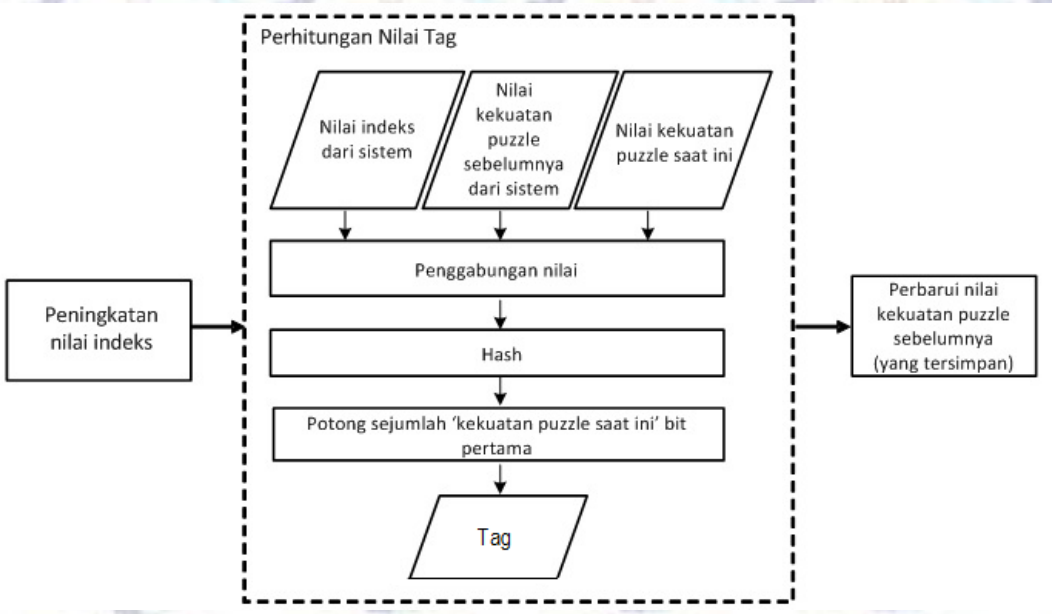
```
1  Accept ⇐ P_DCP
2  MAC ⇐ P_DCP.substr(320, 160)
3  L ⇐ L_max
4  Ver_tag ⇐False
5  while (L > 0) and (not(Ver_tag)) do
6      if (MAC.substr(0, L) == Hash((index + 1) + L_prev + L).substr(0, L) then
7          Ver_tag ⇐ True
8      else
9          L − −
10     end
11 end
12 if Ver_tag then
13     Sig ⇐ P_DCP.substr(0, 320)
14     EncM ⇐ P_DCP.substr(160, 256)
15     MAC ⇐Hash(Sig).substr(0, L)||MAC.substr(L, 160 − L)
16     if MAC == HMAC_SHA1(EncM) then
17         data ⇐ Dec_RC5(EncM)
18         M ⇐ data.substr(0, data.Length() − 64);
19         K_next ⇐ data.substr(data.Length() − 64, 64);
20         if Hash^(index+1)(K_next) == K_0 then
21             if ECDSA_ver(Sig) then
22                 L_prev ⇐ L
23                 index++
24                 Packet_Verified
25             else
26                 Drop_Message_Bad_Signature
27             end
28         else
29             Drop_Message_sessionKey_invalid
30         end
31     else
32         Drop_Message_puzzleSolution_invalid
33     end
34 else
35     Drop_Message_tag_invalid
36 end
```

sesi yang disimpan oleh node sensor. Proses akan dilanjutkan dengan verifikasi kunci sesi jika MAC yang diterima tervalidasi. Kunci sesi dibangun dengan one-

way key chain (Lamport 1981). Penelitian ini memanfaatkan fungsi hash berbasis kriptografi SHA-1 sebagai algoritma utama pembentuknya. Kunci sesi yang diterima dan indeks yang didapatkan dari verifikasi tag dibandingkan dengan kunci commitment yang disimpan pada node sensor. Proses akan dilanjutkan pada verifikasi algoritma signature utama jika kunci sesi tervalidasi. Verifikasi ECDSA pada node sensor memiliki kompleksitas yang paling tinggi oleh karena itulah dibangunlah berbagai filter pendamping.

3.2.3. Tag pada skema puzzle dinamis

Tag dibangun dengan tujuan utama untuk mengirimkan parameter pendukung secara implisit dan hemat biaya komunikasi. Parameter pendukung yang akan dikirimkan antara lain kekuatan puzzle dari solusi puzzle yang dikirim dan indeks. Pada penelitian sebelumnya (Aura et al. 2000; Ning & Liu 2008; Tan et al. 2013), nilai indeks dikirim secara terbuka (eksplisit) pada pesan sedangkan nilai kekuatan puzzle bersifat statis sehingga cukup disimpan di ruang penyimpanan pada pengirim dan penerima. Tag mengaburkan nilai dari parameter pendukung tersebut pada fungsi hash sehingga tidak mudah ditebak oleh penyerang.



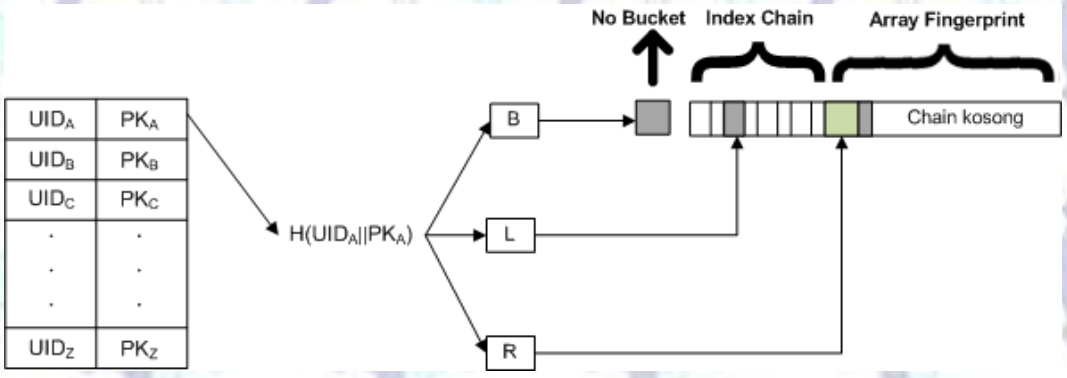
Gambar 3.16 Perhitungan nilai tag

Perhitungan nilai tag digambarkan secara rinci pada Gambar 3.16. Proses ini diawali dengan naiknya nilai indeks karena ada pesan yang akan dikirimkan dari pengguna pada node sensor. Nilai indeks digabungkan dengan kekuatan puzzle terakhir yang dikirim pengguna dan kekuatan puzzle dari pesan yang akan dikirim. Hasil dari penggabungan tersebut digunakan sebagai masukan pada fungsi hash (penelitian ini menggunakan SHA-1). Fungsi hash merupakan salah satu fungsi hash kriptografi generasi pertama yang banyak digunakan dengan ukuran luarannya sebesar 160 bit. Luaran dari fungsi ini dipotong untuk menghemat biaya komunikasi yaitu sebesar panjang kekuatan puzzle dari pesan yang dikirim. Proses ini diperlihatkan pada baris ke-19 pada Algoritma 3.1 untuk DMSP dan baris ke-21 pada Algoritma 3.3 untuk DCP.

Node sensor sebagai pihak yang akan memverifikasi tag, menyimpan nilai dari kekuatan puzzle yang tervalidasi sebelumnya, kekuatan puzzle maksimum dan indeks dari kunci sesi. Proses verifikasi ini diawali dengan mengekstraksi bagian tag sesuai dengan baris ke 2 – 3 pada Algoritma 3.2 untuk DMSP dan baris 2 pada Algoritma 3.4 untuk DCP. Node sensor harus mencoba semua kemungkinan nilai kekuatan puzzle pada pesan yang dikirim (l). Nilai pertama yang dicobakan adalah nilai kekuatan puzzle maksimumnya (l_{maks}) kemudian turun hingga menuju ke angka 1 yang merupakan kekuatan puzzle terendah. Jika tidak sesuai dengan potongan tag yang diterima maka bisa dipastikan bahwa pesan yang diterima palsu. Proses ini ditunjukkan pada baris ke 6 – 12 pada Algoritma 3.2 untuk DMSP dan baris ke 5-11 pada Algoritma 3.4 untuk DCP.

3.3. Skema keanggotaan TinySet pada JSN

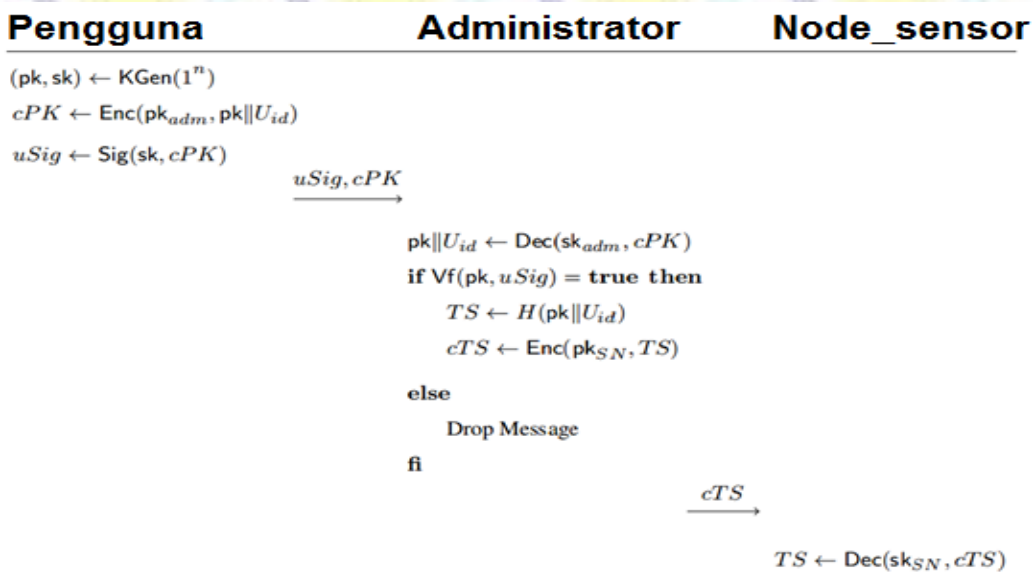
TinySet diusulkan untuk mengatasi jumlah pengguna yang banyak dan akan mengakses JSN. Jika jumlahnya masih terbatas maka informasi pengguna bisa disimpan langsung pada JSN akan tetapi jika jumlahnya besar maka ruang penyimpanan pada JSN memiliki keterbatasan. TinySet merupakan skema keanggotaan yang hemat ruang penyimpanan. Nilai unik dari pengguna JSN terdiri dari kunci publik dan identitas pengguna sehingga skema TinySet yang



Gambar 3.17 TinySet pada JSN

diajukan diilustrasikan pada Gambar 3.17. Posisi bucket, indeks chain dan nilai fingerprint didapatkan dari hasil operasi hash pada gabungan identitas pengguna dan kunci publiknya.

Tugas dari penambahan dan penghapusan pengguna terpusat pada bagian Administrator. Sedangkan pengirim dan penerima dalam penelitian ini yaitu pengguna dan JSN hanya bisa menjalankan proses query. Selain mengatur dan mendistribusikan TinySet, Administrator juga mengontrol kunci sesi yang akan digunakan oleh pengguna. Protokol komunikasi pembentukan TinySet dapat dilihat pada Gambar 3.18.



Gambar 3.18 Protokol komunikasi pengaturan TinySet pada JSN

Pengguna yang akan berkomunikasi dengan JSN mendaftarkan identitas dan kunci publiknya pada Administrator menggunakan autentikasi pengguna sederhana dengan memanfaatkan proses enkripsi dan autentikasi seperti Gambar 3.18. Administrator men-decode pesan yang diterima. Hasil yang diperoleh merupakan parameter identitas pengguna dan kunci publik. Administrator memastikan pengguna merupakan pihak yang berhak mengakses JSN melalui verifikasi signature. Jika valid maka Administrator akan menambahkan identitas pengguna pada TinySet kemudian mengirimkannya pada JSN. Selain itu pengguna akan mendapatkan kunci sesi sebagai pertanda pengguna dapat segera berkomunikasi dengan JSN.

3.3.1. Regularisasi TinySet

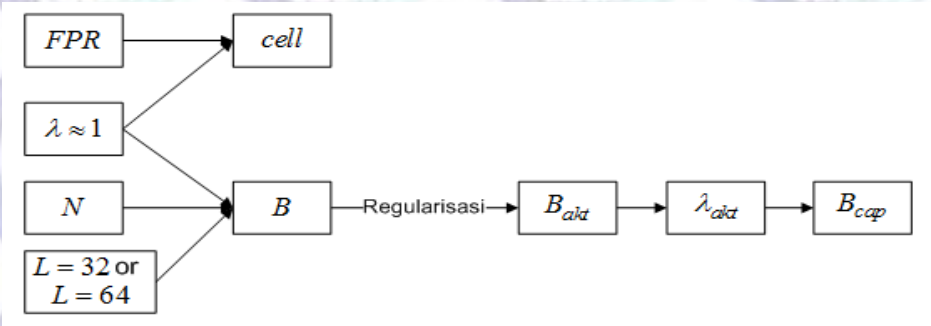
Pada tahap inisialisasi dalam pembentukan TinySet beberapa parameter harus dideklarasikan dan bersifat statis. Parameter-parameter ini menyesuaikan dengan maksimum jumlah pengguna yang akan berkomunikasi dengan JSN dan besarnya false positif yang dapat diterima sebagai ukurun tingkat keamanan dari skema keanggotaan. Sedangkan pada TinySet parameter yang harus ditentukan yaitu jumlah bucket, ukuran indeks, kapasitas bucket dan ukuran fingerprint. Semakin tinggi nilai dari inisialisasi parameter maka semakin tinggi tingkat keamanan akan tetapi ruang penyimpanan menjadi semakin besar. Oleh karena itu, diperlukan skema yang dapat menyesuaikan nilai harapan dari Administrator dengan tingkat keamanan yang tinggi akan tetapi ruang penyimpanan serendah mungkin.

Parameter penentu utama pada TinySet adalah rata-rata jumlah pengguna pada setiap bucket yang akan digunakan atau disimbolkan dengan λ. Nilai dari parameter ini direkomendasikan dibawah satu (Einziger & Friedman 2017). TinySet menghindari penggunaan linked list karena tingginya komputasi, rank indek disusun dengan penelusuran array berbasis indeks sebagai penggantinya. Jika jumlah anggota pada sebuah indeks chain melebihi satu, maka kompleksitas penelusuran akan meningkat dan waktu query pun akan meningkat. Penelusuran atau pencarian fingerprint pada setiap indeks dilakukan dengan

dengan metode pencocokan sederhana yaitu fingerprint tujuan harus dicocokkan satu persatu mulai dari fingerprint pertama pada indeks chain tersebut. Berdasarkan Persamaan (2.18), parameter λ bergantung pada jumlah bucket, panjang indeks dan maksimum jumlah pengguna yang dapat ditampung pada TinySet. Nilai λ diharapkan mendekati satu sehingga setiap indeks pada chain rata-rata memiliki 1 anggota. Jika diketahui nilai $\lambda = 1$ dan Administrator hendak menentukan jumlah bucket yang dibutuhkan untuk memenuhi sejumlah anggota maka diperlukan proses pembulatan. Hal ini dikarenakan jumlah bucket merupakan bilangan integer positif. Pembulatan ini sangat berpengaruh pada TinySet, karena penambahan sebuah bucket diartikan dengan penambahan sejumlah array indeks dengan kapasitas bucket tertentu sedangkan pengurangan sebuah bucket berpengaruh pada proses pencarian yang semakin lama. Penelitian ini mengusulkan proses regularisasi bucket pada TinySet sesuai dengan

$$B_{akt} = \begin{cases} \left\lfloor \frac{N}{\lambda.L} \right\rfloor, & \text{jika } \left| 1 - \frac{N}{\left\lfloor \frac{N}{\lambda.L} \right\rfloor . L} \right| \leq \left| 1 - \frac{N}{\left\lceil \frac{N}{\lambda.L} \right\rceil . L} \right| \\ \left\lceil \frac{N}{\lambda.L} \right\rceil, & \text{jika } \left| 1 - \frac{N}{\left\lfloor \frac{N}{\lambda.L} \right\rfloor . L} \right| > \left| 1 - \frac{N}{\left\lceil \frac{N}{\lambda.L} \right\rceil . L} \right| \end{cases} \quad (3.11)$$

Nilai bucket akan dibulatkan ke bawah jika dari hasil pembulatan tersebut menghasilkan nilai λ_{akt} yang lebih mendekati satu dibandingkan dengan jika menggunakan pembulatan keatas. Sebaliknya, nilai bucket akan dibulatkan ke atas jika dari hasil pembulatan tersebut menghasilkan nilai λ_{akt} yang lebih mendekati satu dibandingkan dengan jika menggunakan pembulatan kebawah.



Gambar 3.19 Proses inisialisasi pada TinySet

Proses inisialisasi pada Administrator dirangkum pada Gambar 3.19. Informasi penting yang perlu ditentutkan di tahap inisialisasi hanya jumlah anggota, sedangkan jumlah indeks akan direkomendasikan di akhir penelitian berdasarkan hasil percobaan. Jumlah bucket ditentukan berdasarkan hasil regularisasi dan nilai λ_{akt} dihitung untuk menentukan nilai kapasitas bucket aktual (B_{cap}) sesuai dengan

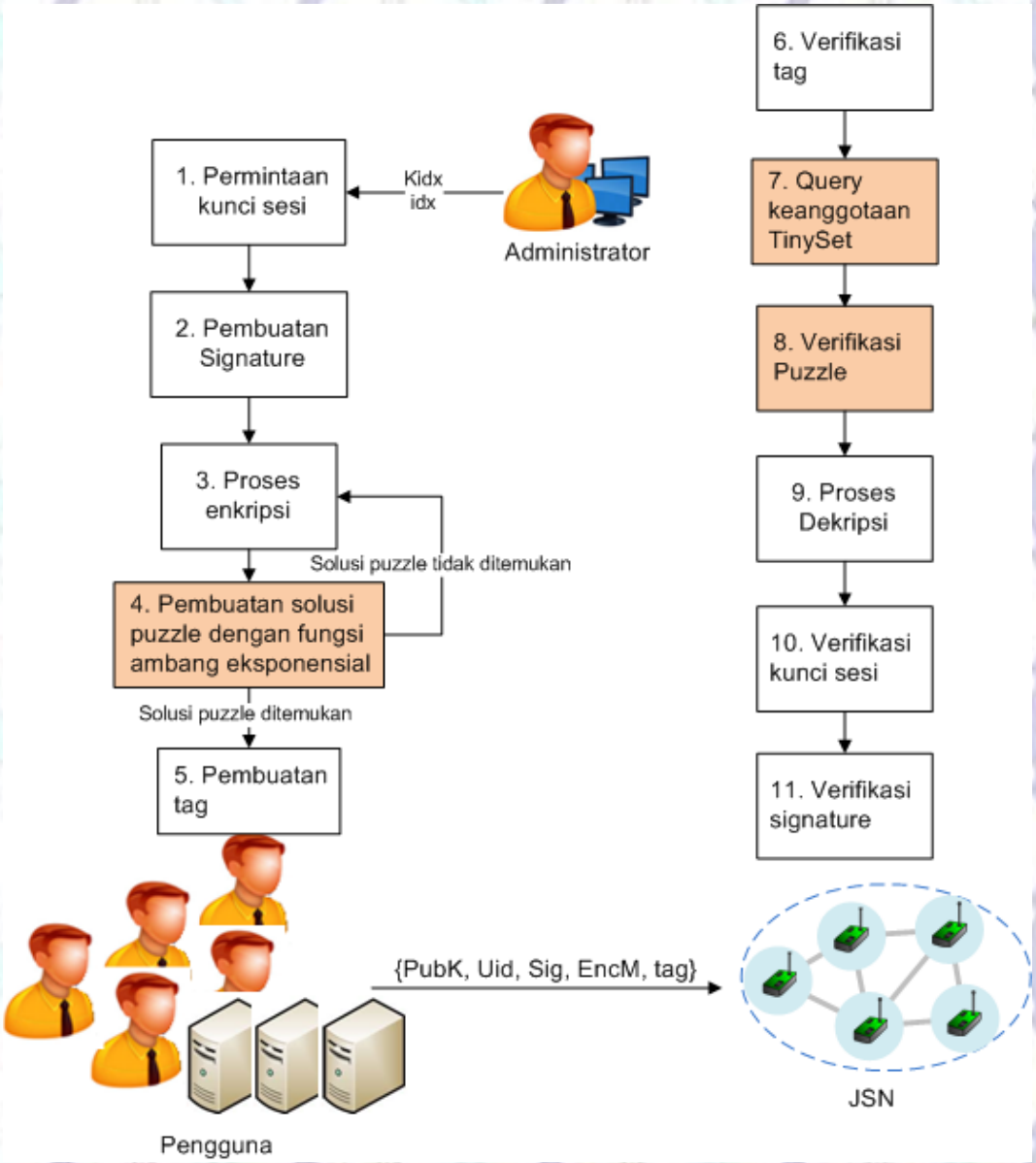
$$B_{cap} = \lceil L \times \lambda_{akt} \rceil. \tag{3.12}$$

Kapasitas bucket inilah yang mempengaruhi ruang penyimpanan sesuai dengan Persamaan (2.23). Semakin tinggi kapasitas bucket maka semakin besar ruang penyimpanan yang dibutuhkan JSN.

3.3.2. Multiuser-Dynamic Cipher Puzzle (M-DCP) dengan TinySet

Komunikasi antara pengguna dan JSN bisa dijalankan setelah pengguna mendaftarkan diri pada Administrator, TinySet didistribusikan oleh Administrator dan JSN telah menerima TinySet yang valid dari Administrator. Skema M-DCP dengan TinySet diusulkan untuk menghadapi serangan DoS berbasis PKC dengan jaminan autentikasi dan confidentiality. Detail proses dari M-DCP ditunjukkan pada Gambar 3.20. Pengguna memulai komunikasi dengan meminta kunci sesi dan indeks pada Administrator. Setelah mendapat jawaban, signature dapat dibentuk dari pesan yang akan dikirim menggunakan kunci rahasia dari tiap-tiap pengguna. Metode yang digunakan pada proses ini adalah ECDSA dengan *partial recovery* yang blok diagramnya telah dijelaskan pada Gambar 2.15 (Ren et al. 2009). Pada proses ini pesan dibagi menjadi 2 bagian yaitu M_1 dengan ukuran dibawah 10 byte yang disisipkan pada signature dan sisa pesannya ditampung pada M_2. Proses ini dilanjutkan dengan mengenkripsi pesan menggunakan meotde RC5 yang detail programnya tercantum pada Lampiran E. Parameter-parameter yang dienkripsi tercantum pada

$$EncM = \mathrm{RC5}_{Kidx-1}(M_2|K_{idx}|P_{idx}), \tag{3.13}$$



Gambar 3.20 Skema M-DCP

yaitu pesan bagian kedua M_2, kunci sesi K_{idx}, dan kandidat solusi puzzle P_{idx}. Langkah selanjutnya adalah pembentukan solusi puzzle dengan fungsi ambang menggunakan metode eksponensial dari proses DCP telah diusulkan pada Bab 3.2.2. Solusi puzzle yang dibentuk harus memenuhi pola sesuai yang tercantum pada

$$H_1(EncM)_{\{l\}} = H_2(Sig)_{\{l\}}. \quad (3.14)$$

Solusi puzzle dibentuk dari 2 fungsi hash yang berbeda. Penelitian ini menggunakan SHA1 dan MD5 yang detailnya tercantum pada Lampiran E. Jika puzzle ditemukan maka proses M-DCP pada sisi pengirim diakhiri dengan pembuatan tag. Pola pada pengguna jamak M-DCP (multiuser) berbeda dengan pola pengguna terbatas DCP. Hal ini dikarenakan pada M-DCP, proses MAC dihapus dan ukuran tag menjadi tetap. Penghapusan MAC dimaksudkan untuk menyesuaikan biaya komunikasi agar tetap memenuhi maksimum payload 102 byte, seperti diketahui kunci publik pada pengguna jamak tidak disimpan pada sisi penerima melainkan harus dikirim bersama pesan. Oleh karena itu tag tidak lagi bisa diintegrasikan pada MAC dan harus berdiri sendiri. Ukuran tag didefinisikan sebesar 3 byte sesuai dengan maksimum kekuatan puzzle yaitu 24 bit. Dikarenakan proses pengaturan pesan diatur oleh Administrator maka komposisi tag pun diubah sesuai dengan

$$tag = H(idx|l_{maks}|l), \tag{3.15}$$

dengan kekuatan puzzle pada proses sebelumnya (l_{prev}) digantikan oleh maksimum kekuatan puzzle (l_{maks}).

Setelah semua rangkaian proses di sisi penerima dilalui, parameter – parameter akan dikirimkan sesuai dengan penjelasan pada Tabel 3.2. Proses verifikasi di sisi penerima atau JSN diawali dengan pengecekan nilai tag. Parameter yang sudah tersimpan di JSN yaitu indeks dan maksimum kekuatan puzzle. Node sensor harus mencoba dari indeks tertinggi digabungkan dengan indeks dan maksimum kekuatan puzzle dan dijalankan proses hash. Apakah hasilnya akan sama dengan tag yang diterima. Jika ya, maka proses dilanjutkan dengan verifikasi keanggotaan kunci publik pada TinySet. Detail dari proses ini telah dijelaskan pada Bab 2.5. Jika fingerprint dari idetitas pengguna dan kunci publik tercantum pada TinySet maka proses dilanjutkan dengan verifikasi puzzle menggunakan pola yang sama pada pengirim sesuai Persamaan (3.14). Jika pola puzzle dari parameter yang diterima menggunakan kunci sesi yang tersimpan pada JSN sesuai maka proses verifikasi dilanjutkan dengan tahap dekripsi dan memeriksa apakah kunci sesi hasil dekripsi sesuai dengan kunci sesi selanjutnya

Tabel 3.2 Rincian isi paket M-DCP

Parameter	Ukuran (byte)
PubK	20
Uid	2
Sig	40
M_2	1-25
Kidx	8
Pidx	4
Tag	3

sesuai dengan indeks yang dikirim. Jika kunci sesi valid maka dilanjutkan pada proses terakhir dari verifikasi pengguna yaitu kevalidan signature. Proses ini memiliki tingkat keamanan yang tinggi diikuti dengan tingginya komputasi pada tahap pembuatan maupun verifikasi.

Berikutnya akan dijelaskan mengenai capaian dari penelitian menggunakan skema yang diajukan. Penjelasan rinci dari capaian ini dibagi menjadi 3 bagian utama yaitu skema Dynamic Message Specific Puzzle menggunakan fungsi ambang eksperimental, Dynamic Cipher Puzzle menggunakan fungsi ambang dengan rumus probabilistik dan Multiuser-Dynamic Cipher Puzzle yang dilengkapi dengan TinySet. Skenario, hasil, pembahasan dan analisa keamanan dari masing-masing skema tersebut akan dijelaskan pada Bab 4, 5 dan 6 secara berurutan.

BAB 4
Dynamic Message Specific Puzzle (DMSP) dengan cara eksperimental

Skema DMSP menggunakan fungsi ambang eksperimental merupakan percobaan awal dalam mendalami metode puzzle dinamis. Tujuan utama dari penggunaan skema ini adalah menurunkan delay di sisi pengirim yang ditandai dengan menurunnya jumlah iterasi hash pada saat pembentukan solusi puzzle. Selain mempertahankan keamanan, performansi khususnya di sisi pengirim menjadi lebih baik.

4.1. Skenario uji coba DMSP

Percobaan diawali dengan mengujikan panjang pesan yang berbeda-beda. Pada penelitian ini digunakan panjang pesan dengan kelipatan 5 dan 10 dengan maksimum 48 byte sesuai dengan payload yang disediakan oleh JSN. Selain itu variasi tersebut telah memenuhi semua kemungkinan blok yang akan terisi pada fungsi hash. Lima varian panjang pesan yang diujikan dijelaskan pada Tabel 4.1. Setiap percobaan varian panjang pesan diulang sebanyak 60 kali dengan memanfaatkan kandidat fungsi ambang. Jumlah pengulangan tersebut dua kali lebih tinggi dari sample kecil yang rata-rata bernilai 30. Percobaan dengan variasi panjang ini diujikan dengan tujuan untuk mengetahui dampak pesan yang dikirim terhadap parameter performansi. Pada skema cipher puzzle, hal ini tidak berpengaruh karena mengirimkan enkripsi dari hashing program image yang panjangnya selalu tetap. Sedangkan hal berbeda terjadi pada skema MSP, yaitu pesan yang dikirim adalah plaintext tanpa enkripsi. Sedangkan kekuatan puzzle

Tabel 4.1 Variasi isi dan panjang pesan yang dikirim

No	Isi pesan	Ukuran (byte)
1	silent	6
2	so shutdown	11
3	save key delete repair	22
4	recovery share send add max power	33
5	wireless sensor network stop start shutdown play	48

maksimum yang digunakan adalah 22. Hal ini, sesuai dengan delay yang masih dapat diterima pada rata – rata iterasi hash (Ning & Liu 2008). Berbagai nilai yang ditetapkan ini akan dipasangkan dengan fungsi ambang yang terbentuk.

Tabel 4.2 menunjukkan hasil dari mekanisme eksperimental pada kekuatan puzzle 10 bit sebelum diolah. Pengolahan data pertama yang dapat dijalankan adalah pengurutan secara menaik untuk mengetahui nilai kuartil dari tiap panjang pesan.

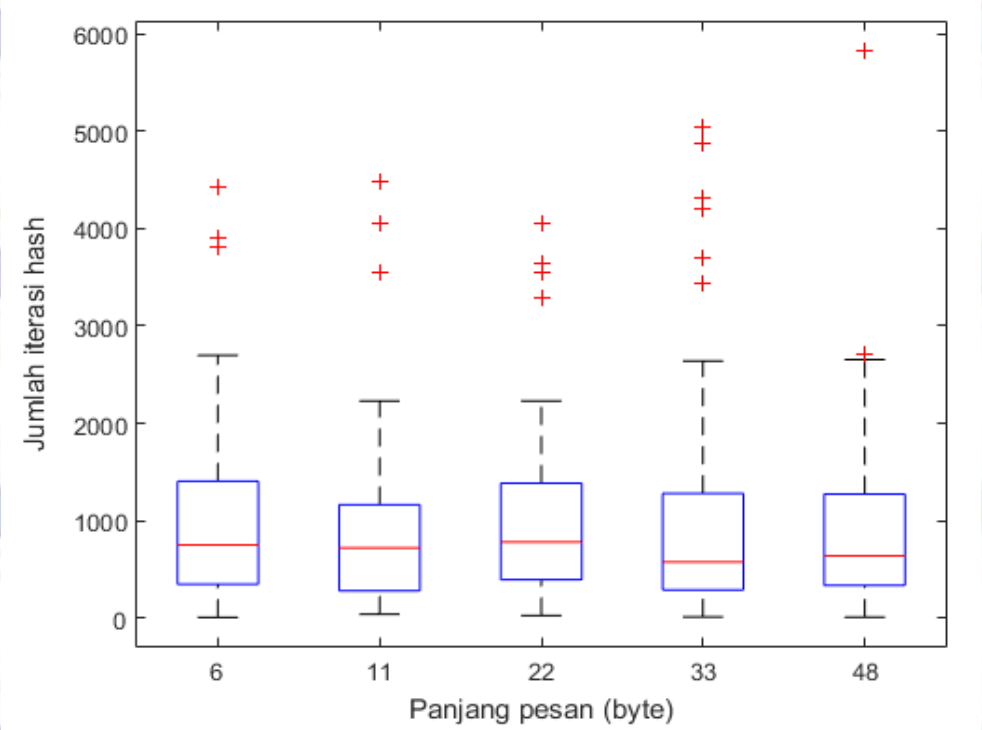
Tabel 4.2 Hasil implementasi MSP pada *l*=10 bit

l=10	Jumlah iterasi hash pada panjang pesan				
Percobaan ke-	6	11	22	33	48
1	220	46	1087	735	414
2	345	709	73	4202	641
3	1066	104	1700	851	1202
4	70	4489	784	174	352
5	664	859	1329	571	273
6	1183	1825	1281	930	1729
7	2112	1037	780	16	948
8	1402	1691	3296	460	1236
9	1007	62	165	191	546
10	1040	545	697	328	401
11	726	1230	524	119	12
12	1127	619	252	4870	1008
13	498	1803	356	795	636
14	2513	204	3285	1040	2533
15	3908	729	422	3690	144
16	1237	322	3637	4313	2516
17	175	539	355	359	96
18	541	491	1298	1338	325
19	23	40	435	247	888
20	637	241	2166	666	255
21	552	266	97	3445	1434
22	364	1217	2086	329	511
23	1681	319	1798	218	1287
24	551	175	606	555	2654
25	759	291	671	27	624
26	931	977	1369	12	459
27	1837	943	565	206	1189

Lanjutan Tabel 4.2

l=10	Jumlah iterasi hash pada panjang pesan				
Percobaan ke-	6	11	22	33	48
28	1336	1023	703	310	599
29	3807	197	25	577	1258
30	749	322	696	1094	1321
31	2697	849	849	270	1141
32	1409	952	952	247	5822
33	27	184	184	572	811
34	1514	424	424	523	2463
35	193	783	783	284	714
36	143	1334	1334	1399	736
37	1641	546	546	543	9
38	347	1110	1110	5032	604
39	421	1398	1398	1848	342
40	751	409	409	1760	30
41	661	177	177	1792	119
42	1941	1079	1079	293	418
43	34	754	754	1642	221
44	221	215	215	920	637
45	1070	1766	1766	232	277
46	1085	1993	1993	696	8
47	1530	4059	4059	2639	2027
48	969	845	845	522	619
49	4420	818	818	1295	1592
50	6	1517	1517	1958	631
51	145	682	682	184	108
52	1355	2229	2229	319	160
53	162	172	172	1163	328
54	572	328	328	436	1653
55	599	1510	1510	294	2715
56	2636	260	260	1265	825
57	372	40	40	635	694
58	257	3552	3552	597	1744
59	138	960	960	740	2044
60	1874	378	378	284	1242

Distribusi data pada tiap panjang pesan digambarkan pada boxplot yang merupakan rangkuman dari hasil percobaan pada kekuatan puzzle 10 bit dan dapat dilihat pada Gambar 4.1. Dari kelima kandidat kuartil yang diperoleh dari



Gambar 4.1 Distribusi data pada MSP untuk l=10

variasi panjang pesan, kami memilih nilai paling maksimum untuk mewakili nilai ambangnya. Sebagai contoh pada kekuatan puzzle 10 bit, ambang batas untuk Q1 dan Q2 bernilai 394 dan 782 yang didapat dari panjang pesan 22 byte sedangkan nilai Q3 yaitu 1406 yang didapat dari panjang pesan 6 byte. Nilai-nilai tersebut dipilih karena merupakan nilai tertinggi dari kelima variasi panjang pesan yang diujicobakan.

Proses eksperimental ini dijalankan dari kekuatan puzzle minimum hingga maksimum (l=1 hingga 22 bit). Hasil yang didapat dari proses eksperimental untuk MSP diurutkan dan diambil perwakilan tiap kuartil. Dikarenakan terdapat 5 variasi panjang pesan maka didapat 5 variasi kuartil, barulah diambil nilai kuartil maksimum dari kelima kandidat nilai kuartil. Rekapitulasi nilai kuartil dengan variasi kekuatan puzzle dapat dilihat pada Tabel 4.3. Nilai inilah yang dijadikan nilai aktual dan didekati dengan proses fitting. Kedekatan antara nilai aktual dan nilai prediksi diukur menggunakan 2 parameter yaitu *RMSE* dan R^2 Kandidat fungsi ambang dari hasil fitting bisa dilihat pada Tabel 4.4.

Tabel 4.3 Nilai aktual iterasi hash pada Q1, Q2 dan Q3

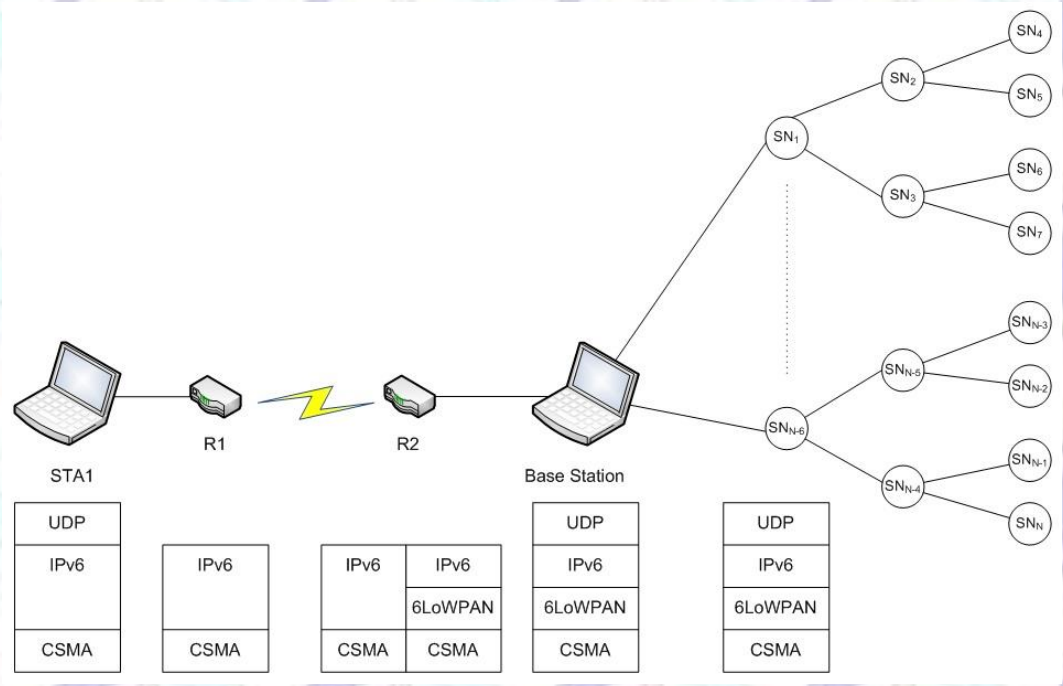
l	Jumlah iterasi hash pada		
	Q1	Q2	Q3
1	2	3	4
2	3	4	7
3	4	7	12
4	7	14	24
5	16	30	50
6	22	49	77
7	43	107	195
8	91	208	412
9	177	413	867
10	394	782	1406
11	791	1670	2815
12	1617	3594	6034
13	3530	6338	11517
14	7725	14370	29209
15	9405	24009	47757
16	24563	71840	113391
17	55821	118004	208997
18	67714	184711	356798
19	179444	377668	739494
20	245187	870212	1300049
21	675715	1537042	3018582
22	1027228	2243569	5706969

Terdapat 9 kandidat fungsi ambang dari metode eksperimental. Fungsi ini didapatkan dari proses fitting menggunakan MATLAB 2015a yaitu toolbox curve fitting. Fungsi yang dipilih adalah power dan eksponensial (EXP). Berdasarkan pemanfaatan nilai kekuatan puzzle, fungsi power sendiri dibagi menjadi dua jenis yaitu *l* yang digunakan sebagai eksponen (POWER1) ataupun basis (POWER2). Kandidat fungsi ambang ini diambil berdasarkan posisi teratas yang paling mendekati nilai aktual. Hal ini terlihat dari nilai R^2 medekati 1 dan nilai *RMSE* yang rendah.

Tabel 4.4 Fungsi ambang metode eksperimental

Fungsi ambang	R^2	*RMSE*
Q1power1 = $\lceil 1.8786^{l} + 1726 \rceil$	0.987	2.81×10^4
Q1power2 = $\lceil l^{12.4} \times 2.41 \times 10^{-11} \rceil$	0.989	2.59×10^4
Q1exp = $\lceil e^{(l \times 0.599)} \times 1.966 \rceil$	0.988	2.72×10^4
Q2power1= $\lceil 1.95^{l} + (2.37 \times 10^4) \rceil$	0.976	8.73×10^4
Q2power2 = $\lceil l^{10.78} \times 7.75 \times 10^{-9} \rceil$	0.994	4.41×10^4
Q2exp = $\lceil e^{(l \times 0.528)} \times 20.86 \rceil$	0.991	5.31×10^4
Q3power1 = $\lceil 2.018^{l} + (6.45 \times 10^4) \rceil$	0.986	1.55×10^4
Q3power2 = $\lceil l^{14.35} \times 3.15 \times 10^{-13} \rceil$	0.999	4.66×10^4
Q3exp = $\lceil e^{(l \times 0.688)} \times 1.53 \rceil$	0.999	4.41×10^4

Analisa lebih mendalam diperlukan dalam mencari fungsi ambang terbaik. Kandidat fungsi ambang ini harus dimplementasikan secara langsung pada komunikasi antara pengguna dan JSN melalui simulasi. Percobaan menggunakan Network Simulator-3 (NS-3) versi 3.20 khususnya modul mengenai JSN. Autentikasi kriptografi yang digunakan pada percobaan ini adalah ECDSA. Algoritma enkripsi yang digunakan adalah RC5. Metode ini dipilih karena implementasinya dianggap paling sesuai dan cocok untuk JSN. Sedangkan, parameter pada RC5 yang akan digunakan yaitu panjang word 32 bit, jumlah round 12, dan panjang kunci 8 byte. Parameter tersebut paling sering digunakan khususnya untuk panjang kunci 8 byte. Mengingat kunci sesi yang digunakan juga untuk enkripsi RC5 memiliki panjang 8 byte. Fungsi hash yang digunakan adalah SHA1. Detail mengenai fungsi hash yang digunakan dijelaskan pada Lampiran F. Implementasi dari percobaan ini menggunakan workstation dengan kemampuan processor Intel(R) Core(TM) i3-2310M CPU @ 2.10GHz - 4CPUs dan memory sebesar 3087 MB DDR3.

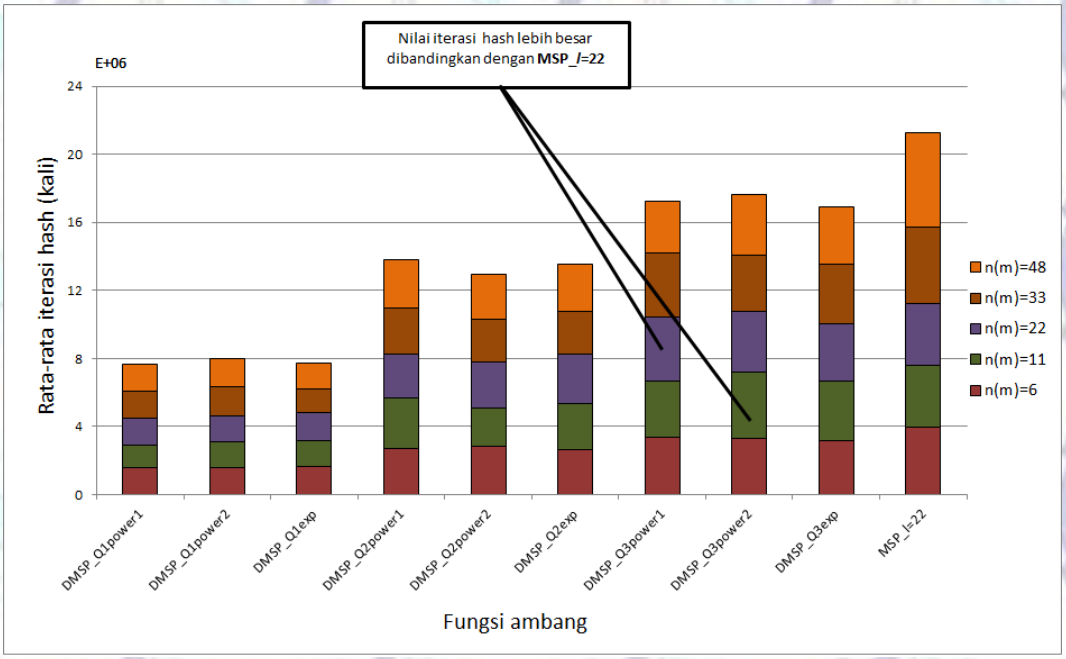


Gambar 4.2 Topologi jaringan simulasi DMSP

Topologi jaringan untuk implementasi metode DMSP menggunakan kandidat fungsi ambang metode eksperimental sesuai dengan Gambar 4.2. Pengirim merupakan workstation dengan nama STA1 sedangkan router difungsikan sebagai penghubung yang terdiri dari R1 dan R2. Pada sisi penerima pesan pertama diterima oleh base station yang diteruskan pada kumpulan node sensor yang terbagi menjadi 3 level. Level pertama, kedua dan ketiga secara berurutan terdiri dari 3, 2 dan 2 node. Tidak ada nilai dan alasan khusus dalam pemilihan baik level maupun jumlah node tersebut.

4.2. Hasil dan pembahasan DMSP

Performansi dan keamanan merupakan dua hal yang saling melengkapi tetapi dampaknya bertolak belakang. Pada perangkat dengan sumber daya terbatas, jika terdapat jaminan yang tinggi terhadap keamanan maka performansi baik dalam segi komputasi, komunikasi maupun kapasitas penyimpanan harus dikorbankan (Pei et al. 2018). Fungsi ambang dibangun agar penurunan performansi berdampak minimum sehingga percobaan bertujuan untuk optimasi parameter rata-rata iterasi fungsi hash, peluang ditemukannya solusi, dan MAD.

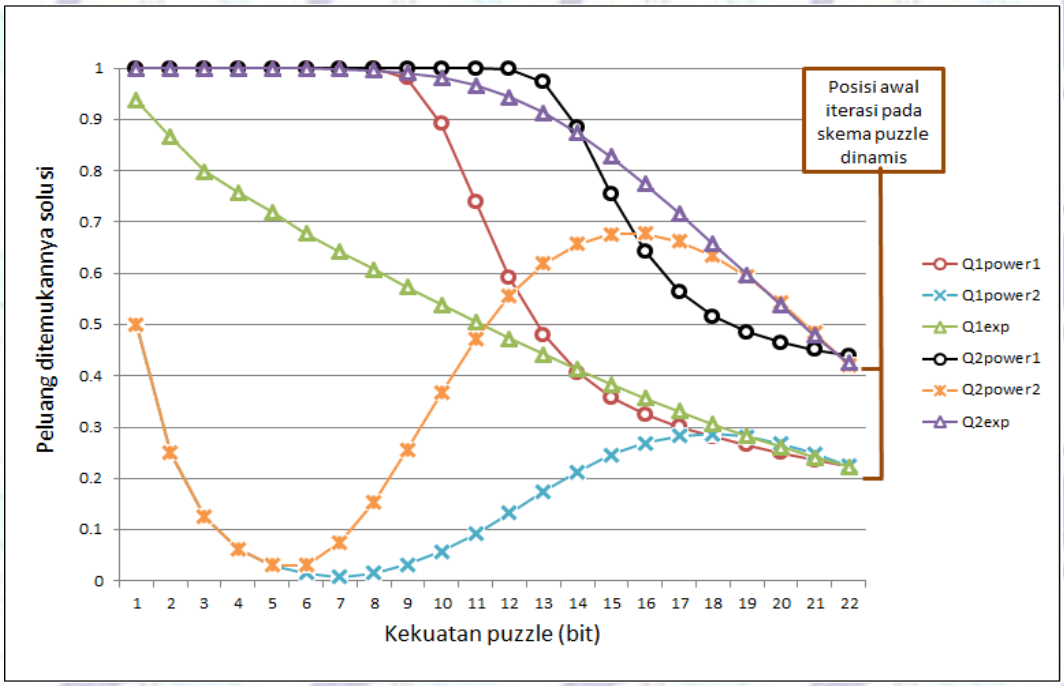


Gambar 4.3 Rata-rata iterasi hash pada DMSP dan MSP

Delay pada sisi pengirim didekati dengan menghitung rata-rata dari jumlah iterasi hash yang dijalankan. Masing-masing fungsi ambang dibagi berdasarkan kuartil kemudian dimanfaatkan pada setiap varian panjang pesan. Pada skema DMSP tidak ada mekanisme enkripsi, pesan yang dikirim berupa plaintext. Hal tersebut tidak membuat delay pada skema MSP menurun, karena ada faktor lain yang lebih berpengaruh yaitu jumlah iterasi hash dalam mencari solusi. Pada percobaan ini ukuran paket yaitu 60, 65, 76, 87 dan 102 bytes yang berasal dari panjang pesan 6, 11, 22, 33 dan 48 berturut-turut. Hasil yang didapat ditunjukkan pada Gambar 4.3. Diagram batang tersebut menunjukkan perbandingan rata-rata iterasi hash pada DMSP menggunakan 9 varian fungsi ambang dengan original MSP dengan kekuatan puzzle 22 bit. Semakin tinggi jumlah iterasi hash menunjukkan delay atau waktu pemrosesan yang tinggi di sisi pengirim. Semakin tinggi nilai kuartil menunjukkan jumlah iterasi yang semakin tinggi. DMSP_Q1power1 memiliki rata-rata iterasi hash yang paling rendah dibandingkan dengan metode lainnya. Nilai yang rendah ini diikuti dengan DMSP_Q1exp dan DMSP_Q2power2. Rata-rata iterasi hash pada MSP dengan kekuatan puzzle 22 bit adalah $2^{22} = 4194304$. Sedangkan rata-rata iterasi hash tertinggi pada kuartil pertama yaitu sebesar 1707555 yang berasal dari

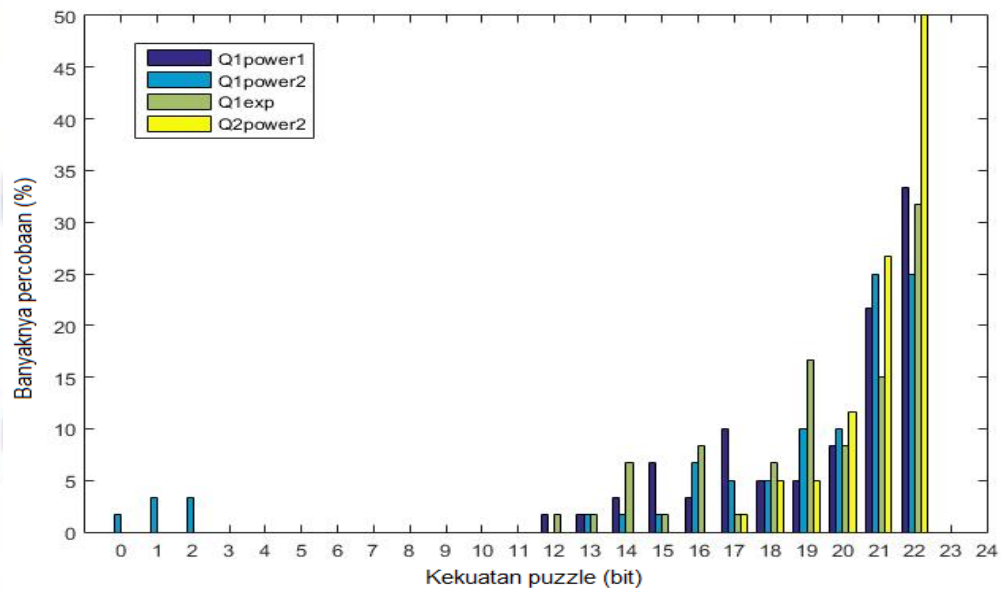
DMSP_Q1power2 dengan panjang pesan 33 bit. Nilai tersebut lebih rendah 60% dibandingkan dengan MSP. Pada kuartil kedua, rata-rata iterasi hash terendah terjadi pada skema DMSP_Q2power2 kemudian diikuti oleh DMSP_Q2exp dan DMSP_Q2power1. Rata-rata iterasi hash pada kuartil 3 secara keseluruhan lebih rendah dibandingkan dengan MSP_*l*=22. Akan tetapi pada pada DMSP_Q3power2 untuk pesan sepanjang 11 bit dan DMSP_Q3power1 untuk pesan sepanjang 22 bit memiliki rata-rata iterasi hash yang lebih tinggi dibandingkan dengan MSP_*l*=22. Hal ini menunjukkan rata-rata iterasi hash pada kuartil ketiga sudah mendekati nilai iterasi hash pada MSP_*l*=22. Oleh karena itu fungsi ambang pada kuartil ketiga tidak direkomendasikan untuk digunakan pada skema puzzle dinamis karena nilai ambang mendekati bahkan bisa melebihi iterasi hash pada skema MSP tanpa fungsi ambang dan delay pada fungsi ambang tersebut masih tinggi.

Parameter lain yang digunakan untuk pemilihian fungsi ambang adalah peluang ditemukannya solusi pada kedinamisan kekuatan puzzle. Analisa mengenai apakah ada jaminan terhadap ditemukannya solusi puzzle perlu diketahui. Informasi ini menentukan apakah keseluruhan proses pencarian solusi puzzle akan diulang jika tidak ditemukan solusi dan hal ini meningkatkan iterasi hash yang berdampak pada tingginya delay. Berdasarkan rata-rata iterasi hash yang didapatkan pada setiap kekuatan puzzle dan Persamaan (2.6) maka peluang kesuksesan pada DMSP dapat dicari dan hasilnya ditunjukkan pada Gambar 4.4. Pada grafik tersebut terlihat dua bagian peluang ditemukannya solusi berdasarkan kuartil pada kekuatan puzzle 22 bit atau kekuatan puzzle maksimum. Semakin tinggi nilai kuartil maka semakin tinggi peluang ditemukannya solusi puzzle. Peluang ditemukannya solusi pada kuartil pertama sebesar 0.22 sedangkan pada kuartil kedua sebesar 0.43. Fungsi ambang dengan nilai peluang tertinggi diraih oleh fungsi ambang EXP diikuti dengan Power1 dan Power2 yang terjadi baik pada kuartil 1 maupun 2. Semakin rendah nilai kekuatan puzzle maka semakin tinggi peluang ditemukannya solusi kecuali pada fungsi ambang power baik pada kuartil 1 yaitu Q1power2 maupun kuartil 2 yaitu Q2power2. Fungsi ambang

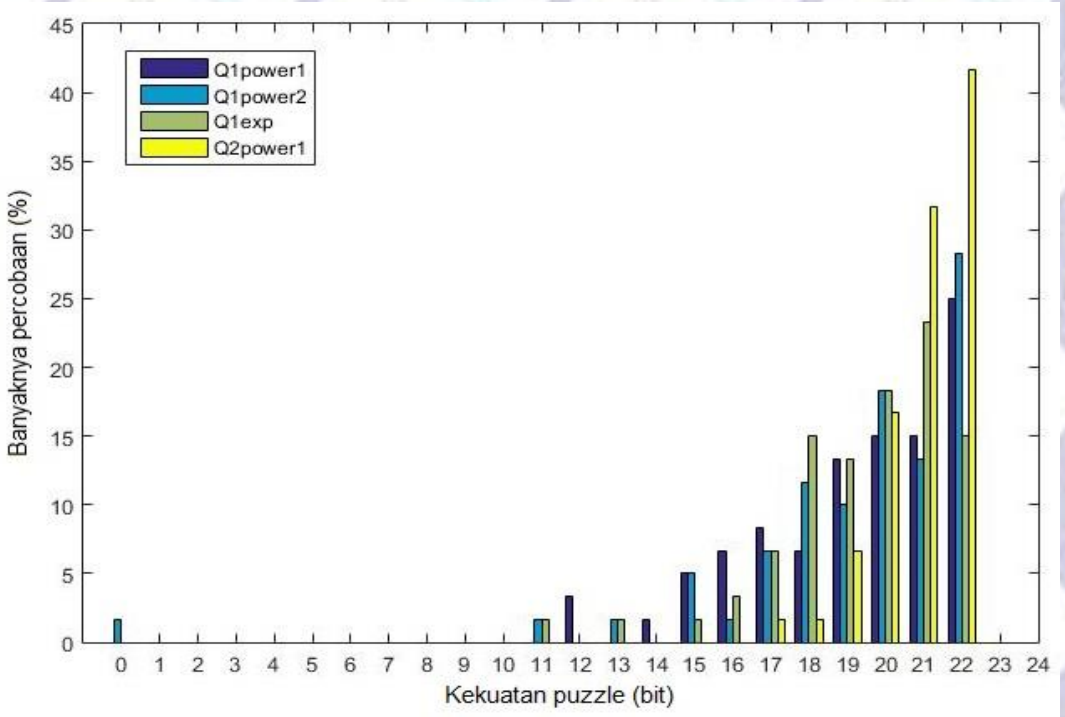


Gambar 4.4 Peluang ditemukannya solusi pada DMSP

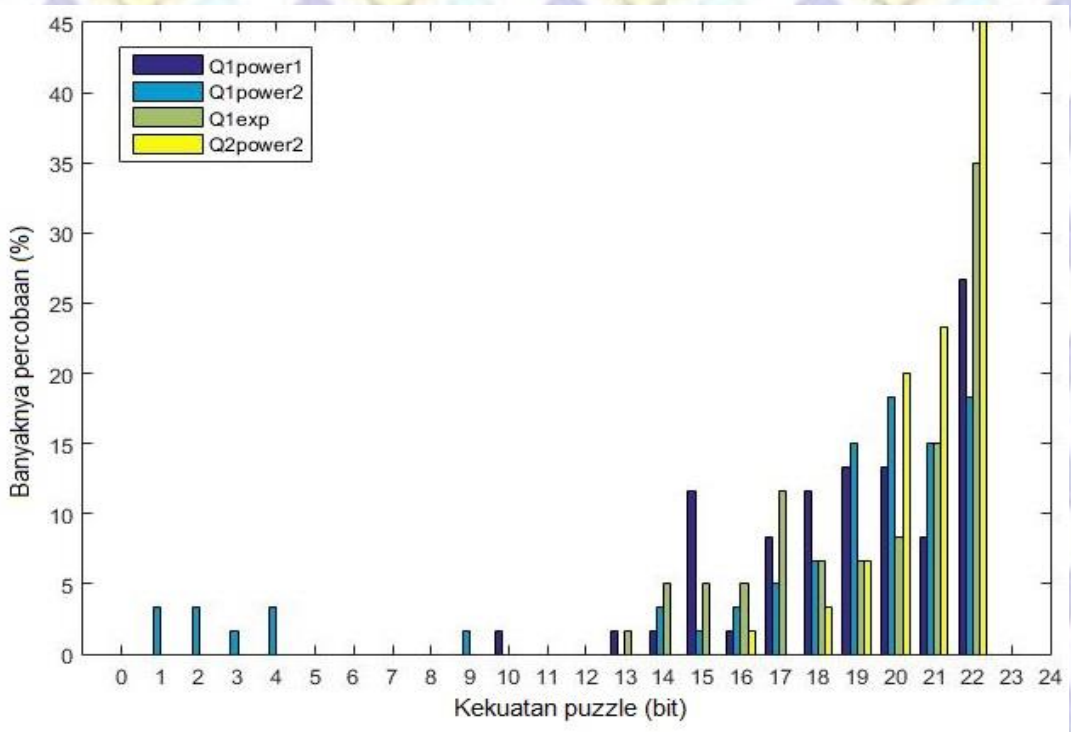
power2 nilainya cenderung menurun meskipun pada titik kekuatan puzzle tertentu peluangnya kembali meningkat. Titik peluang kesuksesan terendah berada pada kekuatan puzzle 7 bit untuk kuartil 1 dan 5 bit pada kuartil 2. Pernyataan ini didukung dengan proses pengulangan percobaan menggunakan fungsi ambang power2 seperti yang tampak pada Gambar 4.5.



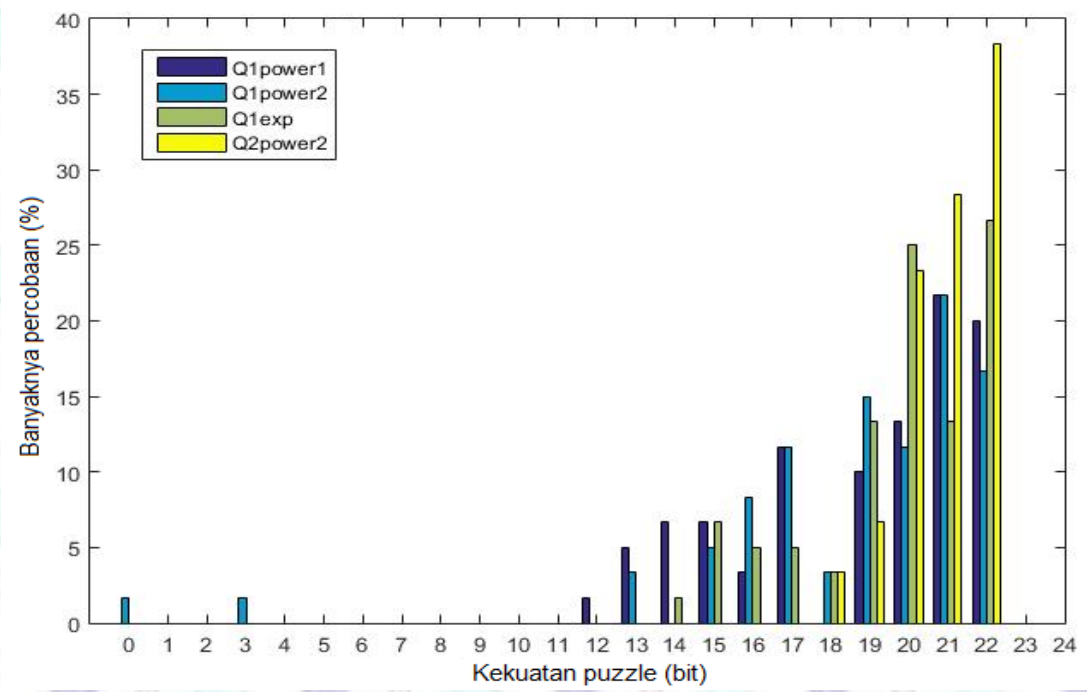
(a). Varian panjang pesan 11 bit



(b). Varian panjang pesan 22 bit



(c). Varian panjang pesan 33 bit

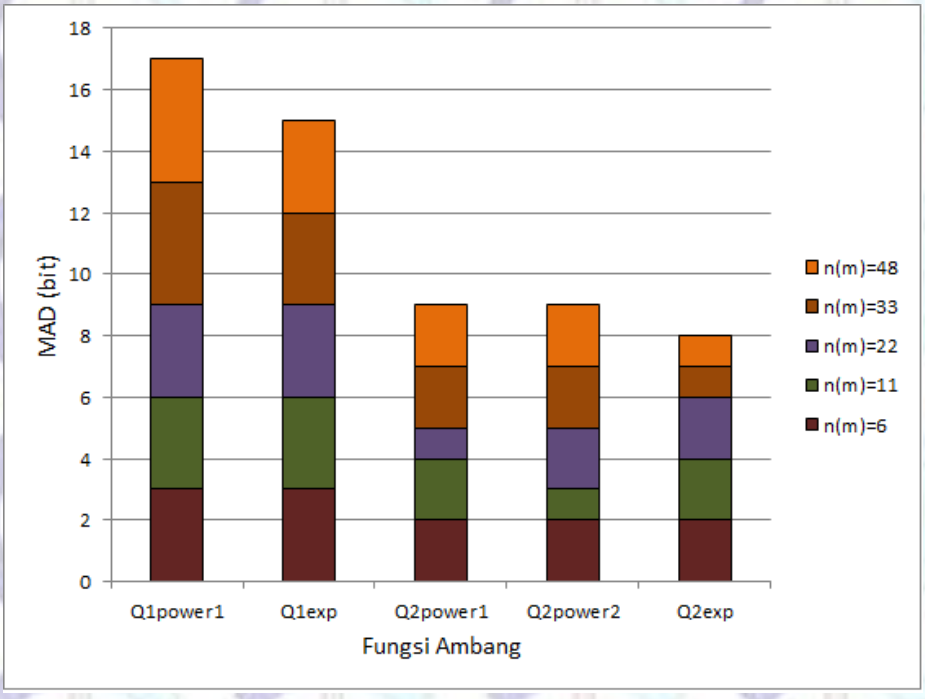


(d). Varian panjang pesan 48 bit

Gambar 4.5 Kekuatan puzzle pada kuartil 1 dan 2 untuk skema DMSP

Informasi mengenai frekuensi nilai kekuatan puzzle yang dihasilkan dari pemanfaatan 4 fungsi ambang yang memiliki rata-rata pengulangan fungsi hash paling rendah pada empat varian panjang pesan didapatkan dari Gambar 4.5. Poin (a), (b) dan (d) pada gambar tersebut menunjukkan terdapat percobaan pada fungsi Q1Power2 yang tidak memiliki solusi pada panjang pesan 11, 22 dan 48 bit. Hasil tersebut sesuai dengan peluang dihasilkannya solusi puzzle yang terlihat pada Gambar 4.4. Meskipun pada panjang pesan 33 bits, poin (c) pada Gambar 4.5, semua percobaan pada fungsi Q1Power2 memiliki solusi puzzle akan tetapi nilai kekuatan puzzle mencapai nilai minimum. Kondisi ini menunjukkan fungsi Q1Power2 tidak handal dalam menemukan solusi puzzle dan tidak direkomendasikan untuk digunakan dalam skema puzzle dinamis.

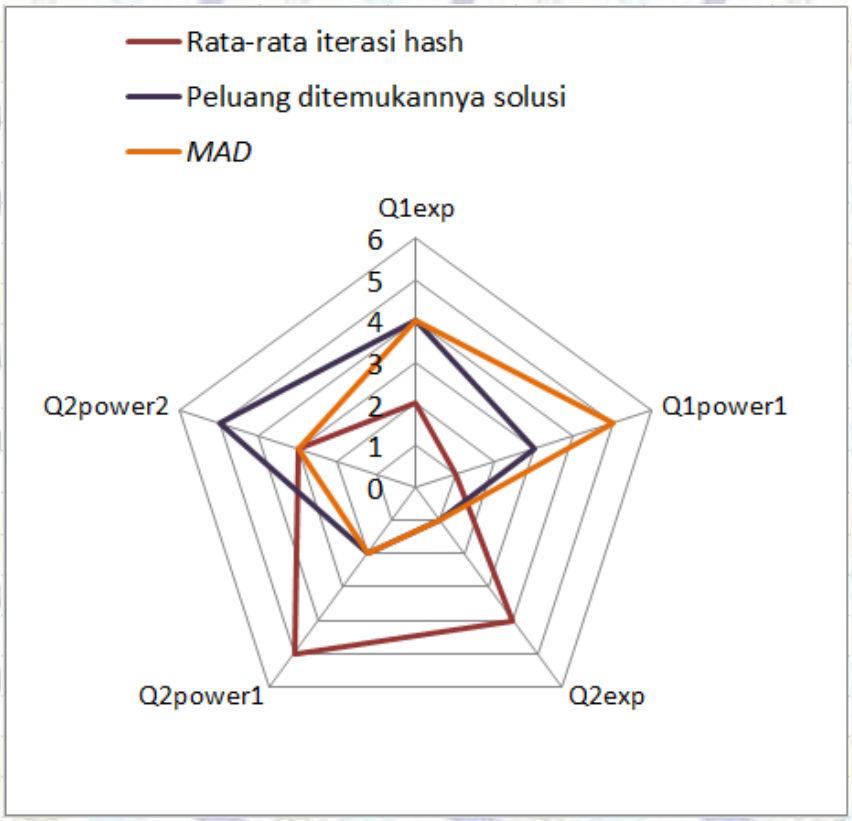
Parameter terakhir yang digunakan dalam menetukan fungsi ambang yang optimum dari kelima kandidat yang tersisa adalah *MAD*. Parameter ini digunakan untuk mengukur jarak kekuatan puzzle pada setiap percobaan dengan kekuatan puzzle maksimum. MAD bernilai 0 tidak diharapkan pada penelitian ini. Hal tersebut dikarenakan nilai 0 diartikan sebagai tidak ada yang dinamis pada



Gambar 4.6 MAD kekuatan puzzle pada skema DMSP

nilai kekuatan puzzle. Dengan kata lain, MAD 0 sama dengan skema MSP yang tidak menggunakan fungsi ambang. Hasil dari perhitungan MAD pada kelima fungsi dibulatkan ke atas dan ditunjukkan pada Gambar 4.6. Q2exp merupakan fungsi ambang dengan nilai MAD terendah yang dapat diartikan sebagai rata-rata kekuatan peluang puzzle yang dihasilkan oleh fungsi ambang tersebut berada di kisaran maksimum kekuatan puzzle yaitu 22 bit. Nilai MAD terendah lainnya terjadi pada Q2power2, Q2power1, dan Q1exp. Nilai MAD tertinggi terjadi pada fungsi ambang Q1power1. Hal ini menandakan bahwa rata-rata kekuatan puzzle menggunakan fungsi ambang tersebut bernilai jauh dari kekuatan puzzle maksimum atau kisaran nilai kekuatan puzzle cukup beragam.

Ketiga parameter yang telah dijelaskan menentukan pemilihan fungsi ambang yang optimal. Diagram spider chart dibangun untuk merangkum hasil percobaan dari kelima kandidat fungsi ambang dan mempermudah dalam menentukan fungsi ambang yang akan digunakan pada komunikasi pengguna dan JSN secara real-time. Diagram ini ditunjukkan pada Gambar 4.6. Fungsi ambang Q1power1 memiliki performansi jumlah iterasi hash dan peluang solusi puzzle yang lebih baik dibandingkan Q1exp. Sedangkan fungsi ambang Q2exp memiliki



Gambar 4.7 Rangkuman hasil percobaan fungsi ambang metode eksperimental

performansi lebih baik pada parameter peluang ditemukannya solusi dan MAD dibandingkan Q2power2 dan Q2power1. Oleh karena itu Q1power1 dan Q2exp direkomendasikan menjadi fungsi ambang pada skema DMSP. Jika keamanan lebih diutamakan maka Q2exp bisa dipilih sedangkan jika waktu merupakan prioritas utama maka Q1power1 bisa dimanfaatkan sebagai fungsi ambang pada skema DMSP.

4.3. Analisa keamanan DMSP

Pada sudut pandang keamanan, kekuatan puzzle yang statis memungkinkan penyerang untuk membuat puzzle palsu lain yang memiliki kekuatan puzzle yang sama. Proses ini disebut serangan probabilitas (Du & Chen 2008). Serangan ini dapat diimplementasikan dengan mudah pada skema MSP. Pada skema puzzle dinamis, serangan ini dapat dimodifikasi dengan menggunakan puzzle palsu yang memiliki kekuatan rendah dan akan membanjiri proses verifikasi di sisi penerima. Serangan ini dihadapi oleh DMSP

menggunakan mekanisme tag. Penyerang harus terlebih dahulu menebak sejumlah parameter seperti kekuatan puzzle pada pengiriman sebelumnya, indeks dan juga kunci sesi. Seperti dijelaskan pada Bab 3.2.3, tag membungkus parameter tersebut pada fungsi hash sehingga tidak mudah bagi penyerang untuk menebaknya karena yang dikirimkan hanya sebagian hasil operasi hash bukan seluruhnya. Terlebih lagi penyerang harus menebak kunci sesi yang dibangun menggunakan *backward one-way key chain*. Penggunaan metode ini tidak mudah diretas kecuali mengetahui keseluruhan isi kunci yang dibangkitkan pada sisi penerima. Menggunakan metode brute force, penyerang harus mencoba sebanyak 2^{64} kali sesuai dengan panjang kunci sesi. Sedangkan untuk menebak nilai tag yang sesuai, penyerang harus mencoba sebanyak 2^{l} sesuai dengan panjang tag. Sehingga kekuatan keamana pada proses pre autentikasi DMSP sebesar $2^{l}. 2^{64} = 2^{(l+64)}$ iterasi percobaan.

Skema DMSP menggunakan fungsi ambang Q1power1 dan Q2exp telah direkomendasikan. Jika keamanan lebih diutamakan maka Q2exp bisa dipilih sedangkan jika waktu merupakan prioritas utama maka Q1power1 bisa dimanfaatkan sebagai fungsi ambang pada skema DMSP. Skema puzzle dinamis pada pesan plaintext ini menggunakan metode eksperimental sehingga dapat menurunkan rata-rata iterasi hash hingga 60%. Mekanisme tag dibangun untuk melengkapi DMSP dalam mengirimkan kekuatan puzzle dan indeks secara implisit. Hal ini meningkatkan kompleksitas serangan dalam memprediksi isi dari pesan yang ditransmisikan. Skema yang diajukan memberi dampak pada peningkatan kompleksitas verifikasi di sisi penerima dengan penambahan operasi paling besar $l \times H(.)$ untuk verifikasi tag.

Halaman ini sengaja dikosongkan

BAB 5

Dynamic Cipher Puzzle (DCP) dengan rumus probabilistik

Skema DCP dibangun dengan tujuan untuk menurunkan delay pada skema cipher puzzle dengan enkripsi pada pesan yang dikirim. Metode yang digunakan yaitu fungsi ambang dengan cara rumus probabilistik yang telah dibahas pada Bab 3.2.2. Penelitian ini mengusulkan 2 kandidat fungsi ambang yaitu linear dan eksponensial. Keduanya akan dibandingkan dengan fungsi ambang metode eksperimental pada kuartil 1 dengan tujuan mendapat fungsi ambang terbaik dan memiliki delay yang terendah. Kelima kandidat fungsi ambang yang diujicobakan tercantum pada Tabel 5.1 dengan dua fungsi ambang pertama didapatkan dari rumus probabilistik dan tiga fungsi ambang sisanya didapatkan dari metode eksperimental. Fungsi ambang pada kuartil kedua tidak diikutsertakan karena delay pada sisi pengirim masih tinggi dibandingkan delay pada kuartil 1.

Maksimum kekuatan puzzle yang dipilih pada penelitian ini adalah 24 bit. Angka ini didapatkan dari kelipatan 8 bit yang paling mendekati 22 bit (nilai rekomendasi yang sesuai dengan performansi skema puzzle) (Ning & Liu 2008).

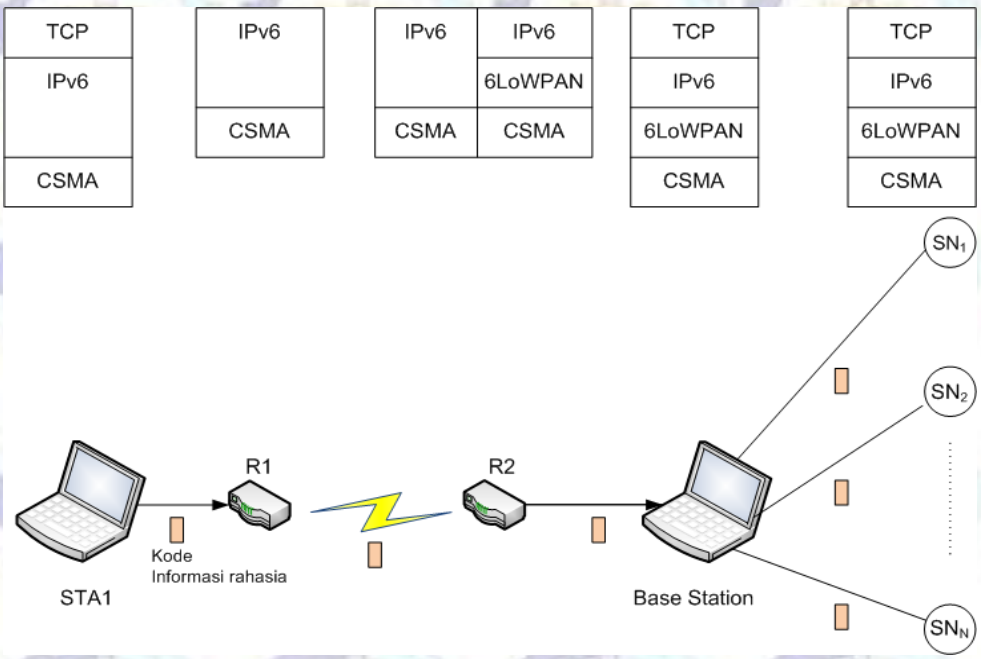
Tabel 5.1 Kandidat fungsi ambang menggunakan rumus probabilistik dan eksperimental

Nama fungsi ambang	Deskripsi
Linear	$\frac{log_{10}\left\{\mathbf{1}-\left[(1-10^{-10})\frac{[l-(l_{maks}+1)]}{(-l_{maks})}\right]\right\}}{log_{10}\left\{\mathbf{1}-\left(\frac{\mathbf{1}}{\mathbf{2}}\right)^{\boldsymbol{l}}\right\}}$
Eksponensial	$\frac{log_{10}\left\{\mathbf{1}-\left[\frac{(1-10^{-10})}{l_{maks}{}^{2}}\left(l-(l_{maks}+1)\right)^{2}\right]\right\}}{log_{10}\left\{\mathbf{1}-\left(\frac{\mathbf{1}}{\mathbf{2}}\right)^{\boldsymbol{l}}\right\}}$
Q1power1	$\lceil 1.8786^{l}+1726\rceil$
Q1power2	$\lceil l^{12.4}\times 2.41\times 10^{-11}\rceil$
Q1exp	$\lceil e^{(l\times 0.599)}\times 1.966\rceil$

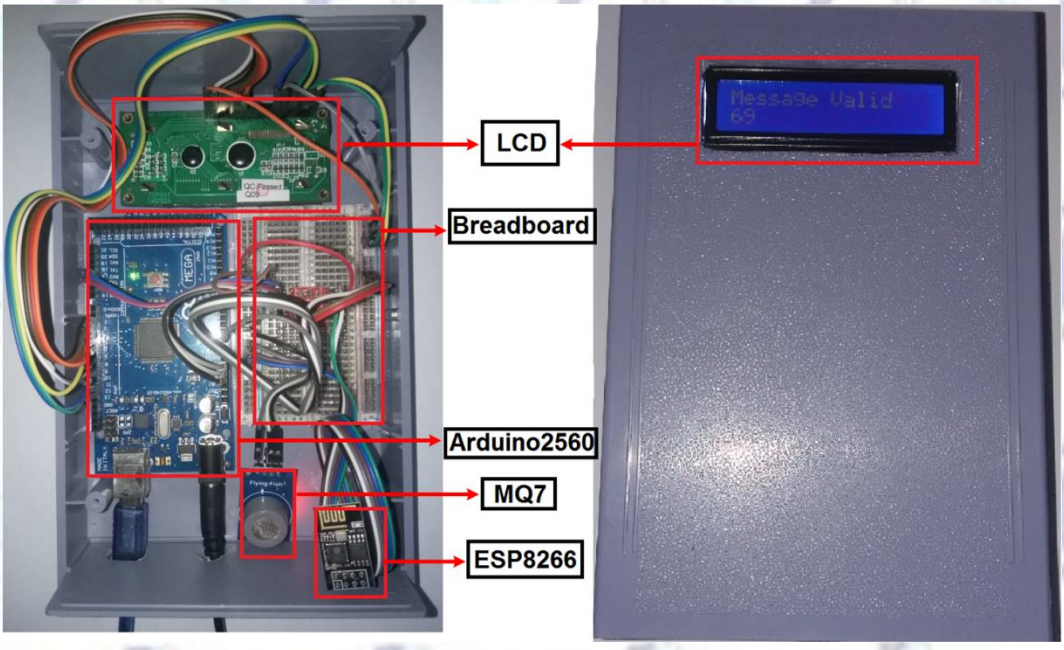
5.1. Skenario uji coba DCP

Dalam menguji kandidat fungsi ambang, penilitian ini menjalankan dua lingkungan percobaan yaitu simulasi dan test bed yang memanfaatkan perangkat mikrokontroller disertai dengan sensor sehingga dapat disebut sebagai sebuah node sensor.

Implementasi dari DCP menggunakan aplikasi NS-3 versi 3.20 yang secara spesifik memanfaatkan modul JSN. Detail dari topologi yang dibangun ditunjukkan pada Gambar 5.1. Pengirim terdiri dari satu workstation yang diberi nama STA1. Sebelum menuju dan menerima ke dan dari Internet perangkat pada kedua sisi perangkat dihubungkan oleh router R1 pada sisi pengirim dan router R2 pada sisi penerima. Setelah melewati router R2 pesan pada sisi penerima akan diteruskan pada base station sebagai pengoleksi dan pengatur JSN. Node sensor penerima hanya memiliki satu tingkat yang terdiri dari 10 node sensor. Tidak ada alasan khusus memilih jumlah node sensor penerima dan bisa digantikan dengan nilai berapapun. Sedangkan jumlah tingkatan hanya satu karena penelitian ini tidak fokus pada routing pada kondisi multihop melainkan fokus pada proses broadcast pada skema flooding sederhana.



Gambar 5.1 Topologi jaringan simulasi DCP



Gambar 5.2 Node sensor yang digunakan untuk percobaan

Selanjutnya DCP juga diujicobakan pada perangkat keras Arduino yang telah terpasang sensor untuk mendeteksi parameter tertentu. Tujuan dari percobaan dengan memanfaatkan perangkat keras adalah mengetahui besarnya peningkatan ruang penyimpanan dan RAM jika ditambahkan skema DCP. Detail dari node sensor yang dirakit ditunjukkan pada Gambar 5.2. Komponen dari perangkat tersebut terdiri dari LCD dengan ukuran 16 karakter x 2 baris, breadbroad sebagai penghubung antar kabel jumper dari komponen-komponen, mikrontroller Arduino Mega 2560, sensor MQ7 untuk mendeteksi karbon monoksida dan ESP8266 sebagai modul untuk komunikasi data. Sedangkan perangkat pengirim memiliki spesifikasi Intel Core i3-2310M CPU @ 2.10 GHz—4CPUs processor and 6144 MB DDR3. Kedua perangkat tersebut terhubung melalui Wireless Access Point.

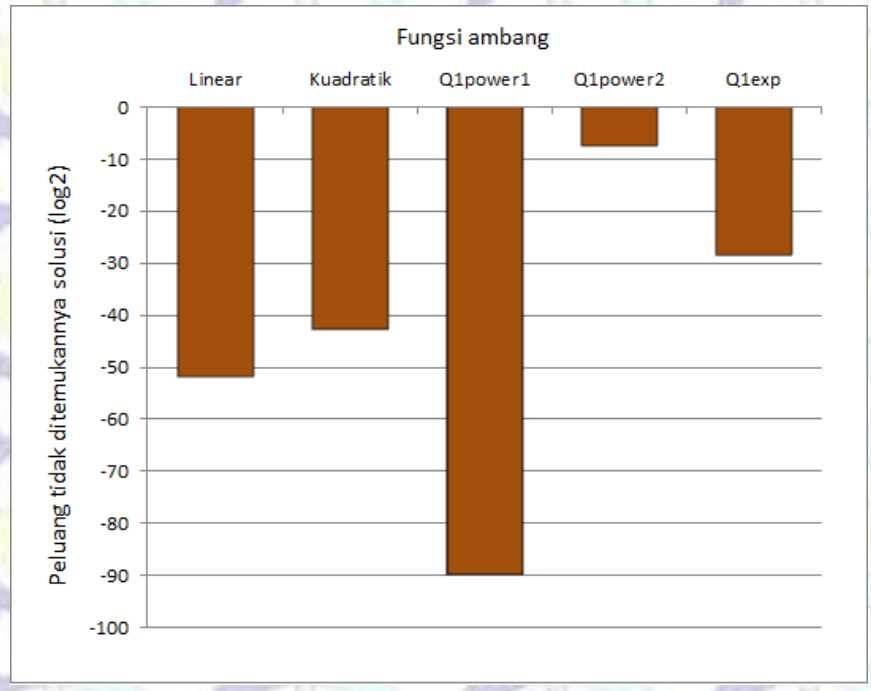
5.2. Hasil dan pembahasan DCP

Performansi dari fungsi ambang pada simulasi DCP ditunjukkan melalui parameter peluang tidak ditemukannya solusi, rata-rata jumlah iterasi hash, standar deviasi dan maksimum iterasi hash. Sedangkan pada uji coba perangkat keras, performansi fungsi hash diketahui melalui parameter sumber

daya yang dibutuhkan untuk menjalankan DCP yaitu waktu eksekusi, ruang penyimpanan, RAM dan energi yang dibutuhkan.

5.2.1. Simulasi skema DCP

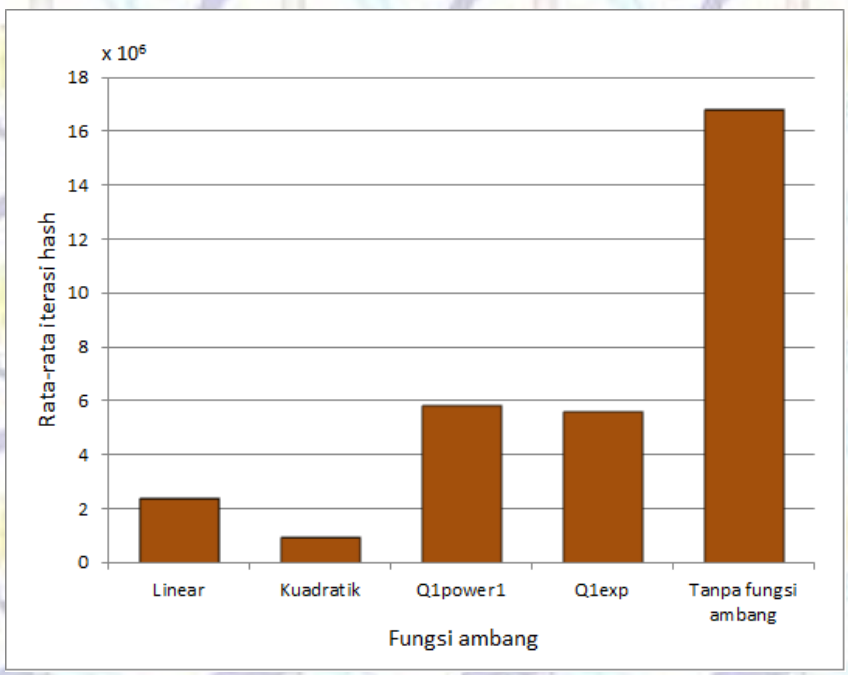
Peluang tidak ditemukannya solusi penting untuk diketahui pada skema DCP. Hal ini dikarenakan pembatasan jumlah iterasi hash pada setiap tingkatan kekuatan puzzle memungkinkan tidak ditemukannya solusi. Fungsi ambang ini menjadi tidak berguna jika dalam satu kali perputaran dari kekuatan puzzle maksimum menuju minimum tidak ditemukan solusi sebab proses harus diulang dari ke putaran selanjutnya yang dimulai lagi dari kekuatan puzzle maksimum. Adanya fungsi ambang pada DCP bukan menurunkan delay pada sisi pengirim bahkan nilainya akan bertambah. Hasil dari peluang tidak ditemukannya solusi pada kelima kandidat fungsi ambang yang disebutkan pada Tabel 5.1 ditunjukkan pada Gambar 5.3. Nilai yang ditampilkan tersebut merupakan nilai logaritmik berbasis 2, karena nilai sesungguhnya sangat kecil. Peluang tidak ditemukannya solusi paling rendah terjadi pada fungsi ambang Q1power1 yaitu 1.06×10^{-27} (-89.61 pada logaritmik berbasis 2) sedangkan peluang paling tinggi



Gambar 5.3 Peluang tidak ditemukannya solusi pada skema DCP

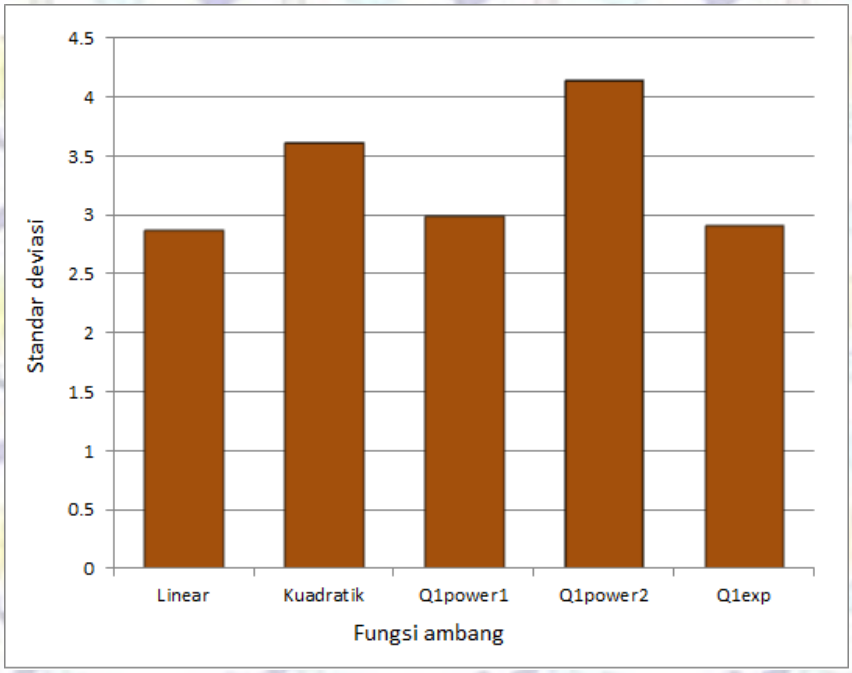
terjadi pada fungsi ambang Q1power2 yaitu 0.00766 (-7.03 pada logaritmik berbasis 2). Peluang tidak ditemukannya solusi pada fungsi ambang linear dan eksponensial bernilai $2.96x10^{-16}$ (-51.58 pada logaritmik berbasis 2) dan $1.73x10^{-13}$ (-42.39 pada logaritmik berbasis 2), secara berurutan. Nilai tersebut lebih rendah dari fungsi ambang Q1exp yang bernilai $3.23x10^{-9}$ (-28.21 pada logaritmik berbasis 2) akan tetapi lebih tinggi dibandingkan Q1power1. Hal ini diartikan bahwa fungsi ambang dengan rumus probabilistik memiliki peluang tidak ditemukannya solusi dengan performansi yang lebih baik dari fungsi ambang Q1exp dan lebih buruk jika dibandingkan dengan fungsi ambang Q1power1.

Rata-rata iterasi hash merupakan parameter selanjutnya yang dianalisa pada kandidat fungsi ambang. Nilai yang didapat merupakan evaluasi dari pengulangan pembangkitan puzzle sebanyak 400 kali pada sisi pengirim. Hasil yang didapatkan ditunjukkan pada Gambar 5.4. Fungsi ambang Q1power2 tidak diikutsertakan dalam pengukuran ini dikarenakan nilai peluang tidak ditemukannya solusi cukup besar sehingga tidak direkomendasikan untuk digunakan (Afianti et al. 2019). Skema cipher puzzle (Tan et al. 2013) diikutsertakan dalam pengukuran untuk mengetahui batas ambang paling atas dari



Gambar 5.4 Rata-rata iterasi hash pada skema DCP

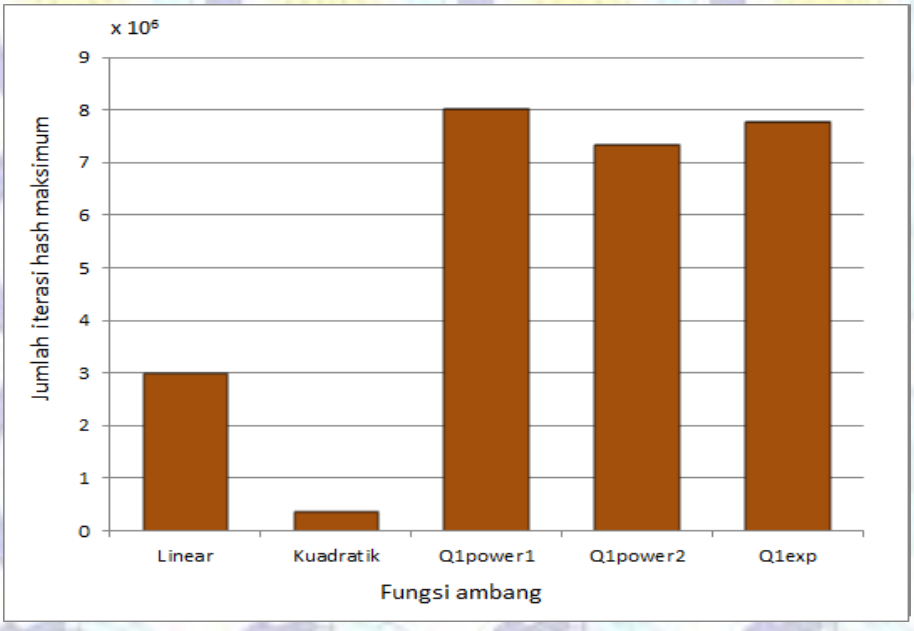
skema puzzle yang tidak menggunakan fungsi ambang. Rata-rata iterasi hash paling tinggi pasti dialami oleh skema cipher puzzle tanpa fungsi ambang yaitu 16.558.307 iterasi hash. Nilai ini tidak berbeda jauh dengan nilai rata-rata iterasi hash berdasarkan rumus yang didapatkan oleh bernoulli trial pada Bab 2.3.4 yang bernilai $2^l = 2^{24} = 16.777.216$. Jika setiap iterasi hash membutuhkan 0.1 ms maka untuk menghasilkan solusi pada skema puzzle tanpa fungsi ambang membutuhkan waktu $0.1x10^{-3}$ x 16.558.307=1655.8 detik atau setara dengan 27.6 menit. Sedangkan rata-rata iterasi hash paling rendah terjadi pada skema DCP menggunakan fungsi ambang eksponensial dengan nilai iterasi hash sebanyak 905.068. Iterasi hash pada fungsi ambang ini turun sebesar 94.53% dibandingkan dengan tanpa menggunakan fungsi ambang. Nilai terendah ini diikuti oleh fungsi ambang linear, Q1exp dan Q1power1 dengan rata-rata iterasi hash sejumlah 2.413.957, 5.514.977 dan 5.832.046 secara berurutan. Meskipun nilai iterasi hash pada fungsi ambang linear lebih tinggi dari eksponensial akan tetapi keduanya jauh lebih rendah dibandingkan dengan fungsi ambang menggunakan cara eksperimental.



Gambar 5.5 Standar deviasi pada skema DCP

Parameter standar deviasi merupakan parameter selanjutnya yang akan dianalisa. Tingkat keragaman kekuatan puzzle diperhitungkan untuk meningkatkan kompleksitas serangan pada skema puzzle dinamis. Hasil standar deviasi dari kelima kandidat fungsi ambang dapat dilihat pada Gambar 5.5. Keragaman kekuatan puzzle paling tinggi terjadi pada fungsi ambang Q1power2 akan tetapi sesuai dengan pernyataan sebelumnya, fungsi ambang ini tidak direkomendasikan. Fungsi ambang tertinggi selanjutnya terdapat pada fungsi ambang eksponensial dengan nilai standar deviasi 3.612. Nilai ini diikuti oleh fungsi ambang Q1power1, Q1exp dan Linear dengan nilai standar deviasi sebesar 2.989, 2.906 dan 2.867 berturut turut. Fungsi ambang linear memiliki standar deviasi paling rendah meskipun jaraknya dengan Q1exp hanya terpaut 0.039.

Parameter terakhir yang dianalisa pada simulasi skema DCP adalah maksimum jumlah iterasi hash. Tujuan pengukuran parameter ini adalah untuk memperkirakan maksimum delay yang dibutuhkan untuk membentuk sebuah solusi puzzle pada sisi pengirim. Parameter ini didapatkan dengan menjumlahkan maksimum iterasi hash dari kekuatan puzzle maksimum hingga paling rendah yaitu 1 dan didapatkan hasil sesuai dengan Gambar 5.6. Maksimum iterasi hash pada fungsi ambang dengan cara eksperimental memiliki nilai yang lebih tinggi

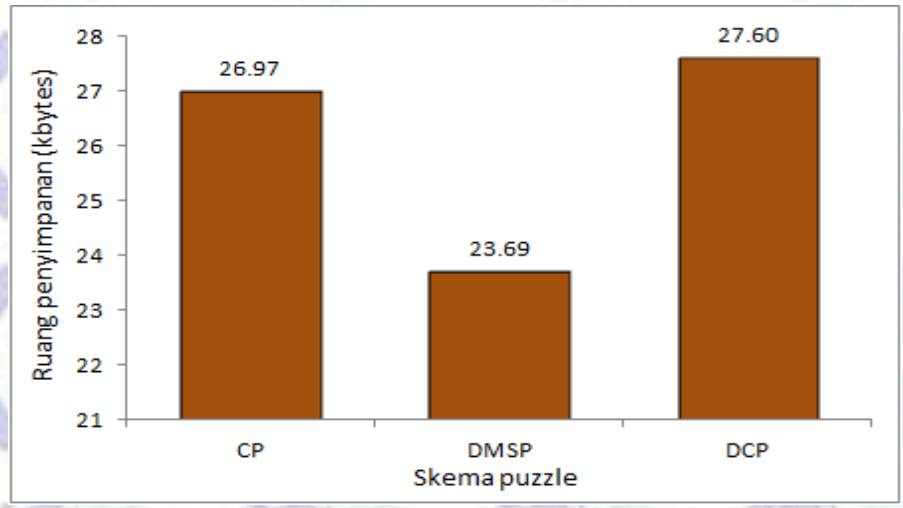


Gambar 5.6 Maksimum jumlah iterasi hash pada skema DCP

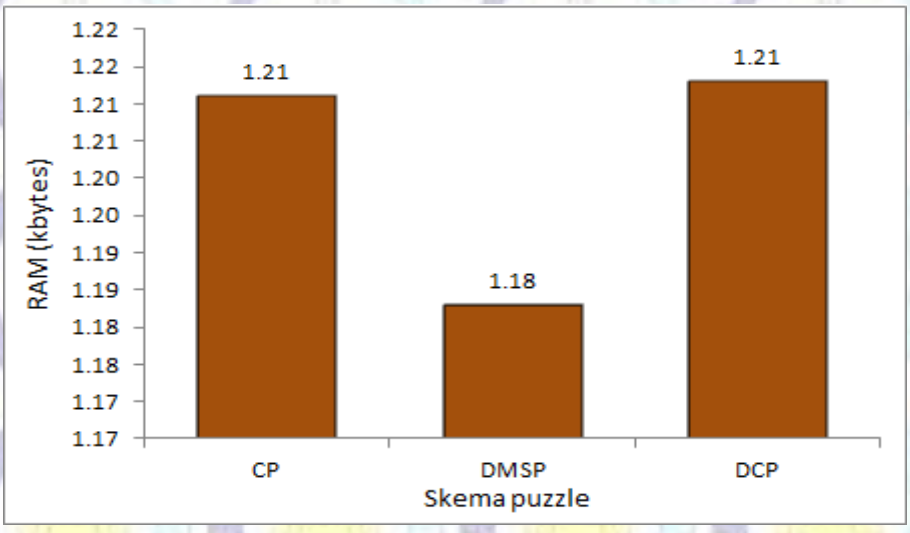
dibandingkan dengan cara rumus probabilistik. Fungsi ambang Q1power1 memiliki nilai maksimum iterasi hash paling tinggi diikuti dengan fungsi ambang Q1exp dan Q1power2 dengan nilai sebesar 8.022.605, 7.764.224 dan 7.329.758 berturut-turut. Hal ini disebabkan fungsi ambang dengan cara eksperimental menggunakan nilai kuartil pertama dari sebaran data. Kuartil pertama bernilai kurang lebih separuh dari nilai rata-rata iterasi hash yaitu sebesar 8.388.608. Sedangkan nilai maksimum jumlah iterasi hash paling rendah terletak pada fungsi ambang eksponensial yaitu bernilai 357.795. Nilai ini hanya berkisar sebesar 4.46% dari maksimum iterasi hash pada fungsi ambang Q1power1 sebagai fungsi ambang tertinggi. Nilai maksimum iterasi hash pada fungsi ambang linear bernilai 2.997.172 atau 8 kali lipat dibanding maksimum iterasi hash pada fungsi ambang eksponensial.

5.2.2. Test bed skema DCP

Percobaan ini dijalankan untuk mengetahui sumber daya khususnya RAM dan ruang penyimpanan yang digunakan pada proses verifikasi skema DCP di sisi penerima menggunakan perangkat keras yang telah dirakit. Skema puzzle yang diujicobakan terdiri dari 3 yaitu DCP, DMSP dan cipher puzzle. Hasil yang didapatkan dapat dilihat pada Gambar 5.7. Secara keseluruhan sumber daya yang dibutuhkan oleh DMSP lebih rendah dibanding dengan 2 skema yang lain.



(a) Ruang penyimpanan



(b) RAM

Gambar 5.7 Sumber daya untuk proses verifikasi pada Arduino

Ruang penyimpanan antara tiga skema puzzle diperlihatkan pada Gambar 5.7(a). DCP membutuhkan ruang penyimpanan yang paling besar yaitu 27.6 kbyte sedangkan cipher puzzle dan DMSP lebih rendah 0.63 dan 3.91 kbyte berturut-turut. Sedangkan, konsumsi RAM yang ditunjukkan pada Gambar 5.7(b). Skema DCP dan cipher puzzle membutuhkan 1.21 kbyte sedangkan DMSP lebih rendah 0.03 kbyte. DMSP secara umum memiliki sumber daya yang paling rendah dikarenakan hanya memverifikasi tag, puzzle, kunci sesi dan signature sedangkan cipher puzzle dan DCP menambahkan proses verifikasi menggunakan ETM untuk menjamin proses *confidentiality*. Sumber daya dari DCP lebih tinggi dari cipher puzzle karena pada skema puzzle dinamis memubutuhkan verifikasi tag. Proses ini digunakan untuk mengirimkan kekuatan puzzle dan indeks secara implisit. Meskipun skema DCP paling tinggi akan tetapi jika dirangkum perbedaanya hanya 2.268% pada ruang penyimpanan dibandingkan cipher puzzle.

Analisa sumber daya pada perangkat Arduino akan disandingkan dengan kompleksitas komputasi dan biaya penggunaan dari algoritma pada proses verifikasi untuk beberapa metode filter yang dirangkum pada Tabel 5.2 dengan **H()** diartikan sebagai komputasi untuk menjalankan sebuah fungsi hash dan

Tabel 5.2 Perbandingan kompleksitas komputasi antara metode filter

No	Metode Filter	Overhead pada sisi penerima (node sensor)			Kompleksitas Serangan
		Kompleksitas	Penyimpanan	Komunikasi	
1	MSP [Ning dan Liu, 2008]	2.**H()**	1.K + 1.idx = 8 + 2 = 10 B	$\{idx, M, Sig, Kidx, Pidx\}$ = 2 + (1-48) + 40 + 8 + 4 = 55 - 102 B	$2^{Kidx+Pidx}$ $= 2^{(8+4)8}$ $= 2^{96}$
2	Dynamic MSP	3.**H()**	1.K + 1.idx + 1.L = 8 + 2 + 3 = 13 B	$\{idx, tagging, M, Sig, Kidx, Pidx\}$ = 2 + 3 + (1 - 45) + 40 + 8 + 4 = 58 - 102 B	$2^{Kidx+Pidx+L}$ $= 2^{(8+4)8+22}$ $= 2^{118}$
3	Pre-authenticator [Du dan Chen, 2008]	1.**H()**	N.ID + 2N.K + N.idx = 1.2 + 2.1.8 + 1.2 = 20 B	$\{idx, M, Sig, K_{ID,idx}\}$ = 2 + (1-52) + 40 + 8 = 51 - 102 B	$2^{K_{ID,idx}}$ $= 2^{8.8}$ $= 2^{64}$
4	Group based [Dong dkk., 2013]	$(m+1)$.**H()** =(3+1).**H()** = 4.**H()**	$(m+S)$.K = (3+2).8 = 40 B	$\{M, Sig, W\}$ = (1-58) + 40 + 4 = 45 - 102 B	2^{K+W} $= 2^{(8+4)8}$ $= 2^{96}$
5	Group based - adaptive regroup [Dong dkk., 2013]	$(m+1)$.**H()** + report + reinisialisasi = (3+1).**H()** + O(1) + O(1) = 4.**H()**	$(m+S)$.K = (3+2).8 = 40 B	$\{M, Sig, W\}$ = (1-58) + 40 + 4 = 45 - 102 B	2^{K+W} $= 2^{(8+4)8}$ $= 2^{96}$
6	Key chain [Dong dkk., 2013]	$\{C(1-f_c)+[(C-1)\tau+1]f_c\}$.**H()** = {2(1-0,04) + [(2-1)5+1]. 0,04{ .**H()** =2,16 .**H()** ≈ 3.**H()**	$(S+2\sqrt{n_1})$.K = $(2+2\sqrt{25})$.8 = 96 B	$\{M, Sig, idx, Kidx, W\}$ = (1-48) + 40 + 2 + 8 + 4 = 55 - 102 B	$2^{K+K_{idx}+W}$ $= 2^{(8+8+4)8}$ $= 2^{160}$
7	Hybrid [Dong dkk., 2013]	$\frac{C\times\tau}{2^{Vtempt}}$.**H()** , $Vtemp1 = \frac{q-d_k-e_{idx}}{m}$ $= \frac{2.5}{2^{(128-64-16)/5}}$ = 2,5 .**H()** ≈ 3.**H()**	$[2S+2\sqrt{n_1}+m]$.K = $[2.2+2\sqrt{25}+3]$.8 = 136 B	$\{M, Sig, idx, Kidx, W\}$ = (1-50) + 40 + 2 + 8 + 2 = 53 - 102 B	$2^{K+K_{idx}+W}$ $= 2^{(8+8+2)8}$ $= 2^{144}$
8	Key pool [Chuchaisri dan Newman, 2012]	$V_{Temp} = q_{BFV}\Sigma_{i=\tau_{KP}}^{min(K,h_k)} i\frac{\binom{h_K}{i}\binom{m_{KP}-h_K}{K-i}}{\binom{m_{KP}}{K}}$.**H()** $= 3\,\Sigma_{i=1}^{min(3,1)} 1\frac{\binom{1}{1}\binom{3-1}{3-1}}{\binom{3}{3}}$.**H()** = 3.**H()**	k_i.$K + q_{BFV}$.H = 3.8 + 3.20 = 84 B	$\{M, tt, Sig, II, BFV\}$ = (1-50) + 2 + 40 + 6 + 4 = 53 - 102 B	2^{II+BFV} $= 2^{(6+4)8}$ $= 2^{80}$
9	Improved key pool [Chuchaisri dan Newman, 2012]	$V_{Temp} + \frac{m}{2} + K + 1$.**H()** = $[3+\frac{3}{2}+3+1]$.**H()** = 8,5 .**H()** ≈ 9.**H()**	k_i.$K + (q_{BFV}+3)$.H = 3.8 + (3+3).20 = 144 B	$\{M, tt, Sig, K_{[idx-1]}, BFV\}$ = (1-48) + 2 + 40 + 8 + 4 = 55 - 102 B	$2^{K_{[idx-1]}+BFV}$ $= 2^{(8+4)8}$ $= 2^{96}$
10	Key chain [Chuchaisri dan Newman, 2012]	$K.((idx-idx')+q_{BFV})$.**H()** = (3(1) + 3).**H()** = 6.**H()**	k_i.$K + (q_{BFV}+1)$.H = (3.8) + (4.20) = 104 B	$\{M, tt, Sig, idx, BFV\}$ = (1-54) + 2 + 40 + 2 + 4 = 49 - 102 B	$2^{idx+BFV}$ $= 2^{(2+4)8}$ $= 2^{48}$
11	Cipher Puzzle [Tan dkk., 2013]	**RC5()** + **H()** + [**HMAC()** ≈ **H()**] = **RC5()** + 2.**H()**	$K_i + P_i$ = 8 + 4 = 12 B	$\{Sig, MAC-L, EncM\}$ = 40 + (20-3) + 32 = 89 - 92 B	2^{K+Pidx} $= 2^{(8+4)8}$ $= 2^{96}$
12	Dynamic Cipher Puzzle	**RC5()** + 2.**H()** + [**HMAC()** ≈ **H()**] = **RC5()** + 3.**H()**	$K_i + P_i + idx + L$ = 8 + 4 + 2 + 3 = 17 B	$\{Sig, tag, MAC, EncM\}$ = 40 + 3 + 17 + 32 = 92 B	$2^{K+Pidx+L}$ $= 2^{(8+4)8+L}$ $= 2^{118}$

RC() diartikan sebagai komputasi untuk menjalankan sebuah proses dekripsi menggunakan algoritma RC5 (detail prosesnya dijelaskan pada Lampiran E). Penelitian ini telah membandingkan 12 skema filter dalam rangka mengatasi serangan DoS berbasis PKC. Parameter yang dibandingkan antara lain kompleksitas komputasi, biaya komunikasi dan ruang penyimpanan yang dibutuhkan oleh node sensor. Selain itu, juga terdapat parameter kompleksitas serangan yang diartikan sebagai metrik untuk menjelaskan kondisi di bawah kendali peretas dan harus ada untuk mengetahui kelemahan sebuah sistem (Maris et al. 2019). Berdasarkan perbandingan beberapa parameter tersebut, dapat dilihat bahwa skema MSP (Ning & Liu 2008) cukup efisien dalam pemanfaatan ruang penyimpanan maupun kompleksitas komputasi meskipun biaya komunikasinya cukup tinggi. Selain itu, pengirim perlu mengirimkan kunci sesi, indeks dan solusi puzzle sehingga pesan yang dikirimkan terbatas pada 48 bytes. Konsep skema puzzle ini dimanfaatkan untuk kasus diseminasi kode pada cipher puzzle (Tan et al. 2013). Sedangkan skema yang memiliki kompleksitas komputasi paling rendah di sisi penerima adalah *Pre-authenticator* (Du & Chen 2008) yang hanya membutuhkan sebuah iterasi hash untuk verifikasi puzzle. Akan tetapi, skema ini membutuhkan biaya penyimpanan yang tinggi untuk menyimpan kunci sesi dari setiap pengirim. Hal ini dikarenakan penggunaan *one-way key chain* yang unik atau berbeda-beda pada setiap pengirim. Skema Hybrid (Dong et al. 2013) memiliki kompleksitas serangan yang tertinggi setelah *Pre-authenticator*. Hal ini juga diikuti dengan ruang penyimpanan yang paling tinggi dikarenakan perlunya menyimpan kunci kelompok dari setiap komunikasi yang akan dijalankan. Skema puzzle memiliki kompleksitas komputasi dan ruang penyimpanan yang rendah dibanding lainnya akan tetapi pada implementasinya masih memiliki delay di sisi pengirim sehingga dikembangkanlah DMSP dan DCP.

Skema puzzle dinamis meningkatkan kompleksitas verikasi sebesar **1.H()** baik pada DMSP maupun DCP. Jika menggunakan pendekatan Big O, peningkatan ini berdampak O(n) atau sebanding dengan panjang input pengguna (Cormen et al. 2009). Dengan biaya komunikasi yang sama, ruang penyimpanan pada DMSP meningkat sebesar 3 byte dibandingkan MSP. Sedangkan DCP

meningkatkan kompleksitas verifikasi sebesar **1.RC5()** dibandingkan dengan DMSP dan sebesar **1.H()** dibandingkan cipher puzzle. Kompleksitas algoritma **RC5()** bergantung pada parameter *round* (*r*) atau perulangan pergerseran bit. Jika menggunakan pendekatan Big O, peningkatan ini hanya berdampak O(*r*). Dengan biaya komunikasi yang sama, ruang penyimpanan pada DCP meningkat sebesar 5 byte dibandingkan cipher puzzle. Peningkatan kompleksitas dan ruang penyimpanan pada kedua skema puzzle dinamis tersebut (DMSP dan DCP) diimbangi dengan peningkatan keamanan yang ditunjukkan dengan peningkatan kompleksitas serangan sebesar 2^l dibandingkan dengan metode sebelumnya (MSP dan cipher puzzle). Jika penelitian ini menggunakan l_{maks} = 22 bit, maka peningkatan kompleksitas serangan sebesar 2^{22}.

Kompleksitas komputasi pada skema DCP dapat dilihat dari dua sisi yaitu pengirim dan penerima. Komputasi tersebut dapat diukur melalui pendekatan Big O. Pada sisi pengirim, kompleksitas komputasi secara keseluruhan ditunjukkan pada Tabel 5.3 dengan analisa algoritma yang didasari dari ide peneliti sebelumnya (Cormen et al. 2009; Gumelar 2013). Komputasi tertinggi dibutuhkan oleh proses pembuatan signature (penelitian ini menggunakan ECDSA) yang ditunjukkan pada baris ke-1. Big O dari proses pembangkitan signature bernilai O(n^3) dengan *n* adalah panjang atau ukuran inputan pengguna. Hal ini dikarenakan ada penggunaan algoritma perkalian skalar pada sebuah titik di kurva elips $y^2 = x^3 + ax + b$. Perkalian skalar ini memerlukan proses kelipatan titik (*double point*) dan penambahan dari titik (Johnson et al. 2001; Cormen et al. 2009). Komputasi ini semakin tinggi jika nilai skalarnya semakin tinggi. Proses dengan komputasi tertinggi kedua adalah pencarian solusi puzzle pada baris ke-6 hingga 19. Proses ini merupakan perulangan fungsi hash hingga ditemukannya solusi. Pada skema DCP maksimum iterasi pada proses ini dilambangkan dengan *iter*. Ada 2 algoritma utama yang dijalankan pada pencarian solusi puzzle yaitu enkripsi menggunakan RC5 dengan kompleksitas yang bergantung pada jumlah round atau O(*r*) dan fungsi hash menggunakan SHA1 yang bergantung pada panjang inputan atau O(*n*). Oleh karena itu, kompleksitas komputasi pada pencarian solusi yaitu O(*iter.r.n*). Proses

Tabel 5.3 Kompleksitas komputasi pengirim pada skema DCP

Baris ke-	Pseudo-code	Nilai Big O
1	*Sig* ← **ECDSA_sign**(*M*)	O(n^3)
2	*L*, L_{prev} ← l_{maks}	O(1)
3	*iter* ← 0	O(1)
4	*Threshold* ← **Fungsi_ambang**(*L*)	O(1)
5	*Finding* ← false	O(1)
6	**While**(**not**(*Finding*)) **and** (*L*>0) **do**	O(*iter*)
7	**If**(*iter* < *Threshold*) **then**	O(1)
8	*iter*++	O(1)
9	*Srand* ←**genRandom**()	O(1)
10	*EncM* ← **Enc_RC5** (*M*|| *Srand*||*Kidx*)	O(*r*)
11	*MAC* ← **HMAC_SHA1**(*EncM*)	O(*n*)
12	**If Hash**(*Sig*).**substr**(*L*) == *MAC*.**substr**(*L*) **then**	O(1)
13	*Finding* ← True	O(1)
14	**End**	O(1)
15	**Else**	O(1)
16	*L*--	O(1)
17	*Threshold* ← **Fungsi_ambang**(*L*)	O(1)
18	**End**	O(1)
19	**End**	O(1)
20	**If** *Finding* **then**	O(1)
21	*Tag* ← **Hash**(*index* + L_{prev} +*L*).**substr**(*L*)	O(*n*)
22	*MAC* ← *MAC*.**substr**(*L*, **Length**(*MAC*)-*L*)	O(*n*)
23	*Index*++	O(1)
24	L_{prev} ← *L*	O(1)
25	P_{DCP} ← *Sig* || *tag*|| *MAC*||*EncM*	O(1)
26	**Send** P_{DCP}	O(1)
27	**Else**	O(1)
28	Zero_Solution	O(1)
29	**End**	O(1)
Jumlah		O(n^3) + O(*iter.r.n*)+O(n)

dengan komputasi tertinggi ketiga yaitu pembentukan MAC dengan kompleksitas O(*n*) yang bergantung pada panjang inputan pengguna. Sehingga secara keseluruhan kompleksitas komputasi di sisi pengirim sebesar O(n^3) + O(*iter.r.n*)+O(*n*).

Pada sisi penerima, kompleksitas verifikasi secara keseluruhan ditunjukkan pada Tabel 5.4. Kompleksitas komputasi tertinggi di sisi penerima yaitu pada proses verifikasi signature ECDSA yang terletak pada baris ke-21. Pada saat pembuatan maupun verifikasi, ECDSA membutuhkan perkalian skalar

yang melibatkan kelipatan sebuah titik dan penambahan dengan kompleksitas komputasi sebesar O(n^3). Sedangkan, kompleksitas komputasi tertinggi kedua

Tabel 5.4 Kompleksitas komputasi penerima pada skema DCP

Baris ke-	Pseudo-code	Nilai Big O
1	Accept ← P_{DCP}	O(1)
2	*MAC* ← P_{DCP}.**substr**(320, 160)	O(*n*)
3	*L* ← l_{maks}	O(1)
4	*Ver_tag* ← false	O(1)
5	**While** (*L*>0) **and** (**not**(*Ver_tag*)) **do**	O(*L*)
6	**If**(*MAC*.**substr**(0,*L*) = **Hash**(*index*+1 + L_{prev} +*L*)).**substr**(0,*L*) **then**	O(*n*)
7	*Ver_tag* ← true	O(1)
8	**Else**	O(1)
9	*L*--	O(1)
10	**End**	O(1)
11	**End**	O(1)
12	**If** *Ver_tag* **then**	O(1)
13	*Sig* ← P_{DCP}.**substr**(0,320)	O(1)
14	*EncM* ← P_{DCP}.**substr**(160,256)	O(1)
15	*MAC* ← **Hash**(*Sig*).**substr**(0,*L*) ||*MAC*. **substr**(*L*,160-*L*)	O(1)
16	**If** *MAC* == **HMAC_SHA1**(*EncM*) **then**	O(*n*)
17	*data* ← **Dec_RC5** (*EncM*)	O(*r*)
18	*M* ← *data*.**substr**(0,*data*.**Length**()-64)	O(1)
19	*Knext* ← *data*.**substr**(data.**Length()**-64,64)	O(1)
20	**If Hash**$^{index+1}$(*Knext*) == *K0* **then**	O(*n*)
21	**If ECDSA_Ver**(*Sig*) **then**	O(n^3)
22	L_{prev} ← *L*	O(1)
23	*Index*++	O(1)
24	Packet_verified	O(1)
25	**Else**	O(1)
26	Drop_Message_Bad_Signature	O(1)
27	**End**	O(1)
28	**Else**	O(1)
29	Drop_Message_SessionKey_invalid	O(1)
30	**End**	O(1)
31	**Else**	O(1)
32	Drop_Message_puzzleSolution_invalid	O(1)
33	**End**	O(1)
34	**Else**	O(1)
35	Drop_Message_tag_invalid	O(1)
36	**End**	O(1)
Jumlah		O(n^3) + O(*L.n*)+O(n)+O(*r*)

adalah verifikasi tag pada baris ke-5 hingga 11 yaitu O($L.n$) dengan $L \leq l_{maks}$. Proses ini menggunakan fungsi hash yang memiliki kompleksitas O(n) pada setiap iterasinya. Kompleksitas komputasi tertinggi ketiga yaitu pada proses MAC yang terletak pada baris ke-16 dan fungsi hash yang terletak pada baris ke-20 dengan kompleksitas komputasi O(n). Kompleksitas komputasi tertinggi selanjutnya adalah proses dekripsi yaitu O(r) dengan r merupakan jumlah round dari algoritma RC5. Sehingga secara keseluruhan kompleksitas komputasi di sisi penerima sebesar O(n^3)+O($L.n$)+O(n)+O(r) dengan $L \leq l_{maks}$. Kompleksitas komputasi di sisi pengirim lebih besar dibandingkan dengan di sisi penerima karena proses pencarian solusi puzzle membutuhkan O($iter.r.n$) sedangkan verifikasi tag dan dekripsi pada penerima hanya membutuhkan O($L.n$)+O(r) dengan $L \leq l_{maks}$ dan $iter > l_{maks}$. Hal ini sesuai dengan ruang lingkup penelitian dengan pengirim merupakan pihak yang tidak memiliki keterbatasan dalam sumber daya sedangkan penerima berupa node sensor yang memiliki keterbatasan baik sumber daya maupun ruang penyimpanan.

Berdasarkan hasil simulasi dapat disimpulkan bahwa skema DCP menggunakan fungsi ambang eksponensial memiliki performansi paling baik dari 3 parameter optimasi yaitu rata-rata iterasi hash, standar deviasi dan maksimum iterasi hash. Sedangkan pada peluang tidak ditemukannya solusi, nilai dari fungsi ambang eksponensial lebih tinggi dari fungsi ambang linear dan Q1power1 tetapi lebih rendah dibandingkan Q1exp. DCP dengan fungsi ambang eksponensial menjamin ditemukannya solusi dengan peluang (1- P_{zs}) = 1- (1.73×10^{-13}). Berdasarkan hasil test bed, ruang penyimpanan dan RAM yang dibutuhkan untuk verifikasi skema DCP paling tinggi dibandingkan kedua skema puzzle lainnya (cipher puzzle dan DMSP). Akan tetapi peningkatan ini dibawah 3% dibandingkan cipher puzzle. Berdasarkan kompleksitas komputasi menggunakan Big O, peningkatan komputasi pada DCP bernilai O(n) dibandingkan dengan cipher puzzle dan bernilai O(r) dibandingkan dengan DMSP. Sedangkan, kompleksitas komputasi pada skema DCP secara keseluruhan lebih besar di sisi pengirim O(n^3)+O($iter.r.n$)+O(n) dibandingkan dengan di sisi penerima O(n^3)+O($L.n$)+O(n)+O(r) yang sesuai dengan ruang lingkup JSN. Hal ini

diimbangi dengan turunnya delay pada sisi pengirim, jaminan confidentiality yang ditawarkan oleh skema DCP dan peningkatan kompleksitas serangan sebesar 2^l.

5.3. Analisa keamanan DCP

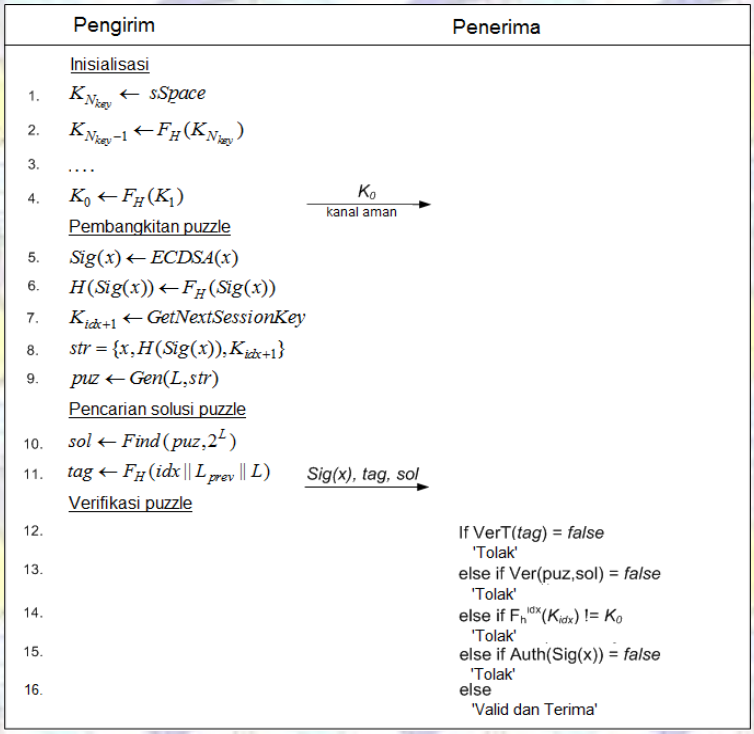
Terdapat tiga properti yang dibutuhkan untuk menjamin ketahanan terhadap serangan DoS yaitu puzzle *unforgeability*, puzzle *difficulty* dan kompleksitas identitas dari penguji. Puzzle *unforgeability* mensyaratkan bahwa tidak ada peretas yang dapat menghasilkan puzzle yang tampak valid sedangkan puzzle *difficulty* mensyaratkan bahwa tidak ada peretas yang dapat memecahkan satu teka-teki lebih cepat dari beberapa batasan yang ditentukan (Groza & Warinschi 2014). Konsep kunci sesi pada metode puzzle hampir menyerupai HashTrail dan HashInversion meskipun secara detail cukup berbeda. Definisi formal dari skema DCP akan dijelaskan sesuai dengan ide yang didapatkan dari penjelasan dari HashTrail dan HashInversion (Groza & Warinschi 2014).

DYNAMIC CIPHER PUZZLE diasumsikan $H : \{0,1\}^* \rightarrow \{0,1\}^k$ merupakan fungsi hash yang umum diketahui, $lSpace = [1, l_{maks}]$, $pSpace = \{0,1\}^*$ dan $sSpace = \{0,1\}^*$. DCP merupakan algoritma quadruple :

- Setup (1^k) merupakan algoritma untuk pendefinisian dengan input indeks (1^k) menghasilkan luaran $atr = K_{idx+1}$,
- Gen (l) merupakan algoritma pembangkit dengan sebuah input yang diambil secara random pada $x \in pSpace$, kemudian dijalankan perhitungan $H(Sig(x))$ dan menghasilkan luaran berupa tupple $puz = \{l, x, H(sig(x)), K_{idx+!}\}$
- Cari (puz, t_{maks}) yang merupakan algoritma penghasil solusi pada input tupple puz dan jumlah iterasi maksimum hash t_{maks} . Solusi puzzle didefinisikan sebagai $sol \in \{0,1\}^l$ yang akan ditemukan jika memenuhi $ETM_{Kidx}(x \parallel sol \parallel K_{idx+1})_{1..l} = H(sig(x))_{1..l}$
- Verifikasi (puz, sol) merupakan algoritma yang memanfaatkan tupple puz, sol sebagai input dan menghasilkan nilai benar atau valid jika memenuhi

$ETM_{Kidx}(x \parallel sol \parallel K_{idx+1})_{1..l} = H(sig(x))_{1..l}$ dan salah atau invalid jika sebaliknya.

Berdasarkan definisi formal dari DCP, telah diperlihatkan bahwa properti *unforgeability* telah dijalankan dengan penggabungan MAC dan puzzle menggunakan kunci sesi. Proses ini menjadikan sulit bagi peretas untuk mengetahui solusi puzzle tanpa mengetahui parameter kunci sesi dan indeks yang selaras dengan kekuatan puzzle. Properti kedua yaitu *difficulty preservation* yang dibagi menjadi dua jenis yaitu *fairness in answering*, yang diartikan jika peretas gagal memecahkan puzzle maka proses kegagalan itu dibatasi oleh *t* iterasi pada kekuatan puzzle yang konstan, dan *fairness in solving*, yang diartikan jika peretas tidak dapat menghasilkan solusi puzzle dalam iterasi yang lebih rendah dari t_{max} dan dibatasi oleh kekuatan puzzle yang konstan. DCP memiliki properti *fairness in answering* karena setiap iterasi pada pencarian solusi puzzle dibatasi oleh maksimum iterasi hash yang telah dibahas pada Bab 5.2.1. Jika peretas gagal menemukan solusi sesuai dengan batasan yang telah disebutkan maka parameter-parameter penyusunnya dimungkinkan telah berubah nilainya seperti kunci sesi,



Gambar 5.8 Protokol komunikasi pada skema DCP

indeks dan MAC. Properti terakhir mengenai kompleksitas identitas dari pengguna juga dipenuhi oleh DCP yang dibuktikan dengan protokol komunikasi dan ditunjukkan pada Gambar 5.8. Proses pembangkitan dan pencarian solusi puzzle dijalankan di sisi pengirim atau pengguna sehingga peretas tidak memiliki informasi tentang penyusun atau isi maupun solusi dari puzzle. Terlebih lagi identitas dari pengirim dilindungi dengan signature. Penelitian ini memanfaatkan metode ECDSA dengan panjang kunci baik publik maupun rahasia sebesar 20 byte. Metode ini memanfaatkan kunci asimetri dan aman untuk melindungi identitas dari pengirim.

Dari sudut pandang peretas, keamanan sebuah sistem bisa dilihat menggunakan *adversary model* (Xu et al. 2018; Wang et al. 2018). Model ini dirancang untuk mengetahui kemampuan peretas di dunia nyata. Kemampuan peretas *Adv* terdiri dari:

C-1. *Adv* mampu mengontrol pertukaran pesan antara pengirim, administrator, base station dan penerima atau node sensor

C-2. *Adv* dapat mendaftar semua kombinasi *tag* yang terdiri dari $D_{idx} \times D_l \times D_{lprev}$ dalam waktu polinomial, dengan D_{idx}, D_l, D_{lprev} adalah ruang untuk indeks, kekuatan puzzle dari puzzle yang dikirim, dan kekuatan puzzle paket sebelumnya secara berurutan.

C-3. *Adv* dapat mengambil alih node sensor dan mengetahui semua parameter yang tersimpan didalamnya.

C-4. *Adv* dapat membuat dan mentransmisikan puzzle palsu dan bertindak menyerupai pengguna yang valid

Kemampuan C-1 diartikan sebagai *Adv* berusaha mencuri dengar, mengintervensi, menghapus, mengubah atau mengirim ulang paket yang ditransmisikan antara pihak-pihak pada skema DCP. Kemampuan C-2 menunjukkan ruang dari parameter penyusun tag terbatas. Ukuran solusi puzzle yaitu 4 byte untuk mengurangi biaya komunikasi, sedangkan kekuatan puzzle maksimum yang pernah diujicobakan yaitu hingga 26 bit (Ning & Liu 2008) dan indeks merepresentasikan jumlah kunci sesi pada sebuah keychain. Oleh karena itu ruang dari peretas untuk menebak nilai tag yaitu $[N_{key} \times 26 \times 26]$.

Kemampuan C-3 merupakan akibat dari karakteristik node sensor yang umumnya diletakkan di daerah yang susah dijangkau. Kemampuan C-4 menunjukkan kemungkinan peretas mengirimkan puzzle palsu dengan kekuatan puzzle rendah dan berjumlah besar pada node sensor. Penyerangan semacam ini bisa menurunkan usia node sensor karena sibuk memverifikasi puzzle palsu.

Serangan probabilitas merupakan salah satu metode dari peretas yang mungkin terjadi pada skema DCP. Proses ini bisa dijalankan karena peretas dan pengirim memiliki peluang yang sama dalam menemukan solusi sebuah puzzle. Berdasarkan kemampuan C-1 dan C-2, peretas dapat mengintervensi paket $\{sig, tag, MAC, EncM\}$ yang ditransmisikan melalui tahap-tahap berikut :

Langkah 1: Ambil kombinasi $\left(idx_i^*, l_i^*, l_{prev}^*\right)$ dari kamus $D_{idx} \times D_l \times D_{lprev}$.

Langkah 2: Hitung $tag_i^* = H\left(idx_i^* + l_i^* + l_{prev}^*\right)$.

Langkah 3: Validasi kebeneran l_i^* bit pertama pada nilai tag_i^* dengan nilai tag yang diintervensi

Langkah 4: Ulangi langkah 1 sampai 3 hingga kombinasinya yang membentuk *tag* yang valid ditemukan.

Langkah 5: Hitung Mac_i^* dari hasil hash untuk penggabungan l_i^* bit pertama dari nilai sig yang diintervensi dan $(160 - l_i^*)$ bit sisa bit setelah tag_i^*

Langkah 6: Nilai prediksi dari pesan terenkripsi, kunci sesi dan solusi puzzle pada $EncM_i^*$ pada pesan yang diinterupsi setelah nilai Mac_i^* diketahui.

Langkah 7: Validasi kebenaran dari solusi puzzle dengan pengecekan apakah nilai ETM dari $EncM_i^*$ menggunakan kunci sesi $Kidx_i^*$ sama dengan Mac_i^*.

Langkah 1 hingga 3 bertujuan untuk mendapatkan informasi mengenai indeks dan kekuatan puzzle pada paket yang ditransmisikan sebelumnya. Nilai ini diperlukan untuk membuat tag palsu yang akan digunakan peretas. Proses ini terlihat sederhana dengan ruang kamus $D_{idx} \times D_l \times D_{lprev}$ yang kecil akan tetapi terjadinya tabrakan atau *collision* tidak terelakkan karena ketiga variabel tersebut memiliki range yang hampir sama. Asumsi jumlah kunci pada sebuah keychain

adalah 100 maka kombinasi dari parameter penyusun tag adalah 100 x 26 x 26 = 67.600. Jika kombinasi tag yang didapatkan sudah valid pun, nilainya tidak bisa langsung dipergunakan karena peretas *Adv* harus memastikan puzzle yang dibuat valid dengan meneruskan proses hingga langkah 7. Langkah 5 dan 6 digunakan untuk menentukan posisi *MAC* dan *EncM* pada paket yang diintervensi. Jika tag yang diprediksi salah maka otomatis nilai *MAC* dan *EncM* pun salah. Sebelum melanjutkan pada langkah yang terakhir *Adv* harus memprediksi atau mengasumsikan kunci sesi yang memiliki panjang 64 bit. *Adv* harus mencoba hingga 2^{64} kombinasi menggunakan metode *brute force*. Oleh karena itu tingkat kerandoman dari pembuatan *MAC* pun bergantung pada kunci sesi. Interval waktu yang dimiliki oleh peretas *Adv* dalam mencoba meretas paket yang diintervensi hanya berkisar antara 6.5 ms dengan payload memiliki panjang 102 byte dan bandwidth 250 kbps (Tan et al. 2013).

Serangan berikutnya yang mungkin dijalankan adalah DoS berbasis PKC dengan membanjiri node sensor dengan signature palsu. Komputasi verifikasi signature yang tinggi dimanfaatkan oleh peretas *Adv* pada node sensor yang memiliki sumber daya terbatas. Akan tetapi untuk dapat mengirim signature palsu *Adv* harus dapat melewati verifikasi tag dan skema puzzle. Detail dari pembuatan puzzle palsu telah dijelaskan pada ketujuh langkah yang telah dibahas sebelumnya. Jika ketujuh proses tersebut telah berhasil dilewati maka terdapat 2 kemungkinan yaitu kunci sesi diikuti pesan terenkripsi yang valid dan kunci sesi yang tampak valid (padahal kenyataanya invalid) diikuti dengan pesan terenkripsi yang invalid. Jika kondisi pertama terjadi maka peretas *Adv* dapat langsung menyerang node sensor menggunakan kunci publik yang didapat dari ruang penyimpanan node sensor atau ditebak dari komunikasi pengirim. Hal ini mungkin terjadi jika peretas *Adv* memiliki kemampuan C-3 dan C-4 dengan kunci sesi commitment dan kunci publik pengguna berhasil terkuak. Oleh karena itu node sensor yang telah diambil alih oleh peretas harus dapat dideteksi dan diasingkan dari JSN menggunakan metode-metode yang telah diteliti sebelumnya (Ping & Csiro 2015; Zheng et al. 2016; Al-Riyami et al. 2016). Langkah selanjutnya yang harus dilakukan adalah membangkitkan ulang kunci sesi dan

mengirimkan kembali commitment pada seluruh node sensor secara broadcast. Hal berbeda jika kemungkinan kedua yang terjadi maka peretas *Adv* memprediksi kombinasi parameter yang salah dari nilai tag, kunci sesi dan pesan terenkripsi sehingga signature palsu bisa diatasi oleh node sensor.

Penggunaan cipher puzzle dalam menghadapi serangan DoS berbasis PKC telah dioptimasi menggunakan fungsi ambang dengan rumus probabilistik. Tujuan pengembangan ini adalah menurunkan delay pada sisi pengirim dengan tetap mempertahankan aspek keamanan dan performansi pada JSN. Skema DCP menggunakan fungsi ambang linear dan eksponensial diusulkan. Hasil percobaan menunjukkan bahwa DCP yang menggunakan fungsi ambang eksponensial memiliki performansi lebih baik dibandingkan fungsi ambang linear maupun fungsi ambang dari metode eksperimental. Skema ini dapat menurunkan rata-rata iterasi hash hingga 94% dengan jaminan ditemukannya solusi hingga (1 - P_{zs}) = 1- ($1.728x10^{-13}$). Hal ini diikuti dengan standar deviasi dari kekuatan puzzle yang

Tabel 5.5 Pemanfaatan fungsi ambang pada skema puzzle dinamis

No	Fungsi ambang	Aplikasi JSN berbasis	Keterangan
1	Eksponensial (rumus probabilistik)	*event-driven*	waktu menjadi prioritas utama karena node sensor harus bertindak reaktif terhadap kejadian tidak lazim agar segera ditangani baik oleh administrator maupun otomatis dari sistem
2	Q1power1 (eksperimental)	query	keamanan informasi menjadi prioritas utama karena data yang ditransmisikan umumnya mengandung ukuran terhadap parameter-parameter tertentu yang dianggap penting bagi pengirim
3	Linear (rumus probabilistik)	*periodic-sensing*	informasi yang dikirimkan berupa hasil monitoring dari parameter yang belum tentu memiliki arti dan waktu transmisinya teratur

tinggi dan pengirimannya dijalankan secara eksplisit berupa tag yang disisipkan pada MAC membuat kompleksitas serangan semakin tinggi. Hal ini dibuktikan dengan *adversary model* yang dibangun, DCP memiliki pertahanan terutama pada serangan probabilitas dan DoS berbasis PKC. Pengembangan metode DCP memberi dampak pada penambahan kebutuhan sumber daya pada node sensor. Kebutuhan akan ruang penyimpanan dan RAM di sisi penerima dari hasil test bed mengalami peningkatan dengan presentase yang cukup kecil yaitu tidak lebih dari 3%.

Pemanfaatan fungsi ambang yang memiliki kelebihan dan kekurangan pada parameter-parameter optimasi dirangkum pada Tabel 5.5. Terdapat 3 varian fungsi ambang yang direkomendasikan untuk membatasi jumlah iterasi hash pada skema puzzle dinamis. Pemanfaatan ketiga fungsi ambang tersebut bisa disesuaikan dengan aplikasi JSN. DCP menggunakan fungsi ambang eksponensial bisa digunakan pada aplikasi JSN *event-driven.* Pada aplikasi tersebut, waktu menjadi prioritas utama karena node sensor harus bertindak reaktif terhadap kejadian tidak lazim agar segera ditangani baik oleh administrator maupun otomatis dari sistem. Sedangkan DCP menggunakan fungsi ambang Q1power1 bisa dimanfaatkan untuk aplikasi JSN berbasis query. Pada aplikasi tersebut, keamanan informasi menjadi prioritas utama karena data yang ditransmisikan umumnya mengandung ukuran terhadap parameter-parameter tertentu yang dianggap penting bagi pengirim. Kondisi tersebut sesuai dengan fungsi ambang Q1power1 karena jarak kekuatan puzzlenya mendekati kekuatan puzzle maksimum. Performansi dari parameter waktu dan keamanan informasi pada DCP menggunakan fungsi ambang linear berada di antara fungsi ambang eksponensial dan Q1power1 sehingga bisa dimanfaatkan pada aplikasi JSN berbasis *periodic-sensing*. Hal ini dikarenakan informasi yang dikirimkan berupa hasil monitoring dari parameter yang belum tentu memiliki arti dan waktu transmisinya teratur.

BAB 6
M-DCP dengan TinySet

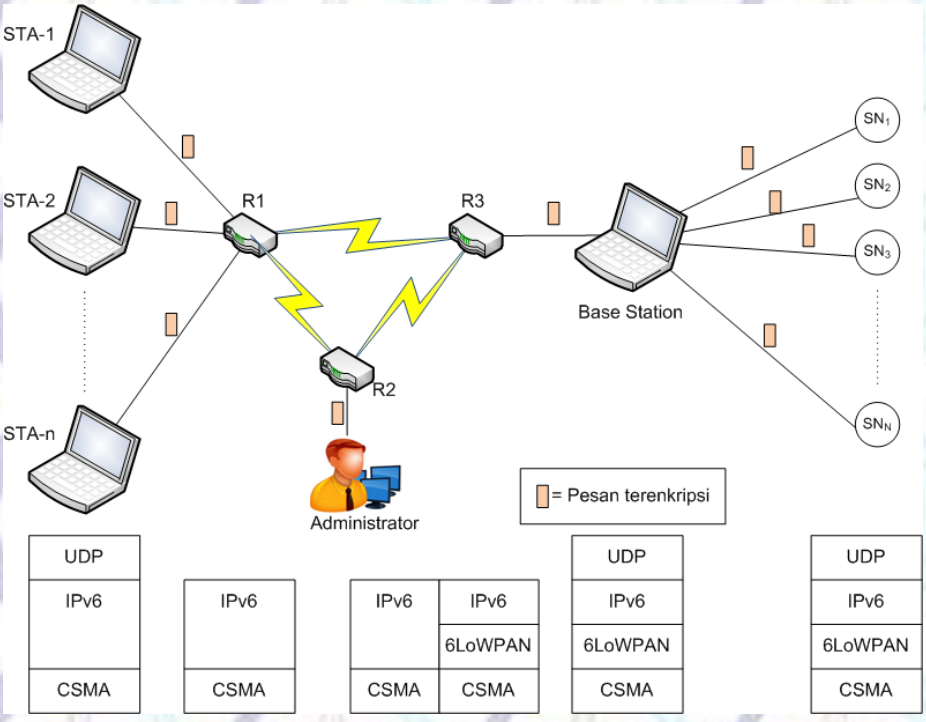
Implementasi DCP pada pengguna dengan jumlah yang besar tidak dapat begitu saja digunakan karena keterbatasan ruang penyimpanan node sensor dalam menyimpan kunci publik dari pengguna. M-DCP menggunakan TinySet yang telah dioptimalisasi menggunakan regularisasi diusulkan pada JSN dengan pengakses dalam jumlah besar.

6.1. Skenario uji coba M-DCP

Dalam menguji performansi metode yang diusulkan, penelitian ini menjalankan percobaan menggunakan NS-3 versi 3.20 dengan topologi yang ditunjukkan pada Gambar 6.1. Pengirim diberikan simbol STA-*N* dengan *N* merupakan banyaknya jumlah pengguna JSN yang diatur menggunakan

$$32.n_{ev} < N < 32.(n_{ev}+1), n_{ev} = 2,4,6,\ldots \quad (6.1)$$

Jumlah pengguna yang spesifik bertujuan untuk menunjukkan penggunaan keragaman jumlah bucket antara indeks 32 dengan 64. Jika rata-rata jumlah anggota pada setiap bucket mendekati satu ($\lambda \approx 1$) maka jumlah bucket pada indeks 64 ($l = 64$) hanya merupakan kelipatan indeks 32 ($l = 32$). Hal ini menjadi berbeda jika jumlah pengguna memenuhi Persamaan (6.1) maka jumlah bucket menjadi menjadi beragam antar kedua tipe indeks dan rata-rata anggota pada setiap bucket bisa menjadi lebih dinamis. Peran lain pada JSN adalah administrator yang memiliki sumber daya tinggi untuk menjalankan proses dengan komputasi yang tinggi. Tugas utama dari administrator adalah mengatur penambahan, pengurangan dan distribusi keanggotaan pada TinySet. Selain itu administrator juga bisa mengatur konfigurasi jaringan pada pengirim maupun penerima. Terdapat 3 router yang menghubungkan 3 jaringan untuk pengirim, penerima dan administrator yaitu R1, R2, dan R3. Sedangkan, pada sisi penerima jumlah node sensor penerima diasumsikan sejumlah 50 node sensor



Gambar 6.1 Topologi jaringan simulasi M-DCP

dalam sebuah level hop. Jumlah penerima atau node sensor bisa diganti dengan angka lain yang lebih merepresentasikan dunia nyata. Detail pemanfaatan lapisan OSI dapat dilihat lebih detail pada Gambar 6.1.

Pada simulasi yang dijalankan terdapat beberapa asumsi untuk mempermudah analisa dari implementasi metode yang diajukan yaitu:

a. Penelitian ini tidak fokus pada protokol *flooding* tertentu sebaliknya hanya menggunakan protokol sederhana.
b. Penelitian ini fokus pada node sensor yang tidak bergerak.
c. Node sensor mempunyai sumber daya yang cukup untuk menjalankan algoritma verifikasi digital signature dan dekripsi. Dalam rangka memperkuat asumsi ini, Tabel 6.1 menunjukkan penggunaan kriptografi pada platform JSN di dunia nyata baik untuk operasi autentikasi dengan kunci asimetri maupun dekripsi.
d. Base station, administrator dan pengirim memiliki kemampuan komputasi yang tidak terbatas dan tidak dapat diambil alih oleh peretas.

Tabel 6.1 Implementasi kriptografi pada JSN

Operasi kriptografi	Algoritma	Platform node sensor
Digital signature	Tiny ECC	MICAz 2, TelosB, Tmote Sky, and Imote2 (Liu & Ning 2008)
	ECDSA	MICAz and TelosB (Meulenaer et al. 2008) ; Arduino (Sethi et al. 2012; Xu et al. 2016)
Enkripsi	RC5	Arduino and Mica2 (Biswas et al. 2016)
	AES-128	MICAz and TelosB (Meulenaer et al. 2008)

6.2.Hasil dan pembahasan M-DCP

Terdapat 2 skema keanggotaan utama yang akan dianalisa yaitu CBF dan TinySet. TinySet dibagi berdasarkan jumlah indeks menjadi dua yaitu jumlah indeks 32 yang diberi nama TS32 dan jumlah indeks 64 yaitu dengan nama TS64. Sehingga total terdapat 3 skema keanggotaan yang akan diujicobakan yaitu CBF, TS32 dan TS64.

Parameter utama pada sumber daya sebuah node sensor adalah ruang penyimpanan dan waktu pemrosesan. Waktu eksekusi tersebut akan bernilai rendah jika TinySet cenderung menambahkan jumlah bucket dibandingkan menambah jumlah anggota pada setiap indeks. Namun pilihan ini memberikan dampak negatif pada ruang penyimpanan yang semakin meningkat jika jumlah bucket semakin besar. Oleh karena itu diperlukan pengaturan jumlah bucket melalui rata-rata jumlah anggota pada setiap bucket (λ_{act}) yang mendekati satu supaya penggunaan bucket menjadi optimal. Pengontrolan nilai λ_{act} melalui pembulatan jumlah bucket yang dibutuhkan untuk sejumlah anggota tertentu.

Tabel 6.2 Variasi parameter false positif pada TinySet

No	*cell*	λ	*FPR*
1	3	0.8	10^{-1}
2	6	0.64	10^{-2}
3	10	1.02	10^{-3}
4	13	0.82	10^{-4}
5	16	0.66	10^{-5}
6	20	1.05	10^{-6}

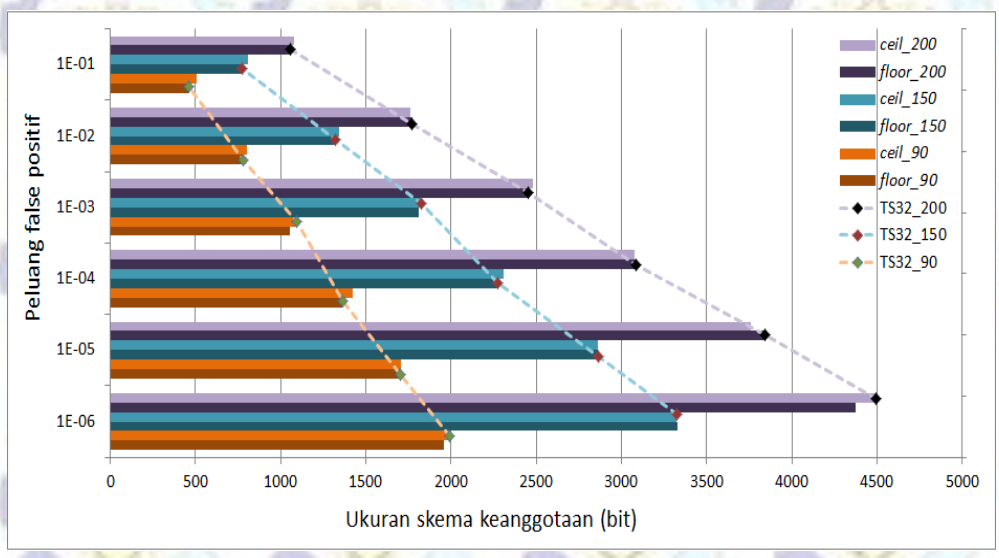
Pembulatan ini terdiri dari pembulatan keatas (*ceil*) dan kebawah (*floor*). Pemanfaatan pembulatan ini diimplementasikan pada 3 variasi pengguna maksimum yaitu 90, 150 dan 200. Nilai ini dipilih untuk mewakili Persamaan (6.1). Jumlah pengguna ini dilengkapi dengan 6 variasi nilai false positif (*FPR*) yang dapat diterima dan diperlihatkan pada Tabel 6.2. Rata-rata anggota pada setiap bucket secara aktual (λ_{akt}) dapat dihitung menggunakan variasi *FPR* sesuai Persamaan (2.17). Tujuan utamanya untuk mengetahui ukuran fingerprint (*cell*) jika λ_{akt} dikondisikan mendekati nilai 1. Nilai luaran λ_{akt} yang didapatkan tidak bulat menunjukkan angka 1 tetapi mendekati. Nilai inilah yang menentukan jumlah bucket dan pembulatannya akan keatas (menambah jumlah bucket dan mengurangi rata-rata anggota di setiap bucket) atau kebawah (tidak menambah jumlah bucket melainkan meningkatkan rata-rata jumlah anggota setiap bucket).

Tabel 6.3 Ruang penyimpanan pada optimasi TinySet

N	λ	TS32				TS64			
		B_{akt}	λ_{akt}	B_{cap}	TS_{size}(bit)	B_{akt}	λ_{akt}	B_{cap}	TS_{size}(bit)
90	0.8	3	0.94	31	474	2	0.7	45	490
	0.64	4	0.7	23	780	2	0.7	45	760
	1.02	3	0.94	31	1125	2	0.7	45	1120
	0.82	3	0.94	31	1404	2	0.7	45	1390
	0.66	4	0.7	23	1700	2	0.7	45	1660
	1.05	3	0.94	31	2055	2	0.7	45	2020
150	0.8	5	0.94	31	795	2	1.17	75	730
	0.64	7	0.67	22	1323	3	0.78	50	1248
	1.02	5	0.94	31	1880	2	1.17	75	1780
	0.82	5	0.94	31	2345	2	1.17	75	2230
	0.66	7	0.67	22	2863	3	0.78	50	2748
	1.05	5	0.94	31	3430	2	1.17	75	3280
200	0.8	7	0.89	29	1057	3	1.04	67	1002
	0.64	9	0.69	23	1773	4	0.78	50	1664
	1.02	6	1.04	34	2454	3	1.04	67	2409
	0.82	7	0.89	29	3087	3	1.04	67	3012
	0.66	9	0.69	23	3843	4	0.78	50	3664
	1.05	6	1.04	34	4494	3	1.04	67	4419

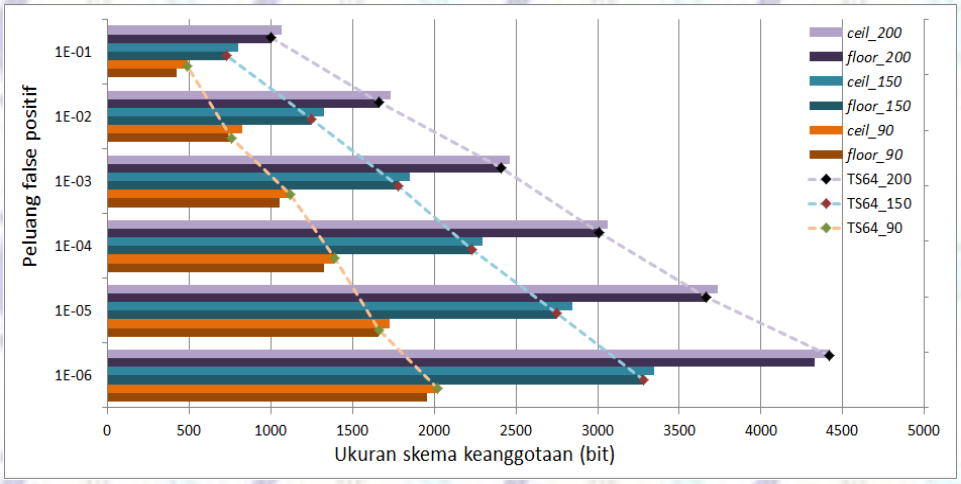
Hasil pemilihan pembulatan terbaik atau regularisasi TinySet, dari parameter-parameter yang telah ditentukan, diperlihatkan pada Tabel 6.3.

Nilai dari bucket aktual (B_{akt}) dibulatkan sesuai dengan regularisasi pada Persamaan (3.11). Sedangkan nilai λ_{akt} lebih mendekati satu dibandingkan dengan λ baik pada TS32 maupun TS64. Jika nilai λ_{akt} antara TS32 dan TS64 sama maka ruang penyimpanan TS64 lebih rendah dibandingkan dengan TS32. Hal ini dikarenakan TS64 cenderung menambah anggota pada setiap bucket dan meminimalisir penambahan bucket itu sendiri seperti yang terlihat pada N=90 dengan λ =0.64, λ =0.66 dan N=200 dengan λ =1.02, λ =1.05. Jika nilai λ_{akt} antara TS32 dan TS64 dibawah 1 maka ruang penyimpanan dari TinySet yang lebih dekat dengan 1, lebih rendah dibandingkan dengan TinySet yang lainnya. Kondisi ini terjadi pada N=90 dengan λ =0.8, λ =1.02, λ =0.82, λ =1.05 ; N=150 dengan λ =0.64, λ =0.66; dan N = 200 dengan λ = 0.64, λ = 0.66. Namun berbeda jika salah satu nilai λ_{akt} bernilai diatas 1, TinySet yang memiliki λ_{akt} diatas satu lah yang memiliki ruang penyimpanan lebih rendah. Kondisi ini terjadi pada N=150 dengan λ =0.8, λ =1.02, λ =0.82, λ =1.05 dan N=200 dengan λ =0.8, λ =0.82.

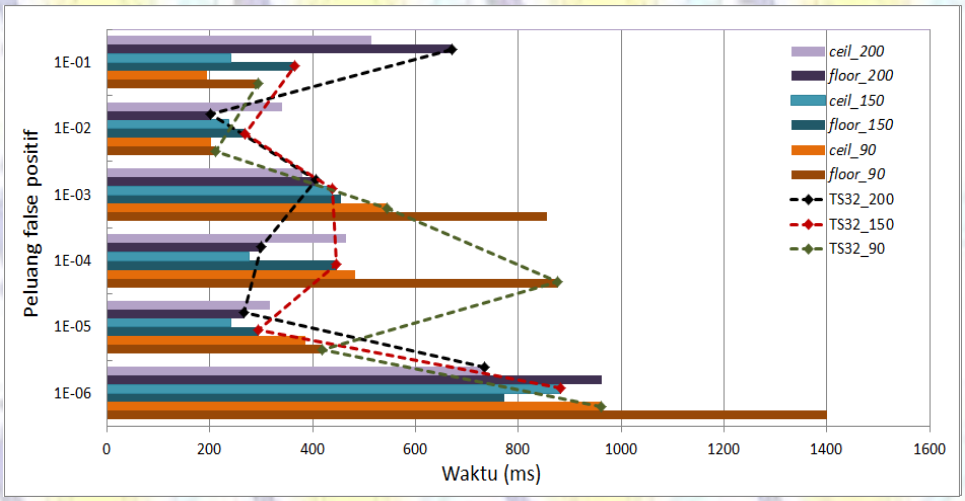


(a) Ruang penyimpanan pada TS32

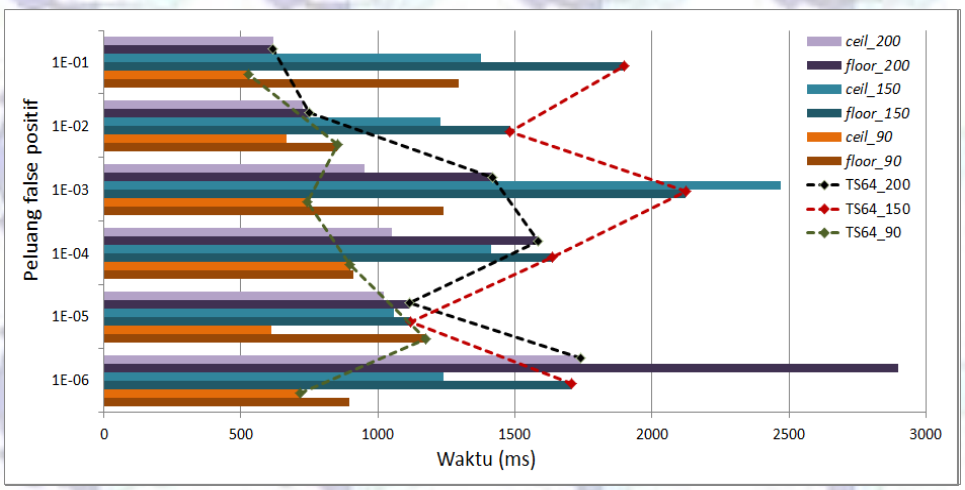
Gambar 6.2 Konsumsi sumber daya pada TinySet dan Regularisasi TinySet



(b) Ruang penyimpanan pada TS64



(c) Waktu penambahan anggota pada TS32



(d) Waktu penambahan anggota pada TS64

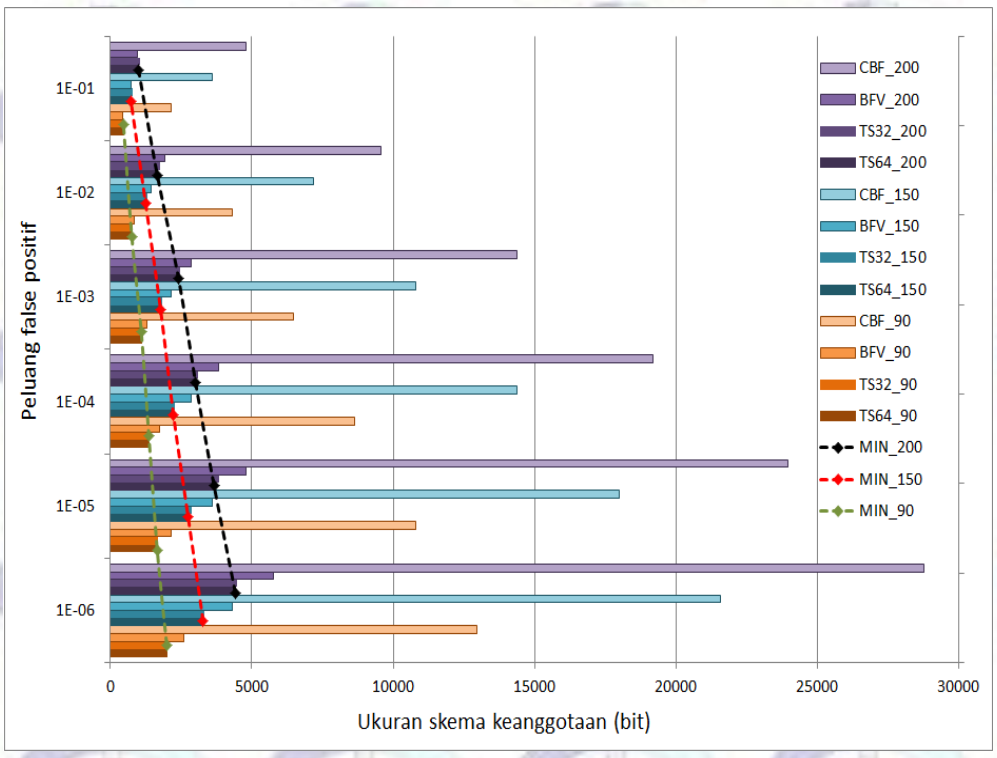
Gambar 6.2 Konsumsi sumber daya pada TinySet dan Regularisasi TinySet

Analisa berikutnya yaitu mengenai hubungan antara ruang penyimpanan dan waktu eksekusi pada saat penambahan pengguna pada skema keanggotaan TinySet. Hasil dari kedua parameter tersebut pada TS32 dan TS64 ditunjukkan pada Gambar 6.2. Diagram bar ini menunjukkan sumber daya yang dibutuhkan pada TinySet dengan pembulatan keatas (*ceil*), pembulatan kebawah (*floor*) dan hasil regularisasi (*TS*). Percobaan dilakukan pada 3 varian jumlah anggota, 6 varian peluang false positif dan 2 tipe indeks dari TinySet. Diamon hijau menunjukkan hasil regularisasi TinySet pada jumlah anggota 90, diamon merah pada jumlah anggota 150 dan diamon hitam pada jumlah anggota 200. Gambar 6.2(a) dan (b) menunjukkan ruang penyimpanan pada jumlah indeks 32 bit dan 64 bit. Diagram tersebut menunjukkan semakin rendah peluang false positif maka semakin tinggi ruang penyimpanan yang dibutuhkan untuk menampung fingerprint anggota. Ruang penyimpanan yang dibutuhkan *ceil* pada umumnya lebih besar dibandingkan *floor*. Selisih antara pembulatan keatas dan kebawah berkisar antara 52 hingga 100 bit pada TinySet dengan indeks 64 bit. Sedangkan pada indeks 32 bit jarak selisihnya lebih tinggi yaitu berkisar antara 1 hingga 119 bit. Akan tetapi terdapat beberapa kondisi yaitu ruang penyimpanan *floor* lebih besar daripada *ceil* yaitu pada N=150 dengan FPR=10^{-6} dan N=200 dengan FPR=10^{-2}, 10^{-4}, 10^{-5}. Gambar 6.2(a) dan (c) menunjukkan keterhubungan antara ruang penyimpanan dengan waktu yang dibutuhkan untuk menambahkan pengguna pada TinySet dengan indeks 32 bit. Jika hasil dari regularisasi TinySet adalah pembulatan kebawah *floor* maka TS32 mengorbankan waktu eksekusi dan lebih mengutamakan ruang penyimpanan. Kondisi ini terjadi pada N=90 dengan FPR 10^{-5} yang diartikan bahwa TS32 mengorbankan waktu eksekusi sebesar 31 ms dalam rangka menghemat ruang penyimpanan sebesar 5 bit. Sebaliknya jika regularisasi menghasilkan pembulatan ke atas maka TS32 mengorbankan ruang penyimpanan dibandingkan dengan waktu eksekusi. Kondisi ini terjadi pada N=90 dengan FPR 10^{-6} yang diartikan bahwa TS32 mengorbankan ruang penyimpanan sebesar 64 bit sehingga waktu eksekusi penambahan pengguna lebih hemat 1157 ms. Hal serupa juga terjadi pada TS64 sebagai contoh hasil pembulatan keatas pada N=200 dengan FPR=10^{-6} yaitu TS64 mengorbankan 89 bit ruang

penyimpanan dan lebih memilih waktu eksekusi yang lebih hemat sebesar 1157 ms. Sedangkan pembulatan kebawah sebagai hasil regularisasi TinySet pada TS64 terjadi pada N=200 dengan FPR=10^{-5} yaitu waktu eksekusi penambahan anggota yang dikorbankan sebesar 94 ms dan penghematan ruang penympanan yang didapat sebesar 71 bit.

Pembahasan berikutnya mengenai sumber daya yang dibutuhkan oleh pihak-pihak terkait pada komunikasi JSN yaitu administrator, pengguna sebagai pengirim, maupun node sensor sebagai penerima Pada administrator, parameter yang akan dianalisa adalah ruang penyimpanan dan waktu eksekusi penambahan pengguna, sedangkan pada pengguna dibutuhkan informasi mengenai waktu yang dibutuhkan untuk memproses pesan dalam mekanisme enkapsulasi. Kedua peran tersebut memiliki sumber daya komputasi yang tinggi, namun pada sisi penerima yaitu node sensor diperlukan informasi mengenai waktu yang dibutuhkan untuk verifikasi mengingat keterbatasan sumber daya menjadi tantangan utama.

Proses penambahan pengguna pada administrator bergantung pada skema keanggotaan yang digunakan dan ditunjukkan pada Gambar 6.3. Skema



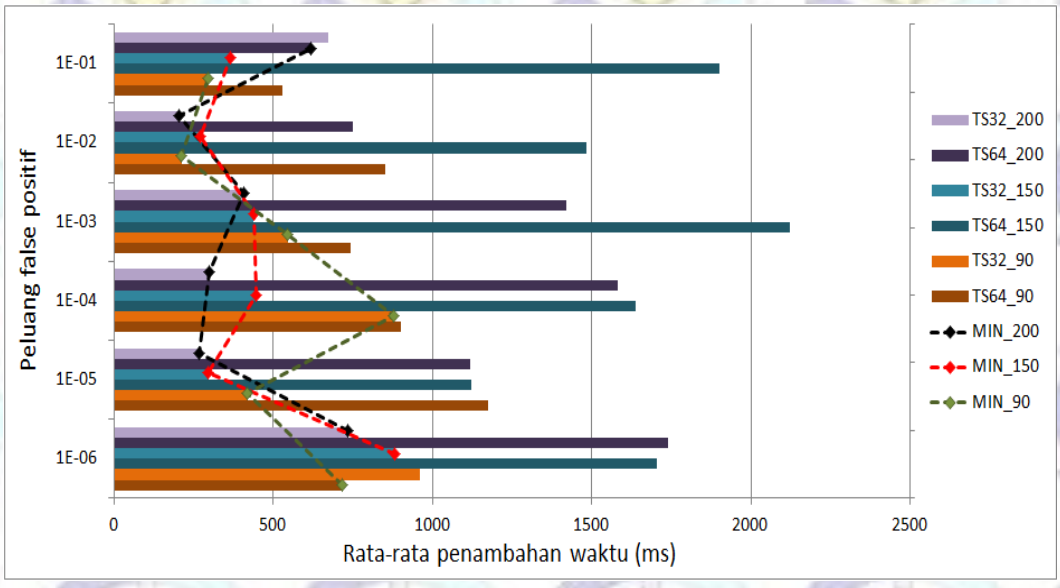
Gambar 6.3 Ruang penyimpanan pada administrator

Tabel 6.4 Perbandingan ruang penyimpanan untuk penambahan pengguna pada skema keanggotaan probabilistik

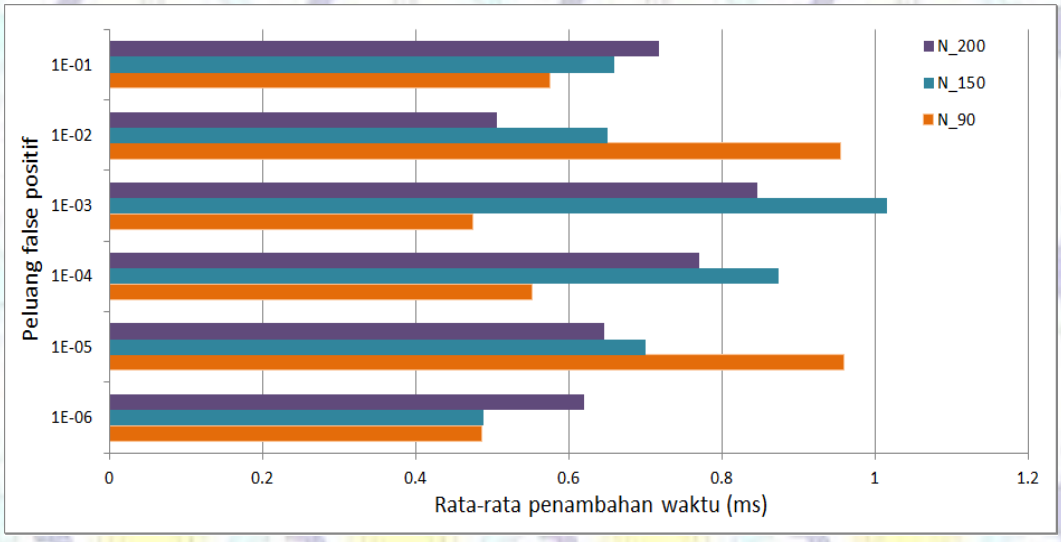
No	Jumlah pengguna	FPR	TS64 (bit)	TS32 (bit)	BFV (bit)	CBF (bit)
1	90 (DAS=15840 bit / 1980 byte)	10^{-1}	490	462	432	2160
2		10^{-2}	760	780	863	4315
3		10^{-3}	1120	1092	1294	6470
4		10^{-4}	1390	1362	1726	8630
5		10^{-5}	1660	1700	2157	10785
6		10^{-6}	2020	1992	2588	12940
7	150 (DAS=26400 bit / 3300 byte)	10^{-1}	730	775	719	3595
8		10^{-2}	1248	1323	1438	7190
9		10^{-3}	1780	1825	2157	10785
10		10^{-4}	2230	2275	2876	14380
11		10^{-5}	2748	2863	3595	17975
12		10^{-6}	3280	3325	4313	21565
13	200 (DAS=35200 bit / 4400 byte)	10^{-1}	1002	1057	959	4795
14		10^{-2}	1664	1773	1917	9585
15		10^{-3}	2409	2454	2876	14380
16		10^{-4}	3012	3087	3834	19170
17		10^{-5}	3664	3843	4793	23965
18		10^{-6}	4419	4494	5751	28755

keanggotaan yang dibandingkan terdiri dari 4 yaitu regularisasi TinySet pada indeks 32 yaitu TS32, pada indeks 64 yaitu TS64, BFV dan CBF. Penggunaan TS64 membutuhkan ruang penyimpanan paling kecil dari keseluruhan kombinasi parameter peluang false positif pada 2 dari 3 varian banyaknya anggota yaitu untuk N=150 juga N=200. Namun pada jumlah anggota 90, TS32 memiliki ruang penyimpanan paling kecil dibandingkan kedua varian lainnya kecuali pada FPR=10^{-5} dan 10^{-2}. Secara keseluruhan pemanfaatan TinySet dengan indeks 32 maupun 64 menghemat ruang penyimpanan hingga 77% atau berkisar antara 1698 hingga 24261 bit dibandingkan dengan CBF. Sedangkan antara TinySet indeks 32 dengan 64 bit perbedaannya berkisar antara 28 hingga 115 bit atau 6.2% dan berlaku di semua varian peluang false positif. Bahkan TinySet 64 lebih hemat hingga 21% dibandingkan dengan BFV dengan selisih berkisar antara 568 hingga

1332 bit. Tabel perbandingan ruang penyimpanan pada skema keanggotaan ditunjukkan pada Tabel 6.4. Berdasarkan diagram batang pada Gambar 6.3 yang diperjelas dengan Tabel 6.4 dapat dilihat bahwa semakin tinggi jumlah pengguna maka semakin tinggi selisih ruang penyimpanan yang dibutuhkan antara pemanfaatan skema keanggotaan (TS32, TS64, BFV, CBF) dan tanpa penggunaan skema keanggotaan (DAS). Hal ini terlihat pada jumlah pengguna 90, 150 dan 200 dengan selisih antara CBF dan DAS yang bernilai 2900, 4835, dan 6425 bit pada false positif paling rendah yaitu 10^{-6}. Perbedaaan tersebut kurang lebih bernilai 18%. Selisih dari penghematan ini tentu akan lebih besar jika dibandingkan antara TS64 dan DAS yaitu sekitar 87% dengan selisih nilai sebesar 13820, 23120, dan 30781 bit pada jumlah pengguna 90, 150 dan 200 untuk false positif paling rendah yaitu 10^{-6}. Bahkan, jika TS64 dibandingkan dengan BFV maka ruang penyimpanan tetap bisa dihemat yaitu sebesar 21-23%. Oleh karena itu, kebutuhan ruang penyimpanan paling tinggi terjadi pada DAS yang diikuti dengan pemanfaatan skema keanggotaan menggunakan CBF, BFV, TS32 dan TS64 secara berurutan. Hasil terbaik didapatkan oleh TS64 baik pada jumlah pengguna 90, 150 maupun 200 karena kebutuhan ruang penyimpanannya paling rendah dibandingkan dengan DAS maupun skema keanggotaan yang lain.



(a) TinySet



(b) CBF

Gambar 6.4 Waktu eksekusi penambahan anggota

Rata-rata waktu eksekusi dari penambahan pengguna di pihak administrator ditunjukkan pada Gambar 6.4. Percobaan pada setiap varian jumlah anggota dan peluang false positif diulang sebanyak 200 kali untuk memenuhi jumlah minimum sampel. Waktu eksekusi penambahan pengguna pada TinySet ditunjukkan pada Gambar 6.4(a). Sebagian besar waktu yang dibutuhkan TS32 lebih kecil dibandingkan TS64 kecuali pada N=90 dengan FPR =10^{-6} dan N=200 dengan FPR=10^{-1}. Rata-rata waktu yang dibutuhkan untuk penambahan anggota pada TS32 adalah sekitar 477 ms, sedangkan pada TS64 yaitu sekitar 1228 ms atau lebih dari 2 kali lipat. Hal ini dikarenakan kompleksitas TinySet akan meningkat jika nilai indeks dan jumlah anggota pada tiap indeks lebih tinggi. TS32 cenderung mereduksi jumlah anggota pada tiap indeks dan menambah jumlah bucket. Hal ini dapat menurunkkan kompleksitas dan waktu pemrosesan akan tetapi dibutuhkan ruang penyimpanan yang tinggi. Waktu eksekusi penambahan pengguna pada CBF ditunjukkan pada Gambar 6.4(b). Skema keanggotaan CBF dipisah dari TinySet dikarenakan selisih waktu yang tinggi antara keduanya. Rata-rata waktu yang dibutuhkan untuk menambahkan satu pengguna pada CBF adalah sekitar 0.7 ms. Hal ini dikarenakan CBF hanya membutuhkan operasi hash sebanyak k pada setiap penambahan pengguna dengan waktu eksekusi sebuah hash berkisar antara 0.1-0.2 ms. Waktu yang dibutuhkan

Tabel 6.5 Waktu pemrosesan pada pembungkusan pesan

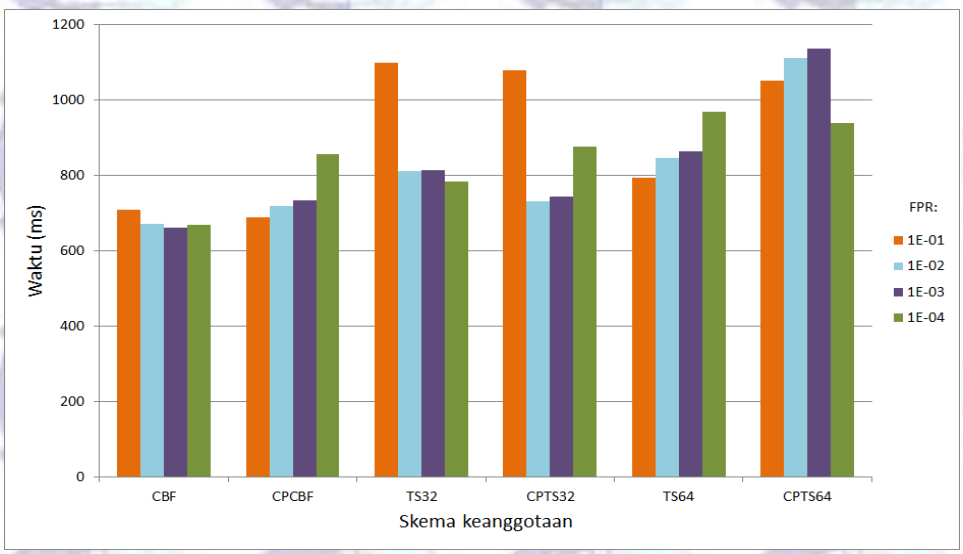
BAS		M-DCP		
Indeks	Waktu eksekusi (ms)	*l*	*nH*	Waktu eksekusi (ms)
1	918	16	340518	58742
2	908	16	338745	58742
3	975	15	344893	58939
4	893	12	354383	61618
5	1055	16	339795	60601
6	1226	14	351108	58602
7	1055	16	338726	57529
8	970	11	355839	57260
9	944	14	355579	59659
10	1087	14	350114	60081

oleh CBF untuk menjalankan operasi hash pada penambahan pengguna berkisar antara 0.5-1 ms karena *k* bernilai 5. Berdasarkan kedua parameter pada penambahan pengguna di sisi administrator, dapat disimpulkan bahwa TS32 dan TS64 memiliki kelebihan dalam ruang penyimpanan sedangkan CBF memiliki kompleksitas rendah yang berdampak pada waktu pemrosesan yang lebih kecil.

Pihak pengirim atau pengguna memiliki peranan penting sebagai inisiator dalam proses komunikasi dengan JSN. Proses utama yang harus dijalankan adalah pembungkusan pesan dengan beberapa parameter tambahan dalam rangka menjaga keamanan pesan maupun identitas pengirim. Jika pesan yang dikirim bersifat rahasia maka metode enkripsi harus ditambahkan. Pada M-DCP terdapat 5 metode yang perlu ditambahkan pada sisi pengirim sesuai dengan blok diagram pada Gambar 3.8. Waktu paling tinggi dibutuhkan untuk proses pencarian solusi puzzle (proses nomer 4). Akan tetapi skema M-DCP menjamin jumlah iterasi hash pencarian solusi tidak lebih dari 357795 kali. Angka ini didapatkan dari Persamaan (3.10) yang menggunakan maksimum kekuatan puzzle 24 bit dan fungsi ambang eksponensial (Afianti et al. 2018). Jika maksimum waktu yang dibutuhkan untuk sebuah iterasi hash berkisar antara 0.2 ms maka maksimum total waktu pencarian puzzle sekitar 71559 ms. Perbandingan antara 2

metode pembungkusan pesan yaitu BAS dan M-DCP ditunjukkan pada Tabel 6.5. Rata-rata waktu yang dibutuhkan pada metode BAS untuk memproses sebuah pesan menjadi paket utuh adalah sekitar 1003 ms, sedangkan M-DCP membutuhkan waktu sekitar 59177 ms dengan rata-rata kekuatan puzzle sebesar 14 bit. Perbedaan antara kedua metode pembungkusan tersebut yaitu sekitar 98.3%. Akan tetapi, M-DCP memiliki kelebihan pada pesan yang telah terenkripsi dan ketahanan terhadap serangan DoS. Skema puzzle dan kunci sesi meningkatkan keamanan JSN terutama pada DoS berbasis PKC yang tidak dimiliki oleh metode BAS.

Peran terakhir yang akan dibahas yaitu kompleksitas yang ditunjukkan dengan waktu pemrosesan pada sisi penerima atau node sensor. M-DCP memiliki 6 tahap verifikasi yang kesemuanya memiliki kompleksitas yang rendah. Waktu yang dibutuhkan untuk verifikasi pesan pada node sensor ditunjukkan pada Gambar 6.5. 'CPCBF', 'CPTS32' dan 'CPTS64' menunjukkan penggunaan M-DCP yang diintegrasikan dengan skema keanggotaan CBF, TS32 dan TS64, secara berurutan. CBF tanpa M-DCP memiliki waktu verifikasi yang paling kecil dari metode lainnya yaitu sekitar 677 ms. Sedangkan CPTS64 memiliki waktu verifikasi paling tinggi yaitu sekitar 1060 ms. Jumlah mekanisme verifikasi pada CPTS32 dan CPTS64 lebih tinggi dibandingkan CBF akan tetapi perbedaan waktu



Gambar 6.5 Waktu verifikasi pada node sensor

verifikasi antara dua jenis skema keanggotaan tersebut hanya bernilai sekitar 382 ms atau 36%. Nilai tersebut masih dibawah setengah detik. Selisih perbedaan waktu tersebut dinilai masih wajar. Parameter lainnya yang berbeda antara CPCBF dengan CBF adalah tag, puzzle dan kunci sesi. Waktu verifikasi untuk ketiga parameter tersebut kurang dari 0.5 ms. Hal inilah yang menjadi alasan mengapa perbedaan waktu verifikasi antara dua mekanisme autentikasi dan skema keanggotaannya cukup kecil.

Penelitian ini fokus pada skema autentikasi multiuser yang berbiaya rendah pada JSN yang memiliki keterbatasan dalam sumber daya. Selain itu, keamanan komunikasi pada JSN harus terjaga mengingat terdapat informasi sensitif yang harus dipertukarkan. Oleh karena itu, skema M-DCP dilengkapi dengan TS64 diusulkan. Meskipun dibutuhkan waktu yang lebih tinggi pada proses penambahan pengguna dibandingkan dengan metode TS32, akan tetapi proses penambahan ini dijalankan di sisi administrator. Sebagaimana yang dijelaskan sebelumnya, administrator memiliki sumber daya tinggi. JSN hanya butuh menyimpan TinySet dengan ruang penyimpanan yang rendah dan proses verifikasi yang rendah komputasi.

6.3.Analisa keamanan M-DCP

Sebelum membahas detail mengenai kemampuan M-DCP dalam mempertahankan keamanan informasi, terlebih dahulu dijelaskan asumsi kemampuan dari penyerang. Kemampuan pada penyerang untuk DCP dengan pengguna tunggal bisa disetarakan penggunaan skema keanggotaan DAS yang telah dibahas pada Bab 5.3. Akan tetapi penyerang, yang disebut *Adv*, pada autentikasi JSN pengguna jamak lebih komplek dengan kemampuan seperti yang dibahas berikut:

C-1 *Adv* memiliki kemampuan untuk mengontrol pertukaran informasi antara administrator, base station dan node sensor

C-2 *Adv* memiliki kemampuan untuk mendapatkan seluruh kombinasi dari kunci publik dan identitas pengguna yaitu $D_{PubK} \times D_{Uid}$ dengan waktu polinomial

dengan D_{PubK} adalah ruang kemungkinan untuk kunci publik sedang D_{Uid} adalah ruang kemungkinan untuk identitas pengguna.

C-3 *Adv* memiliki kemampuan untuk mengambil alih dan mengekspos parameter yang tersimpan pada node sensor.

C-4 *Adv* memiliki kemampuan untuk membuat dan mentransmisikan pesan palsu dalam jumlah besar.

C-5 *Adv* memiliki kemampuan untuk membuat dan mentransmisikan kunci publik dan identitas pengguna yang palsu seakan-akan sama dengan pengguna yang valid.

Kemampuan C-1 diartikan bahwa *Adv* dapat mengintervensi, memodifikasi dan menguping pesan yang ditransmisikan oleh pihak-pihak yang berhubungan dengan komunikasi JSN. Setiap modifikasi pesan dapat langsung diketahui oleh node sensor baik melalui skema tag, skema keanggotaan maupun skema puzzle. Ketiga mekanisme ini memiliki tingkat keamanan dan tujuan masing-masing. Tingkat keamanan dari verifikasi tag telah dibahas pada Bab 5.3. Perbedaannya pada M-DCP ukuran tag tetap sehingga dapat meningkatkan kompleksitas serangan dari 2^l menjadi 2^{24}. Sedangkan tingkat keamanan pada skema keanggotaan TinySet bergantung pada peluang false positif. Semakin rendah nilai false positif maka semakin tinggi kompleksitas serangan dan semakin panjang ukuran fingerprint yang harus tersimpan pada sisi penerima atau node sensor. Pada penelitian ini, skema keanggotaan dengan false positif terendah bernilai 10^{-6} membutuhkan 4419 bit atau 553 byte ruang penyimpanan pada TS64.

Tabel 6.6 Kemungkinan kombinasi pada serangan brute force

No	Proses verifikasi	BAS	M-DCP
1	Indeks	2^{16}	-
2	Tag	-	2^{24}
3	Keanggotaan	2^{176}	2^{176}
4	Puzzle	-	2^{32}
5	Kunci sesi	-	2^{64}
6	Signature	2^{160}	2^{160}
Total		$9.174x10^{105}$	$1.86x10^{137}$

Peluang kegagalan dari penyerang jika mengujicobakan identitas palsu yaitu sebesar $1\text{-}10^{-6}$. Tingkatan keamanan dari skema puzzle bergantung pada maksimum kekuatan puzzle (l_{maks}). Meskipun pada M-DCP ukurannya kecil, akan tetapi kekuatan puzzle dilindungi dengan kunci sesi. Terlebih lagi isi pesan, solusi puzzle dan kunci sesi berikutnya dikirim dalam kondisi terenkripsi. Jika *Adv* memodifikasi hanya pada bagian pesan pada paket yang ditransmisikan maka paket palsu tersebut bisa melewati ketiga mekanisme keamanan. Akan tetapi tidak bisa melewati verifikasi signature karena pesan yang datang dilindungi oleh kunci privat dari pengirim. Penyerang harus mencoba 2^{160} iterasi sesuai dengan panjang kunci pada ECDSA yang menghasilkan 20 byte signature.

Kemampuan C-2 sama dengan menjalankan serangan brute force yang berusaha untuk mengetahui isi dari TinySet. Jumlah iterasi yang dibutuhkan untuk serangan brute force secara keseluruhan ditunjukkan pada Tabel 6.6. Kunci publik dan identitas pengguna memiliki kemungkinan kombinasi sebesar $D_{PubK} \times D_{Uid} = 2^{160} \text{ x } 2^{16} = 2^{176}$ iterasi. Dari sudut pandang penyerang peluang keberhasilan untuk berhasil menembus skema keanggotaan adalah *FPR*. Jika setiap iterasi membutuhkan waktu 0.1 ms dan $FPR = 10^{-6}$ maka *Adv* membutuhkan waktu sebesar $(2^{176} \text{ x } 0.1 \text{ x } 10^{-3}) : (3600 \text{ x } 24) = 1.108 \text{ x } 10^{44}$ hari untuk mencoba semua kemungkinan dengan tingkat keberhasilan bisa melewati verifikasi TinySet sebesar 10^{-6}.

Kemampuan C-3 diartikan bahwa penyerang *Adv* mengetahui segala informasi yang tersimpan pada node sensor. Parameter tersebut yaitu indeks, kekuatan puzzle pada indeks sebelumnya, TinySet, kunci komitmen, dan kunci rahasia dari node sensor. Akan tetapi untuk dapat menyerang JSN, *Adv* tetap harus membuat puzzle baru yang valid. Proses pembuatan ini dilindungi kunci sesi. Terkuaknya kunci komitmen, tidak lantas membongkar keseluruhan rantai kunci sesi maupun kunci sesi selanjutnya. Seperti yang telah dibahas pada Bab 3.2.2, bahwa metode yang digunakan untuk membangun serangkaian kunci sesi adalah keychain arah backward dan tidak memungkinkan kunci komitmen untuk menjalankan fungsi hash yang bersifat *irreversible*. Kunci komitmen terletak pada indeks pertama dari keychain dan hanya digunakan untuk verifikasi berdasarkan

kunci sesi selanjutnya dan indeks yang diterima. Akan tetapi kunci komitmen tidak dapat mengetahui isi dari kunci sesi selanjutnya. Oleh karena itu jika node sensor diambil alih, skema puzzle dan kunci sesi tetap aman. Keamanan dari kunci sesi bisa dibuktikan dengan *Random Oracle Model* (ROM). Penelitian ini memanfaatkan metode kontradiksi berdasarkan ide dari penelitian sebelumnya (Das 2016; Han et al. 2018). Misal random oracle black box disebut dengan $RO(.)$ yang menerima input unik dan menghasilkan luaran secara random dan bersifat uniform. Fungsi ini akan memetakan tanpa terkecuali semua input yang diterima menjadi output spesifik, dengan nilai masukan yang sama akan menghasilkan luaran yang sama pula. Secara teoritis skema M-DCP aman dalam menghadapi penyerang, *Adv*, dalam pengambilalihan kunci sesi antara pengguna dan JSN dengan asumsi $RO(.)$ bertindak sebagai fungsi hash kriptografi. Hal ini dibuktikan dengan Algoritma 6.1 dengan *Adv* bertujuan untuk menginterupsi komunikasi antara pengguna dengan JSN dan mendapatkan dua kunci sesi secara berurutan. Asumsi *Adv* dapat mengakses $RO(.)$ dalam mencapai tujuannya. Sebagaimana telah dijelaskan pada Bab 3.3, kunci sesi pada sebuah komunikasi dilindungi oleh metode enkripsi yang memanfaatkan kunci sesi yang telah dikirim pada komunikasi sebelumnya. Peluang keberhasilan jika Algoritma 6.1 dijalankan

Algoritma 6.1 ROM pada keamanan kunci sesi

Input: Pesan yang diintervensi dari komunikasi antara pengguna dan JSN
Output: Rangkaian kunci sesi
Intervensi pesan dari $u_{(i)}$ ke JSN $\{PubK_{(i)}, Uid_{(i)}, Sig_{(i)}, M'_{(i)}, K'_{(i)}, P'_{(i)}, tag_{(i)}\}$
Intervensi pesan dari $u_{(i+1)}$ ke JSN $\{PubK_{(i+1)}, Uid_{(i+1)}, Sig_{(i+1)}, M'_{(i+1)}, K'_{(i+1)}, P'_{(i+1)}, tag_{(i+1)}\}$
$\mathrm{RO}(tag_{(i)}) = idx_{(i)}$
$\mathrm{RO}(tag_{(i+1)}) = idx_{(i+1)}$
$\mathrm{RO}(K_{(1)}) = K_{(0)}$
$\mathrm{RO}(K_{(i)}) = K_{(i-1)}$
$\mathrm{Dec\_RC5}_{K_{(i-1)}}(K'_{(i)}) = K_{(i)}$
$\mathrm{Dec\_RC5}_{K_{(i)}}(K'_{(i+1)}) = K_{(i+1)}$
if $\mathrm{RO}(K_{(i+1)}) = K_{(i)}$ and $\mathrm{RO}(K_{(0)}, idx_{(i)}) = K_{(i)}$ and $\mathrm{RO}(K_{(0)}, idx_{(i+1)}) = K_{(i+1)}$ **then**
    Menerima $K_{(i)}$ dan $K_{(i+1)}$ sebagai kunci sesi yang valid
    Return 1 (Berhasil)
else
    Return 0 (Gagal)
end

yaitu $Adv_{sukses} = |\Pr[Algorithm6.1 = 1] - 1|$. Fungsi *advantage* merupakan sebuah pengukuran untuk mengetahui keberhasilan peretas dalam menyerang algoritma kriptografi dengan cara lain yang berbeda dengan versi ideal algoritma pembangkitannya (Katz & Lindell 2008). Fungsi *advantage* (Adv_{sukses}) dari algoritma ini bergantung pada 2 parameter yaitu waktu eksekusi (et) dan waktu query (qrt) untuk mendapatkan hasil query dan kondisi yang sesuai dengan Algoritma 6.1 baris ke-9. Fungsi ini diformulasikan sebagai $F_{adv}(et, qrt) = \max_{adv}\{Adv_{sukses}\}$. M-DCP dikatakan aman terhadap serangan *Adv* jika fungsi *advantage* memenuhi $F_{adv}(et, qrt) \leq \varepsilon$, dengan nilai $\varepsilon > 0$ dan bernilai sangat kecil. Oleh karena itu, Algoritma 6.1 menjadi tidak mungkin untuk dijalankan dikarenakan sifat hash yang digunakan pada M-DCP memiliki luaran yang tidak saling bentrok (collision) dan *irreversible*.

Kemampuan C-4 diartikan bahwa *Adv* tidak memiliki informasi mengenai parameter yang tersimpan pada node sensor akan tetapi mencoba untuk mentransmisikan modifikasi dari bagian pesan tertentu. Tujuan dari kemampuan ini adalah node sensor akan sibuk untuk memverifikasi pesan palsu. M-DCP mempunyai mekanisme verifikasi menggunakan tag sebagai pertahanan awal yang membutuhkan waktu sekitar 0.6 ms. Serangan yang dapat dijalankan sesuai dengan kemampuan C-4 yaitu serangan probabilitas dan DoS berbasis PKC. Detail dari pertahanan terhadap kedua serangan tersebut telah dibahas pada Bab 5.3.

Berdasarkan ruang kemungkinan kombinasi pada kemampuan C-2 dan informasi TinySet pada kemampuan C-3, *Adv* dapat berpura-pura menjadi pengguna yang valid sesuai dengan kemampuan C-5. *Adv* harus menemukan pasangan puzzle dan kunci sesi yang valid untuk dapat berhasil dengan mencoba sebanyak $D_{puzzle} \times D_{sesK}$ = 2^{32} x 2^{64} = 2^{96} iterasi. Jika setiap iterasi membutuhkan waktu 0.1 ms maka *Adv* membutuhkan waktu sebesar 9.169x10^{19} hari.

Pemanfaatan M-DCP yang dilengkapi dengan TinySet meningkatkan skalabilitas pada pengaturan pengguna khususnya pada JSN. Pengguna yang mengakses JSN dapat ditambah, dikurangi maupun dicari pada TinySet. Dengan peluang false positif yang sama, TinySet lebih hemat ruang penyimpanan hingga

77% dibandingkan CBF. Penambahan anggota pada skema TinySet memiliki kompleksitas lebih tinggi yang ditunjukkan dengan peningkatan waktu eksekusi. Akan tetapi proses ini dijalankan pada administrator dengan fakta bahwa sumber daya bukan merupakan keterbatasan. Sedangkan performansi pada sisi penerima atau node sensor yang menggunakan M-DCP meningkat hanya kurang dari 0.5 detik atau 36% dibandingkan menggunakan BAS. TinySet dengan regularisasi mempermudah administrator dalam pendefinisian TinySet di fase inisialisasi. Tujuan regularisasi ini utamanya untuk menjaga agar rata-rata jumlah anggota di setiap indeks pada bucket mendekati satu sehingga terdapat keseimbangan antara pemanfaataan ruang penyimpanan dengan waktu pemrosesan baik untuk penambahan atau pencarian anggota pada TinySet. Ruang penyimpanan pada TS64 lebih kecil dibandingkan dengan TS32 dengan perbedaaan waktu pemrosesan dibawah 1 detik. M-DCP mengeliminasi pemanfaatan MAC pada DCP dan menambahkan sebuah operasi hash sebagai gantinya. Hal ini diikuti dengan ukuran tag yang tetap yaitu 3 byte sehingga kompleksitas serangan meningkat yang dibuktikan dengan ROM.

Halaman ini sengaja dikosongkan

BAB 7
KESIMPULAN

7.1. Kesimpulan

Penggunaan skema puzzle dinamis dengan fungsi ambang terbaik yang dilengkapi dengan optimum TinySet diajukan sebagai metode utama pada JSN multiuser dalam menghadapi serangan DoS berbasis PKC. Skema ini bisa memverifikasi pengirim dalam jumlah besar dengan kapasitas ruang penyimpanan yang lebih rendah dibandingkan dengan Bloom filter. Selain itu, kerahasiaan pesan yang dikirim menjadi terjaga dengan adanya metode enkripsi. Beberapa poin utama kesimpulan dijelaskan secara lebih spesifik sebagai berikut :

a. Kompleksitas serangan meningkat khususnya spesifik pada serangan probabilitas dan DoS berbasis PKC. Hal ini dibuktikan dengan *adversary model* dan *Random Oracle Model*. Selain itu kompleksitas serangan dengan cara brute force terhadap M-DCP yang dilengkapi TinySet akan meningkat hingga $1.86x10^{137}$ trial, jika setiap trial membutuhkan waktu 0.1 ms maka akan dibutuhkan waktu selama 2.153×10^{128} hari untuk mencoba semua kemungkinan dengan peluang keberhasilan bisa melewati verifikasi TinySet sebesar 10^{-6}.
b. Fungsi ambang terbaik pada metode eksperimental adalah fungsi power pada kuartil 1 dan eksponensial pada kuartil 2. Metode ini disempurnakan dengan rumus probabilistik sehingga dapat ditentukan bahwa fungsi ambang eksponensial adalah fungsi ambang terbaik berdasarkan jumlah iterasi hash, standar deviasi dan peluang tidak ditemukannya solusi. Skema puzzle dinamis dengan fungsi ambang eksponensial bisa dimanfaatkan sesuai dengan kebutuhan akan data yang dikirim. Jika data yang dikirimkan bersifat rahasia maka DCP menjadi pilihan akan tetapi jika data yang dikirim tidak harus terenkripsi karena waktu pemrosesan menjadi utama maka DMSP bisa digunakan Peningkatan keamanan ini berakibat pada menurunnya performansi

yang ditunjukkan dengan peningkatan kompleksitas verifikasi pada JSN dan berdampak pada waktu pemrosesan yang lebih tinggi. Penggunaan DCP untuk multiuser (M-DCP) meningkatkan waktu verifikasi pada node sensor hanya kurang dari 0.5 detik atau 36% lebih tinggi dibandingkan BAS.

c. Metode DCP menjalankan proses enkripsi pada pesan, kunci sesi dan solusi puzzle menggunakan algoritma RC5. Adanya enkripsi meningkatkan waktu pemrosesan pesan di sisi pengirim karena proses ini dijalankan pada setiap langkah sebelum dijalankan iterasi hash. Akan tetapi peningkatan kompleksitas ini ditutupi dengan rendahnya jumlah iterasi hash yang dijalankan menggunakan fungsi ambang eksponensial. Jika setiap proses enkripsi membutuhkan waktu 1 ms maka maksimum waktu yang dibutuhkan untuk pemrosesan pesan sekitar 4 menit.

d. Pemanfaatan skema keanggotaan TinySet pada JSN menurunkan kebutuhan ruang penyimpanan dengan tetap menjaga waktu penambahan maupun verifikasi pengguna. Dengan peluang false positif yang sama, kebutuhan ruang penyimpanan TinySet lebih hemat dibandingkan dengan CBF. Selisih terkecil terjadi pada jumlah pengguna 200 dengan peluang keberhasilan 0.1 yaitu sebesar 77.95% dibandingkan dengan CBF. Hal ini diimbangi dengan peningkatan kompleksitas komputasi yang ditunjukkan dengan meningkatnya waktu yang dibutuhkan untuk penambahan pengguna untuk TinySet terutama pada indeks 64 bit. Peningkatan ini tentu tidak menjadi masalah, karena proses penambahan pengguna dijalankan oleh Administrator yang memiliki sumber daya komputasi tinggi.

7.2. Saran

Beberapa poin yang disarankan untuk pengembangan penelitian ini adalah:

a. Pemanfaatan skema M-DCP pada komunikasi antar pengguna dan JSN dengan topologi jaringan dari penerima yaitu node sensor yang disesuaikan agar mendekati kondisi riil yaitu multihop. Sehingga delay pada sisi penerima tidak hanya dipengaruhi oleh kompleksitas verifikasi pengguna melainkan

bagaimana mekanisme routing dan prioritas verifikasi pada kondisi seperti apa yang akan didahulukan.

b. Keamanan menjadi aspek yang penting pada JSN. Jika terdapat node sensor yang telah diambil alih oleh peretas maka perlu dibangun mekanisme yang tepat untuk melokalisir node tersebut sehingga tidak mengganggu komunikasi pada node sensor lainnya atau bahkan base station.

c. Pemanfaatan base station untuk mengatur lalu lintas komunikasi pada JSN memegang peranan penting. Skenario JSN jika base station diambil alih atau disusupi oleh peretas menjadi bahan penting untuk pengembangan skema yang akan diajukan berikutnya.

Halaman ini sengaja dikosongkan

DAFTAR PUSTAKA

Afianti, F., Wirawan & Suryani, T., 2018. Dynamic Cipher Puzzle for Efficient Broadcast Authentication in Wireless Sensor Networks. *Sensors*, 18(11), hal.4021.

Afianti, F., Wirawan & Suryani, T., 2019. Dynamic Message Puzzle as Pre-Authentication Scheme in Wireless Sensor Networks. *International Journal on Advanced Science, Engineering and Information Technology*, 9(1), hal.204–212.

Al-Riyami, A., Zhang, N. & Keane, J., 2016. An Adaptive Early Node Compromise Detection Scheme for Hierarchical WSNs. *IEEE Access*, 4, hal.4183–4206.

Alemdar, H. & Ersoy, C., 2010. Wireless sensor networks for healthcare: A survey. *Computer Networks*, 54(15), hal.2688–2710.

Anastasi, G., 2011. Data Collection in Wireless Sensor Networks with Mobile Elements : A Survey Data Collection in Wireless Sensor Networks with Mobile Elements : A Survey. *ACM Transactions on Sensor Networks (TOSN)*, 8(1), hal.7.

Aura, T., Nikander, P. & Leiwo, J., 2000. DOS-resistant authentication with client puzzles. In *In International workshop on security protocols*. Berlin, Heidelberg: Springer, hal. 170–177. Available at: http://www.springerlink.com/index/T4DMUWDG8V49LXCL.pdf%5Cnhttp://www.tcs.hut.fi/old/papers/aura/aura-nikander-leiwo-protocols00.pdf.

Ayday, E. & Fekri, F., 2012. A secure broadcasting scheme to provide availability , reliability and authentication for wireless sensor networks. *Ad Hoc Networks*, 10(7), hal.1278–1290. Available at: http://dx.doi.org/10.1016/j.adhoc.2012.03.010.

Biswas, K. et al., 2016. Performance Evaluation of Block Ciphers for Wireless Sensor Networks. In R. K. Choudhary et al., ed. *Advanced Computing and Communication Technologies: Proceedings of the 9th ICACCT, 2015*. Singapore: Springer Singapore, hal. 443–452. Available at: https://doi.org/10.1007/978-981-10-1023-1_44.

Brewster, T., 2015. Accidental DDoS? How China's Censorship Machine Can Cause Unintended Web Blackouts. Available at: https://www.forbes.com/sites/thomasbrewster/2015/01/26/china-great-firewall-causing-ddos-attacks/#6a5ce2566a47.

Broder, A. & Mitzenmacher, M., 2004. Network Applications of Bloom Filters : A Survey. *Internet mathematics*, 1(4), hal.485–509.

Cao, X. et al., 2008. IMBAS: Identity-based multi-user broadcast authentication

in wireless sensor networks. *Computer communications*, 31(4), hal.659–667.

Chen, Y.-S. et al., 2008. Broadcast authentication in sensor networks using compressed Bloom filters. *Distributed Computing in Sensor Systems*, 5067, hal.99–111.

Chen, Y., Chen, C. & Ma, J., 2009. Query-Based Data Collection in Wireless Sensor Networks with Mobile Sinks ∗ . In *Proceedings of the 2009 international conference on wireless communications and mobile computing: connecting the World wirelessly*. ACM, hal. 1157–1162.

Cheng, C., Lin, I. & Huang, S., 2015. An RSA-Like Scheme for Multiuser Broadcast Authentication in Wireless Sensor Networks. *International Journal of Distributed Sensor Networks*, 2015.

Chuchaisri, P. & Newman, R., 2012. Fast response PKC-based broadcast authentication in wireless sensor networks. *Mobile Networks and Applications*, 17(4), hal.508–525.

Cormen, T.H. et al., 2009. *Introduction to Algorithms* Third., MIT Press.

Dargie, W. & Poellabauer, C., 2010. *Fundamentals of Wireless Sensor Networks*, John Wiley & Sons. Available at: http://doi.wiley.com/10.1002/9780470666388.

Das, A.K., 2016. A secure and robust temporal credential-based three-factor user authentication scheme for wireless sensor networks. *Peer-to-Peer Networking and Applications*, 9(1), hal.223–244.

Das, A.K., 2009. An Unconditionally Secure Key Management Scheme for Large-Scale Heterogeneous Wireless Sensor Networks. In *In Communication Systems and Networks and Workshops, 2009. COMSNETS 2009. First International*. Bangalore, India: IEEE, hal. 1–10.

Dong, Q., Liu, D. & Ning, P., 2013. Providing DoS resistance for signature-based broadcast authentication in sensor networks. *ACM Transactions on Embedded Computing Systems*, 12(3), hal.1–26. Available at: http://dl.acm.org/citation.cfm?doid=2442116.2442123.

Du, X. & Chen, H., 2008. Defending DoS Attacks on Broadcast Authentication in Wireless Sensor Networks. In *2008 IEEE International Conference on Communications*. Beijing: IEEE, hal. 1653–1657.

Eastlake, D. & Jones, P., 2001. *RFC 3174-US Secure Hash Algorithm 1 (SHA1)(2001)*,

Einziger, G. & Friedman, R., 2017. TinySet - An access efficient self adjusting bloom filter construction. *IEEE/ACM Transactions on Networking*, 25(4), hal.2295–2307.

Fadel, E. et al., 2015. A survey on wireless sensor networks for smart grid. *Computer communications*, 71, hal.22–33.

Gan, X. & Li, Q., 2009. A Multi-user DoS-containment broadcast authentication scheme for wireless sensor networks. In *Proceedings - 2009 International Conference on Information Technology and Computer Science, ITCS 2009*. hal. 472–475. Available at: http://www.scopus.com/inward/record.url?eid=2-s2.0-71049167923&partnerID=40&md5=33d7aa935762e42859475b800cbc1af0.

Grover, K. & Lim, A., 2015. A survey of broadcast authentication schemes for wireless networks. *Ad Hoc Networks*, 24(PA), hal.288–316. Available at: http://dx.doi.org/10.1016/j.adhoc.2014.06.008.

Groza, B., 2008. Broadcast authentication with practically unbounded one-way chains. *Journal of Software*, 3(3), hal.11–20.

Groza, B. & Warinschi, B., 2014. Cryptographic puzzles and DoS resilience, revisited. *Designs, Codes, and Cryptography*, 73(1), hal.177–207.

Gumelar, B., 2013. *Analisis performansi algoritma knapsack untuk optimalisasi pemilihan proyek di PT.Gits Indonesia*. Universitas Komputer Indonesia. Available at: https://elib.unikom.ac.id/gdl.php?mod=browse&op=read&id=jbptunikompp-gdl-bagjagumel-31178.

Hamida, E. Ben & Chelius, G., 2008. Strategies for data dissemination to mobile sinks in wireless sensor networks. *IEEE Wireless Communications*, 15(6), hal.31–37.

Han, L. et al., 2018. An efficient and secure three-factor based authenticated key exchange scheme using elliptic curve cryptosystems. *Peer-to-Peer Networking and Applications*, 11(1), hal.63–73.

He, D. et al., 2012. DiCode: DoS-resistant and distributed code dissemination in wireless sensor networks. *IEEE Transactions on Wireless Communications*, 11(5), hal.1946–1956.

He, D. et al., 2013. Secure Data Discovery and Dissemination based on Hash Tree for Wireless Sensor Networks. *IEEE Transactions on Wireless Communications*, 12(9), hal.4638–4646.

Hogg, J., 2005. *Web service security: Scenarios, patterns, and implementation guidance for Web Services Enhancements (WSE) 3.0.*, O'Reilly Media, Inc.

Hui, J.W. & Culler, D., 2004. The dynamic behavior of a data dissemination protocol for network programming at scale. *Proceedings of the 2nd international conference on Embedded networked sensor systems - SenSys '04*, (January), hal.81. Available at: http://portal.acm.org/citation.cfm?doid=1031495.1031506.

Hyun, S. & Ning, P., 2008. Seluge: Secure and dos-resistant code dissemination in wireless sensor networks. In *In Information Processing in Sensor Networks, 2008. IPSN'08.* hal. 445–456.

ITU-T (CCITT) Recommendation, 1991. REC, I. T. U. T. X. 800 security architecture for open systems interconnection for ccitt applications. , hal.1–48.

Jianjun, Y. & Changjun, J., 2010. A Biometric-Based User Authentication for Wireless Sensor Networks. *Wuhan University Journal of Natural Sciences*, 15(3), hal.272–276.

Johnson, D., Menezes, A. & Vanstone, S., 2001. The Elliptic Curve Digital Signature Algorithm (ECDSA). *International Journal of Information Security*, 1(1), hal.36–63.

Kanizo, Y., Hay, D. & Keslassy, I., 2013. Access-efficient Balanced Bloom Filters. *Computer Communications*, 36(4), hal.373–385. Available at: http://dx.doi.org/10.1016/j.comcom.2012.11.007.

Katz, J. & Lindell, Y., 2008. *Introduction to Modern Cryptography*, CRC press.

Kim, D. & An, S., 2016. PKC-based DoS Attacks-Resistant Scheme in Wireless Sensor Networks. *IEEE Sensors Journal*, 16(8), hal.2217–2218.

Kim, D., Kim, D. & An, S., 2016. Source Authentication for Code Dissemination Supporting Dynamic Packet Size in Wireless Sensor Networks. *Sensors*, 16(7), hal.1063.

Kovacs, E., 2014. DDoS Attacks Cost $40,000 Per Hour: Incapsula. Available at: https://www.securityweek.com/ddos-attacks-cost-40000-hour-incapsula.

Krawczyk, H., Bellare, M. & Canetti, R., 1997. *HMAC: Keyed-hashing for message authentication, RFC 2104*, Available at: http://www.rfc-editor.org/rfc/pdfrfc/rfc2104.txt.pdf.

Kumar, A. et al., 2012. A dynamic password-based user authentication scheme for hierarchical wireless sensor networks. *Journal of Network and Computer Applications*, 35(5), hal.1646–1656. Available at: http://dx.doi.org/10.1016/j.jnca.2012.03.011.

Kwon, T. & Hong, J., 2010. Secure and efficient broadcast authentication in wireless sensor networks. *IEEE Transactions on Computers*, 59(8), hal.1120–1133.

Lamport, L., 1981. Password authentication with insecure communication. *Communications of the ACM*, 24(11), hal.770–772.

Li, S., Cui, J. & Li, Z., 2011. Wireless Sensor Network for Precise Agriculture Monitoring. In *2011 Fourth International Conference on Intelligent Computation Technology and Automation*. hal. 307–310.

Li, S., Yılmaz, Y. & Wang, X., 2015. Quickest Detection of False Data Injection Attack in Wide-Area Smart Grids. *IEEE Transactions on Smart Grid*, 6(6), hal.2725–2735.

Lim, C.H., 2011. Secure code dissemination and remote image management using

short-lived signatures in WSNs. *IEEE Communications Letters*, 15(4), hal.362–364.

Liu, A. & Ning, P., 2008. TinyECC : A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks. In *Proceedings of the 7th international conference on Information processing in sensor networks*. IEEE Computer Society, hal. 245–256.

Liu, D. et al., 2005. Practical broadcast authentication in sensor networks. *MobiQuitous 2005: Second Annual International Conference on Mobile and Ubiquitous Systems -Networking and Services*, hal.118–129.

Liu, D. & Ning, P., 2004. Multilevel μ TESLA : Broadcast Authentication for Distributed Sensor Networks. *ACM Transactions on Embedded Computing Systems*, 3(4), hal.800–836.

Liu, Y., Li, J. & Guizani, M., 2012. PKC based broadcast authentication using signature amortization for WSNs. *IEEE Transactions on Wireless Communications*, 11(6), hal.2106–2115.

Lu, H., Li, J. & Guizani, M., 2014. Secure and Efficient Data Transmission for Cluster-Based Wireless Sensor Networks. *IEEE transactions on parallel and distributed systems*, 25(3), hal.750–761.

Manjeshwar, A. & Agrawal, D.P., 2001. TEEN : A Routing Protocol for Enhanced Efficiency in Wireless Sensor Networks. In *Proceedings 15th International Parallel and Distributed Processing Symposium. IPDPS 2001*. San Francisco, CA, USA, USA: IEEE, hal. 2009–2015.

Maris, A., Kundu, A. & Yoon, A., 2019. *Common Vulnerability Scoring System version 3.1: Specification Document*, Available at: https://www.first.org/cvss/specification-document.

Menezes, A.J., Oorschot, P.C. Van & Vanstone, S.A., 1996. *Handbook of Applied Cryptography*, CRC press.

Merkle, J. & Lochter, M., 2010. Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation, RFC 5639. , hal.1–27. Available at: https://tools.ietf.org/html/rfc5639.

Meulenaer, G. De et al., 2008. On the Energy Cost of Communication and Cryptography in Wireless Sensor Networks. In *WIMOB'08 IEEE International Conference on Wireless and Mobile Computing*. IEEE, hal. 580–585.

Moore, D.S., McCabe, G.P. & Craig, B.A., 2009. *Introduction to the Practice of Statistics*, Available at: http://books.google.com/books?id=20xsSwAACAAJ&pgis=1.

Ning, P. & Liu, A.N., 2008. Mitigating DoS Attacks against Broadcast Authentication in Wireless Sensor Networks. *ACM Transactions on Sensor Networks (TOSN)*, 4(1), hal.1–35.

Ojha, T., Misra, S. & Raghuwanshi, N.S., 2015. Wireless sensor networks for agriculture: The state-of-the-art in practice and future challenges. *Computers and Electronics in Agriculture*, 118, hal.66–84.

Oliveira, L.B. et al., 2009. Secure-TWS : Authenticating Node to Multi-user Communication in Shared Sensor Networks. *The Computer Journal*, 55(4), hal.384–396.

Pei, C. et al., 2018. Trade-off of security and performance of lightweight block ciphers in Industrial Wireless Sensor Networks. *EURASIP Journal on Wireless Communications and Networking*, 2018(1), hal.117.

Perrig, A. et al., 2002. SPINS: Security protocols for sensor networks. *Wireless Networks*, 8(5), hal.521–534.

Perrig, A., 2001. The BiBa One-Time Signature and Broadcast Authentication Protocol. In *In Proceedings of the 8th ACM conference on Computer and Communications Security*. ACM, hal. 28–37.

Ping, R. & Csiro, L., 2015. A Sybil Attack Detection Scheme for a Centralized Clustering-based Hierarchical Network. *Trustcom/BigDataSE/ISPA*, 1(August), hal.318–325.

Prasuna, P. & Hemalatha, R., 2013. Comparative Study of Multi User Broadcast Authentication in Wireless Sensor Network. *International Journal of Computer Science and Informatics*, 3(2), hal.108–110.

Putze, F., Sanders, P. & Singler, J., 2007. Cache- , Hash- and Space-Efficient Bloom Filters. In *International Workshop on Experimental and Efficient Algorithms*. Springer, Berlin, Heidelberg, hal. 108–121.

Qiao, Y., Li, T. & Chen, S., 2011. One Memory Access Bloom Filters and Their Generalization. In *2011 Proceedings IEEE INFOCOM*. IEEE, hal. 1745–1753.

Ravipati, S., 2017. University Hackers Attacked 5,000 IoT Devices on Campus. Available at: https://campustechnology.com/articles/2017/02/13/university-hackers-attacked-5000-iot-devices-on-campus.aspx.

Ren, K. et al., 2009. Multi-user broadcast authentication in wireless sensor networks. *IEEE Transactions on Vehicular Technology*, 58(8), hal.4554–4564.

Ren, K. et al., 2007. On Broadcast Authentication in Wireless Sensor Networks. *IEEE Transactions on Wireless Communications*, 6(11), hal.4136–4144.

Reyzin, L. & Reyzin, N., 2002. Better than BiBa : Short One-time Signatures with Fast Signing and Verifying. In *In Australasian Conference on Information Security and Privacy*. Springer, Berlin, Heidelberg, hal. 144–153.

Sethi, M., Arkko, J. & Keranen, A., 2012. End-to-end Security for Sleepy Smart Object Networks. In *IEEE 37th Conference onLocal Computer Networks*

Workshops (LCN Workshops). hal. 964–972.

Shim, K., 2017. BASIS : A Practical Multi-User Broadcast Authentication Scheme in Wireless Sensor Networks. *IEEE Transactions on Information Forensics and Security*, 12(7), hal.1545–1554.

Shim, K.A., Lee, Y.R. & Park, C.M., 2013. EIBAS: An efficient identity-based broadcast authentication scheme in wireless sensor networks. *Ad Hoc Networks*, 11(1), hal.182–189. Available at: http://dx.doi.org/10.1016/j.adhoc.2012.04.015.

Shiuhpyng, S.M.C. et al., 2006. An Efficient Broadcast Authentication Scheme in Wireless Sensor Networks. In *In Proceedings of the 2006 ACM Symposium on Information, computer and communications security*. ACM, hal. 311–320.

Stevens, M. et al., 2017. The First Collision for Full SHA-1. In *Annual International Cryptology Conference*. Springer, Cham, hal. 570–596.

Tan, H. et al., 2013. A confidential and DoS-resistant multi-hop code dissemination protocol for wireless sensor networks. *Computers and Security*, 32, hal.36–55. Available at: http://dx.doi.org/10.1016/j.cose.2012.09.012.

Tarkoma, S., Rothenberg, C.E. & Lagerspetz, E., 2012. Theory and practice of bloom filters for distributed systems. *IEEE Communications Surveys and Tutorials*, 14(1), hal.131–155.

Turner, L., 2013. Anonymous hackers jailed for DDoS attacks on Visa, Mastercard and Paypal. Available at: https://www.independent.co.uk/news/uk/crime/anonymous-hackers-jailed-for-ddos-attacks-on-visa-mastercard-and-paypal-8465791.html.

Wang, D., Li, W. & Wang, P., 2018. Measuring Two-Factor Authentication Schemes for Real-Time Data Access in Industrial Wireless Sensor Networks. *IEEE Transactions on Industrial Informatics*, 14(9), hal.4081–4092.

Wong, K.H.M., Cao, J. & Wang, S., 2006. A Dynamic User Authentication Scheme for Wireless Sensor Networks. In *Sensor Networks, Ubiquitous, and Trustworthy Computing, 2006. IEEE International Conference on.* Taichung, Taiwan: IEEE Computer Society.

Woolf, N., 2016. DDoS attack that disrupted internet was largest of its kind in history, experts say. *The Guardian*. Available at: https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet.

Xu, A. et al., 2016. Improving Efficiency of Authenticated OpenFlow Handshake using Coprocessors. In *IEEE 8th International Conference on Information Technology in Medicine and Education (ITME)*. IEEE, hal. 576–580.

Xu, G. et al., 2018. A multi-server two-factor authentication scheme with un-traceability using elliptic curve cryptography. *Sensors*, 18(7), hal.1–19.

Yang, S., 2013. *Wireless Sensor Networks Principles, Design and Applications*, Springer.

Yang, S. & Cao, Y., 2008. Networked control systems and wireless sensor networks : theories and applications. *International Journal System Science*, 39(11), hal.1041–1044.

Ye, F. et al., 2002. A Two-Tier Data Dissemination Model for Large-scale Wireless Sensor Networks. In *In Proceedings of the 8th annual international conference on Mobile computing and networking*. ACM, hal. 148–159.

Zheng, Z. et al., 2016. Energy and Memory Efficient Clone Detection in Wireless Sensor Networks. *IEEE Transactions on Mobile Computing*, 15(5), hal.1130–1143.

Zhou, Y., Zhang, Y. & Fang, Y., 2007. Access control in wireless sensor networks. *Ad Hoc Networks*, 5(1)(July 2006), hal.3–13.

LAMPIRAN

A. Tabel jumlah iterasi hash, peluang tidak ditemukannya solusi dan standar deviasi pada kekuatan puzzle = 1
B. Tabel jumlah iterasi hash, peluang tidak ditemukannya solusi dan standar deviasi pada kekuatan puzzle = 2
C. Tabel jumlah iterasi hash, peluang tidak ditemukannya solusi dan standar deviasi pada kekuatan puzzle = 3
D. Algoritma ECDSA
E. Algoritma RC5
F. Algoritma SHA1
G. Kode program simulasi skema puzzle dinamis pada NS3
H. Kode program test bed skema puzzle dinamis pada Arduino

Halaman ini sengaja dikosongkan

Lampiran A. Tabel jumlah iterasi hash, peluang tidak ditemukannya solusi dan standar deviasi pada kekuatan puzzle = 1

Trial ke-	$P_{[success,L=1]}$	*nDCP*	*Pzs*	σ_l
1	0.1	1	0.9	0.300
2	0.2	1	0.8	0.400
3	0.3	1	0.7	0.458
4	0.4	1	0.6	0.490
5	0.5	1	0.5	0.500
6	0.6	2	0.4	0.490
7	0.7	2	0.3	0.458
8	0.8	3	0.2	0.400
9	0.9	4	0.1	0.300

Halaman ini sengaja dikosongkan

Lampiran B. Tabel jumlah iterasi hash, peluang tidak ditemukannya solusi dan standar deviasi pada kekuatan puzzle = 2

Trial ke-	$P_{[success,L=1]}$	$P_{[success,L=2]}$	*nDCP*	*Pzs*	σ_l
1	0.1	0.1	2	0.81	0.581
2	0.1	0.2	2	0.72	0.695
3	0.1	0.3	3	0.63	0.734
4	0.1	0.4	3	0.54	0.722
5	0.1	0.5	4	0.45	0.672
6	0.1	0.6	5	0.36	0.591
7	0.1	0.7	6	0.27	0.483
8	0.1	0.8	7	0.18	0.351
9	0.1	0.9	10	0.09	0.198
10	0.2	0.1	2	0.72	0.576
11	0.2	0.2	2	0.64	0.668
12	0.2	0.3	3	0.56	0.697
13	0.2	0.4	3	0.48	0.684
14	0.2	0.5	4	0.4	0.637
15	0.2	0.6	5	0.32	0.563
16	0.2	0.7	6	0.24	0.466
17	0.2	0.8	7	0.16	0.347
18	0.2	0.9	10	0.08	0.206
19	0.3	0.1	2	0.63	0.557
20	0.3	0.2	2	0.56	0.633
21	0.3	0.3	3	0.49	0.658
22	0.3	0.4	3	0.42	0.645
23	0.3	0.5	4	0.35	0.604
24	0.3	0.6	5	0.28	0.538
25	0.3	0.7	6	0.21	0.451
26	0.3	0.8	7	0.14	0.344
27	0.3	0.9	10	0.07	0.216
28	0.4	0.1	2	0.54	0.526
29	0.4	0.2	2	0.48	0.594
30	0.4	0.3	3	0.42	0.617
31	0.4	0.4	3	0.36	0.607
32	0.4	0.5	4	0.3	0.573
33	0.4	0.6	5	0.24	0.516
34	0.4	0.7	6	0.18	0.440
35	0.4	0.8	7	0.12	0.345
36	0.4	0.9	10	0.06	0.226

Trial ke-	$P_{[success,L=1]}$	$P_{[success,L=2]}$	*nDCP*	*Pzs*	σ_I
37	0.5	0.1	2	0.45	0.487
38	0.5	0.2	2	0.4	0.551
39	0.5	0.3	3	0.35	0.576
40	0.5	0.4	3	0.3	0.572
41	0.5	0.5	4	0.25	0.545
42	0.5	0.6	5	0.2	0.498
43	0.5	0.7	6	0.15	0.433
44	0.5	0.8	7	0.1	0.348
45	0.5	0.9	10	0.05	0.237
46	0.6	0.1	3	0.36	0.442
47	0.6	0.2	3	0.32	0.508
48	0.6	0.3	4	0.28	0.537
49	0.6	0.4	4	0.24	0.540
50	0.6	0.5	5	0.2	0.522
51	0.6	0.6	6	0.16	0.484
52	0.6	0.7	7	0.12	0.429
53	0.6	0.8	8	0.08	0.353
54	0.6	0.9	11	0.04	0.249
55	0.7	0.1	3	0.27	0.394
56	0.7	0.2	3	0.24	0.466
57	0.7	0.3	4	0.21	0.502
58	0.7	0.4	4	0.18	0.514
59	0.7	0.5	5	0.15	0.504
60	0.7	0.6	6	0.12	0.476
61	0.7	0.7	7	0.09	0.430
62	0.7	0.8	8	0.06	0.362
63	0.7	0.9	11	0.03	0.261
64	0.8	0.1	4	0.18	0.348
65	0.8	0.2	4	0.16	0.431
66	0.8	0.3	5	0.14	0.475
67	0.8	0.4	5	0.12	0.495
68	0.8	0.5	6	0.1	0.494
69	0.8	0.6	7	0.08	0.474
70	0.8	0.7	8	0.06	0.435
71	0.8	0.8	9	0.04	0.372
72	0.8	0.9	12	0.02	0.274
73	0.9	0.1	5	0.09	0.313
74	0.9	0.2	5	0.08	0.407
75	0.9	0.3	6	0.07	0.460
76	0.9	0.4	6	0.06	0.486

Trial ke-	$P_{[success,L=1]}$	$P_{[success,L=2]}$	*nDCP*	*Pzs*	σ_I
77	0.9	0.5	7	0.05	0.492
78	0.9	0.6	8	0.04	0.479
79	0.9	0.7	9	0.03	0.444
80	0.9	0.8	10	0.02	0.385
81	0.9	0.9	13	0.01	0.287

Halaman ini sengaja dikosongkan

Lampiran C. Tabel jumlah iterasi hash, peluang tidak ditemukannya solusi dan standar deviasi pada kekuatan puzzle = 3

Trial ke-	$P_{[success,L=1]}$	$P_{[success,L=2]}$	$P_{[success,L=3]}$	*nDCP*	*Pzs*	σ_I
1	0.1	0.1	0.1	3	0.729	0.893
2	0.1	0.1	0.2	4	0.648	1.025
3	0.1	0.1	0.3	5	0.567	1.066
4	0.1	0.1	0.4	6	0.486	1.043
5	0.1	0.1	0.5	8	0.405	0.971
6	0.1	0.1	0.6	9	0.324	0.858
7	0.1	0.1	0.7	12	0.243	0.708
8	0.1	0.1	0.8	15	0.162	0.527
9	0.1	0.1	0.9	20	0.081	0.314
10	0.1	0.2	0.1	3	0.648	0.899
11	0.1	0.2	0.2	4	0.576	0.989
12	0.1	0.2	0.3	5	0.504	1.009
13	0.1	0.2	0.4	6	0.432	0.978
14	0.1	0.2	0.5	8	0.36	0.907
15	0.1	0.2	0.6	9	0.288	0.801
16	0.1	0.2	0.7	12	0.216	0.664
17	0.1	0.2	0.8	15	0.144	0.499
18	0.1	0.2	0.9	20	0.072	0.305
19	0.1	0.3	0.1	4	0.567	0.875
20	0.1	0.3	0.2	5	0.504	0.935
21	0.1	0.3	0.3	6	0.441	0.943
22	0.1	0.3	0.4	7	0.378	0.909
23	0.1	0.3	0.5	9	0.315	0.841
24	0.1	0.3	0.6	10	0.252	0.744
25	0.1	0.3	0.7	13	0.189	0.621
26	0.1	0.3	0.8	16	0.126	0.473
27	0.1	0.3	0.9	21	0.063	0.297
28	0.1	0.4	0.1	4	0.486	0.825
29	0.1	0.4	0.2	5	0.432	0.867
30	0.1	0.4	0.3	6	0.378	0.868
31	0.1	0.4	0.4	7	0.324	0.835
32	0.1	0.4	0.5	9	0.27	0.774
33	0.1	0.4	0.6	10	0.216	0.689
34	0.1	0.4	0.7	13	0.162	0.581
35	0.1	0.4	0.8	16	0.108	0.450
36	0.1	0.4	0.9	21	0.054	0.292
37	0.1	0.5	0.1	5	0.405	0.753

38	0.1	0.5	0.2	6	0.36	0.787
39	0.1	0.5	0.3	7	0.315	0.787
40	0.1	0.5	0.4	8	0.27	0.760
41	0.1	0.5	0.5	10	0.225	0.708
42	0.1	0.5	0.6	11	0.18	0.636
43	0.1	0.5	0.7	14	0.135	0.544
44	0.1	0.5	0.8	17	0.09	0.430
45	0.1	0.5	0.9	22	0.045	0.288
46	0.1	0.6	0.1	6	0.324	0.664
47	0.1	0.6	0.2	7	0.288	0.698
48	0.1	0.6	0.3	8	0.252	0.703
49	0.1	0.6	0.4	9	0.216	0.685
50	0.1	0.6	0.5	11	0.18	0.645
51	0.1	0.6	0.6	12	0.144	0.588
52	0.1	0.6	0.7	15	0.108	0.511
53	0.1	0.6	0.8	18	0.072	0.414
54	0.1	0.6	0.9	23	0.036	0.286
55	0.1	0.7	0.1	7	0.243	0.563
56	0.1	0.7	0.2	8	0.216	0.605
57	0.1	0.7	0.3	9	0.189	0.620
58	0.1	0.7	0.4	10	0.162	0.614
59	0.1	0.7	0.5	12	0.135	0.589
60	0.1	0.7	0.6	13	0.108	0.546
61	0.1	0.7	0.7	16	0.081	0.485
62	0.1	0.7	0.8	19	0.054	0.402
63	0.1	0.7	0.9	24	0.027	0.287
64	0.1	0.8	0.1	8	0.162	0.455
65	0.1	0.8	0.2	9	0.144	0.513
66	0.1	0.8	0.3	10	0.126	0.544
67	0.1	0.8	0.4	11	0.108	0.552
68	0.1	0.8	0.5	13	0.09	0.542
69	0.1	0.8	0.6	14	0.072	0.514
70	0.1	0.8	0.7	17	0.054	0.466
71	0.1	0.8	0.8	20	0.036	0.396
72	0.1	0.8	0.9	25	0.018	0.289
73	0.1	0.9	0.1	11	0.081	0.356
74	0.1	0.9	0.2	12	0.072	0.438
75	0.1	0.9	0.3	13	0.063	0.485
76	0.1	0.9	0.4	14	0.054	0.508
77	0.1	0.9	0.5	16	0.045	0.511
78	0.1	0.9	0.6	17	0.036	0.494

79	0.1	0.9	0.7	20	0.027	0.457
80	0.1	0.9	0.8	23	0.018	0.395
81	0.1	0.9	0.9	28	0.009	0.293
82	0.2	0.1	0.1	3	0.648	0.862
83	0.2	0.1	0.2	4	0.576	0.988
84	0.2	0.1	0.3	5	0.504	1.030
85	0.2	0.1	0.4	6	0.432	1.014
86	0.2	0.1	0.5	8	0.36	0.952
87	0.2	0.1	0.6	9	0.288	0.851
88	0.2	0.1	0.7	12	0.216	0.717
89	0.2	0.1	0.8	15	0.144	0.550
90	0.2	0.1	0.9	20	0.072	0.349
91	0.2	0.2	0.1	3	0.576	0.863
92	0.2	0.2	0.2	4	0.512	0.953
93	0.2	0.2	0.3	5	0.448	0.977
94	0.2	0.2	0.4	6	0.384	0.954
95	0.2	0.2	0.5	8	0.32	0.893
96	0.2	0.2	0.6	9	0.256	0.799
97	0.2	0.2	0.7	12	0.192	0.676
98	0.2	0.2	0.8	15	0.128	0.524
99	0.2	0.2	0.9	20	0.064	0.338
100	0.2	0.3	0.1	4	0.504	0.838
101	0.2	0.3	0.2	5	0.448	0.903
102	0.2	0.3	0.3	6	0.392	0.916
103	0.2	0.3	0.4	7	0.336	0.891
104	0.2	0.3	0.5	9	0.28	0.833
105	0.2	0.3	0.6	10	0.224	0.747
106	0.2	0.3	0.7	13	0.168	0.636
107	0.2	0.3	0.8	16	0.112	0.499
108	0.2	0.3	0.9	21	0.056	0.329
109	0.2	0.4	0.1	4	0.432	0.791
110	0.2	0.4	0.2	5	0.384	0.840
111	0.2	0.4	0.3	6	0.336	0.848
112	0.2	0.4	0.4	7	0.288	0.824
113	0.2	0.4	0.5	9	0.24	0.772
114	0.2	0.4	0.6	10	0.192	0.697
115	0.2	0.4	0.7	13	0.144	0.598
116	0.2	0.4	0.8	16	0.096	0.475
117	0.2	0.4	0.9	21	0.048	0.320
118	0.2	0.5	0.1	5	0.36	0.726
119	0.2	0.5	0.2	6	0.32	0.767

120	0.2	0.5	0.3	7	0.28	0.775
121	0.2	0.5	0.4	8	0.24	0.755
122	0.2	0.5	0.5	10	0.2	0.712
123	0.2	0.5	0.6	11	0.16	0.648
124	0.2	0.5	0.7	14	0.12	0.563
125	0.2	0.5	0.8	17	0.08	0.454
126	0.2	0.5	0.9	22	0.04	0.313
127	0.2	0.6	0.1	6	0.288	0.645
128	0.2	0.6	0.2	7	0.256	0.687
129	0.2	0.6	0.3	8	0.224	0.698
130	0.2	0.6	0.4	9	0.192	0.687
131	0.2	0.6	0.5	11	0.16	0.654
132	0.2	0.6	0.6	12	0.128	0.602
133	0.2	0.6	0.7	15	0.096	0.530
134	0.2	0.6	0.8	18	0.064	0.436
135	0.2	0.6	0.9	23	0.032	0.307
136	0.2	0.7	0.1	7	0.216	0.553
137	0.2	0.7	0.2	8	0.192	0.602
138	0.2	0.7	0.3	9	0.168	0.622
139	0.2	0.7	0.4	10	0.144	0.621
140	0.2	0.7	0.5	12	0.12	0.600
141	0.2	0.7	0.6	13	0.096	0.561
142	0.2	0.7	0.7	16	0.072	0.503
143	0.2	0.7	0.8	19	0.048	0.421
144	0.2	0.7	0.9	24	0.024	0.303
145	0.2	0.8	0.1	8	0.144	0.456
146	0.2	0.8	0.2	9	0.128	0.518
147	0.2	0.8	0.3	10	0.112	0.551
148	0.2	0.8	0.4	11	0.096	0.562
149	0.2	0.8	0.5	13	0.08	0.554
150	0.2	0.8	0.6	14	0.064	0.527
151	0.2	0.8	0.7	17	0.048	0.481
152	0.2	0.8	0.8	20	0.032	0.410
153	0.2	0.8	0.9	25	0.016	0.300
154	0.2	0.9	0.1	11	0.072	0.363
155	0.2	0.9	0.2	12	0.064	0.445
156	0.2	0.9	0.3	13	0.056	0.493
157	0.2	0.9	0.4	14	0.048	0.516
158	0.2	0.9	0.5	16	0.04	0.519
159	0.2	0.9	0.6	17	0.032	0.503
160	0.2	0.9	0.7	20	0.024	0.465

161	0.2	0.9	0.8	23	0.016	0.403
162	0.2	0.9	0.9	28	0.008	0.299
163	0.3	0.1	0.1	3	0.567	0.827
164	0.3	0.1	0.2	4	0.504	0.950
165	0.3	0.1	0.3	5	0.441	0.996
166	0.3	0.1	0.4	6	0.378	0.988
167	0.3	0.1	0.5	8	0.315	0.936
168	0.3	0.1	0.6	9	0.252	0.848
169	0.3	0.1	0.7	12	0.189	0.728
170	0.3	0.1	0.8	15	0.126	0.575
171	0.3	0.1	0.9	20	0.063	0.382
172	0.3	0.2	0.1	3	0.504	0.826
173	0.3	0.2	0.2	4	0.448	0.917
174	0.3	0.2	0.3	5	0.392	0.948
175	0.3	0.2	0.4	6	0.336	0.934
176	0.3	0.2	0.5	8	0.28	0.883
177	0.3	0.2	0.6	9	0.224	0.801
178	0.3	0.2	0.7	12	0.168	0.690
179	0.3	0.2	0.8	15	0.112	0.549
180	0.3	0.2	0.9	20	0.056	0.369
181	0.3	0.3	0.1	4	0.441	0.802
182	0.3	0.3	0.2	5	0.392	0.872
183	0.3	0.3	0.3	6	0.343	0.893
184	0.3	0.3	0.4	7	0.294	0.876
185	0.3	0.3	0.5	9	0.245	0.829
186	0.3	0.3	0.6	10	0.196	0.754
187	0.3	0.3	0.7	13	0.147	0.653
188	0.3	0.3	0.8	16	0.098	0.524
189	0.3	0.3	0.9	21	0.049	0.358
190	0.3	0.4	0.1	4	0.378	0.759
191	0.3	0.4	0.2	5	0.336	0.815
192	0.3	0.4	0.3	6	0.294	0.831
193	0.3	0.4	0.4	7	0.252	0.816
194	0.3	0.4	0.5	9	0.21	0.773
195	0.3	0.4	0.6	10	0.168	0.707
196	0.3	0.4	0.7	13	0.126	0.617
197	0.3	0.4	0.8	16	0.084	0.501
198	0.3	0.4	0.9	21	0.042	0.347
199	0.3	0.5	0.1	5	0.315	0.701
200	0.3	0.5	0.2	6	0.28	0.750
201	0.3	0.5	0.3	7	0.245	0.765

202	0.3	0.5	0.4	8	0.21	0.753
203	0.3	0.5	0.5	10	0.175	0.718
204	0.3	0.5	0.6	11	0.14	0.661
205	0.3	0.5	0.7	14	0.105	0.582
206	0.3	0.5	0.8	17	0.07	0.478
207	0.3	0.5	0.9	22	0.035	0.337
208	0.3	0.6	0.1	6	0.252	0.629
209	0.3	0.6	0.2	7	0.224	0.678
210	0.3	0.6	0.3	8	0.196	0.696
211	0.3	0.6	0.4	9	0.168	0.691
212	0.3	0.6	0.5	11	0.14	0.664
213	0.3	0.6	0.6	12	0.112	0.618
214	0.3	0.6	0.7	15	0.084	0.550
215	0.3	0.6	0.8	18	0.056	0.458
216	0.3	0.6	0.9	23	0.028	0.327
217	0.3	0.7	0.1	7	0.189	0.547
218	0.3	0.7	0.2	8	0.168	0.601
219	0.3	0.7	0.3	9	0.147	0.627
220	0.3	0.7	0.4	10	0.126	0.630
221	0.3	0.7	0.5	12	0.105	0.613
222	0.3	0.7	0.6	13	0.084	0.577
223	0.3	0.7	0.7	16	0.063	0.521
224	0.3	0.7	0.8	19	0.042	0.439
225	0.3	0.7	0.9	24	0.021	0.319
226	0.3	0.8	0.1	8	0.126	0.458
227	0.3	0.8	0.2	9	0.112	0.524
228	0.3	0.8	0.3	10	0.098	0.560
229	0.3	0.8	0.4	11	0.084	0.573
230	0.3	0.8	0.5	13	0.07	0.567
231	0.3	0.8	0.6	14	0.056	0.541
232	0.3	0.8	0.7	17	0.042	0.495
233	0.3	0.8	0.8	20	0.028	0.423
234	0.3	0.8	0.9	25	0.014	0.312
235	0.3	0.9	0.1	11	0.063	0.371
236	0.3	0.9	0.2	12	0.056	0.453
237	0.3	0.9	0.3	13	0.049	0.501
238	0.3	0.9	0.4	14	0.042	0.525
239	0.3	0.9	0.5	16	0.035	0.528
240	0.3	0.9	0.6	17	0.028	0.511
241	0.3	0.9	0.7	20	0.021	0.474
242	0.3	0.9	0.8	23	0.014	0.410

243	0.3	0.9	0.9	28	0.007	0.305
244	0.4	0.1	0.1	3	0.486	0.790
245	0.4	0.1	0.2	4	0.432	0.914
246	0.4	0.1	0.3	5	0.378	0.965
247	0.4	0.1	0.4	6	0.324	0.966
248	0.4	0.1	0.5	8	0.27	0.925
249	0.4	0.1	0.6	9	0.216	0.850
250	0.4	0.1	0.7	12	0.162	0.742
251	0.4	0.1	0.8	15	0.108	0.600
252	0.4	0.1	0.9	20	0.054	0.413
253	0.4	0.2	0.1	3	0.432	0.788
254	0.4	0.2	0.2	4	0.384	0.884
255	0.4	0.2	0.3	5	0.336	0.922
256	0.4	0.2	0.4	6	0.288	0.917
257	0.4	0.2	0.5	8	0.24	0.878
258	0.4	0.2	0.6	9	0.192	0.807
259	0.4	0.2	0.7	12	0.144	0.707
260	0.4	0.2	0.8	15	0.096	0.575
261	0.4	0.2	0.9	20	0.048	0.399
262	0.4	0.3	0.1	4	0.378	0.767
263	0.4	0.3	0.2	5	0.336	0.844
264	0.4	0.3	0.3	6	0.294	0.873
265	0.4	0.3	0.4	7	0.252	0.866
266	0.4	0.3	0.5	9	0.21	0.828
267	0.4	0.3	0.6	10	0.168	0.763
268	0.4	0.3	0.7	13	0.126	0.672
269	0.4	0.3	0.8	16	0.084	0.550
270	0.4	0.3	0.9	21	0.042	0.385
271	0.4	0.4	0.1	4	0.324	0.730
272	0.4	0.4	0.2	5	0.288	0.794
273	0.4	0.4	0.3	6	0.252	0.818
274	0.4	0.4	0.4	7	0.216	0.811
275	0.4	0.4	0.5	9	0.18	0.778
276	0.4	0.4	0.6	10	0.144	0.720
277	0.4	0.4	0.7	13	0.108	0.637
278	0.4	0.4	0.8	16	0.072	0.526
279	0.4	0.4	0.9	21	0.036	0.372
280	0.4	0.5	0.1	5	0.27	0.679
281	0.4	0.5	0.2	6	0.24	0.736
282	0.4	0.5	0.3	7	0.21	0.759
283	0.4	0.5	0.4	8	0.18	0.755

284	0.4	0.5	0.5	10	0.15	0.727
285	0.4	0.5	0.6	11	0.12	0.677
286	0.4	0.5	0.7	14	0.09	0.603
287	0.4	0.5	0.8	17	0.06	0.502
288	0.4	0.5	0.9	22	0.03	0.359
289	0.4	0.6	0.1	6	0.216	0.616
290	0.4	0.6	0.2	7	0.192	0.672
291	0.4	0.6	0.3	8	0.168	0.697
292	0.4	0.6	0.4	9	0.144	0.697
293	0.4	0.6	0.5	11	0.12	0.676
294	0.4	0.6	0.6	12	0.096	0.634
295	0.4	0.6	0.7	15	0.072	0.570
296	0.4	0.6	0.8	18	0.048	0.479
297	0.4	0.6	0.9	23	0.024	0.347
298	0.4	0.7	0.1	7	0.162	0.543
299	0.4	0.7	0.2	8	0.144	0.603
300	0.4	0.7	0.3	9	0.126	0.633
301	0.4	0.7	0.4	10	0.108	0.640
302	0.4	0.7	0.5	12	0.09	0.627
303	0.4	0.7	0.6	13	0.072	0.593
304	0.4	0.7	0.7	16	0.054	0.539
305	0.4	0.7	0.8	19	0.036	0.457
306	0.4	0.7	0.9	24	0.018	0.334
307	0.4	0.8	0.1	8	0.108	0.463
308	0.4	0.8	0.2	9	0.096	0.531
309	0.4	0.8	0.3	10	0.084	0.569
310	0.4	0.8	0.4	11	0.072	0.585
311	0.4	0.8	0.5	13	0.06	0.580
312	0.4	0.8	0.6	14	0.048	0.555
313	0.4	0.8	0.7	17	0.036	0.509
314	0.4	0.8	0.8	20	0.024	0.437
315	0.4	0.8	0.9	25	0.012	0.322
316	0.4	0.9	0.1	11	0.054	0.379
317	0.4	0.9	0.2	12	0.048	0.461
318	0.4	0.9	0.3	13	0.042	0.509
319	0.4	0.9	0.4	14	0.036	0.533
320	0.4	0.9	0.5	16	0.03	0.537
321	0.4	0.9	0.6	17	0.024	0.520
322	0.4	0.9	0.7	20	0.018	0.482
323	0.4	0.9	0.8	23	0.012	0.417
324	0.4	0.9	0.9	28	0.006	0.311

325	0.5	0.1	0.1	3	0.405	0.751
326	0.5	0.1	0.2	4	0.36	0.880
327	0.5	0.1	0.3	5	0.315	0.938
328	0.5	0.1	0.4	6	0.27	0.948
329	0.5	0.1	0.5	8	0.225	0.919
330	0.5	0.1	0.6	9	0.18	0.856
331	0.5	0.1	0.7	12	0.135	0.760
332	0.5	0.1	0.8	15	0.09	0.627
333	0.5	0.1	0.9	20	0.045	0.442
334	0.5	0.2	0.1	3	0.36	0.752
335	0.5	0.2	0.2	4	0.32	0.855
336	0.5	0.2	0.3	5	0.28	0.901
337	0.5	0.2	0.4	6	0.24	0.906
338	0.5	0.2	0.5	8	0.2	0.876
339	0.5	0.2	0.6	9	0.16	0.816
340	0.5	0.2	0.7	12	0.12	0.726
341	0.5	0.2	0.8	15	0.08	0.602
342	0.5	0.2	0.9	20	0.04	0.427
343	0.5	0.3	0.1	4	0.315	0.735
344	0.5	0.3	0.2	5	0.28	0.820
345	0.5	0.3	0.3	6	0.245	0.858
346	0.5	0.3	0.4	7	0.21	0.860
347	0.5	0.3	0.5	9	0.175	0.832
348	0.5	0.3	0.6	10	0.14	0.776
349	0.5	0.3	0.7	13	0.105	0.692
350	0.5	0.3	0.8	16	0.07	0.576
351	0.5	0.3	0.9	21	0.035	0.412
352	0.5	0.4	0.1	4	0.27	0.705
353	0.5	0.4	0.2	5	0.24	0.777
354	0.5	0.4	0.3	6	0.21	0.810
355	0.5	0.4	0.4	7	0.18	0.811
356	0.5	0.4	0.5	9	0.15	0.786
357	0.5	0.4	0.6	10	0.12	0.735
358	0.5	0.4	0.7	13	0.09	0.659
359	0.5	0.4	0.8	16	0.06	0.551
360	0.5	0.4	0.9	21	0.03	0.396
361	0.5	0.5	0.1	5	0.225	0.661
362	0.5	0.5	0.2	6	0.2	0.727
363	0.5	0.5	0.3	7	0.175	0.757
364	0.5	0.5	0.4	8	0.15	0.760
365	0.5	0.5	0.5	10	0.125	0.738

366	0.5	0.5	0.6	11	0.1	0.694
367	0.5	0.5	0.7	14	0.075	0.624
368	0.5	0.5	0.8	17	0.05	0.526
369	0.5	0.5	0.9	22	0.025	0.381
370	0.5	0.6	0.1	6	0.18	0.607
371	0.5	0.6	0.2	7	0.16	0.670
372	0.5	0.6	0.3	8	0.14	0.701
373	0.5	0.6	0.4	9	0.12	0.706
374	0.5	0.6	0.5	11	0.1	0.690
375	0.5	0.6	0.6	12	0.08	0.652
376	0.5	0.6	0.7	15	0.06	0.590
377	0.5	0.6	0.8	18	0.04	0.500
378	0.5	0.6	0.9	23	0.02	0.365
379	0.5	0.7	0.1	7	0.135	0.542
380	0.5	0.7	0.2	8	0.12	0.607
381	0.5	0.7	0.3	9	0.105	0.641
382	0.5	0.7	0.4	10	0.09	0.652
383	0.5	0.7	0.5	12	0.075	0.641
384	0.5	0.7	0.6	13	0.06	0.610
385	0.5	0.7	0.7	16	0.045	0.557
386	0.5	0.7	0.8	19	0.03	0.475
387	0.5	0.7	0.9	24	0.015	0.349
388	0.5	0.8	0.1	8	0.09	0.469
389	0.5	0.8	0.2	9	0.08	0.540
390	0.5	0.8	0.3	10	0.07	0.580
391	0.5	0.8	0.4	11	0.06	0.597
392	0.5	0.8	0.5	13	0.05	0.593
393	0.5	0.8	0.6	14	0.04	0.569
394	0.5	0.8	0.7	17	0.03	0.523
395	0.5	0.8	0.8	20	0.02	0.450
396	0.5	0.8	0.9	25	0.01	0.333
397	0.5	0.9	0.1	11	0.045	0.387
398	0.5	0.9	0.2	12	0.04	0.470
399	0.5	0.9	0.3	13	0.035	0.518
400	0.5	0.9	0.4	14	0.03	0.542
401	0.5	0.9	0.5	16	0.025	0.545
402	0.5	0.9	0.6	17	0.02	0.529
403	0.5	0.9	0.7	20	0.015	0.490
404	0.5	0.9	0.8	23	0.01	0.425
405	0.5	0.9	0.9	28	0.005	0.316
406	0.6	0.1	0.1	4	0.324	0.714

407	0.6	0.1	0.2	5	0.288	0.850
408	0.6	0.1	0.3	6	0.252	0.917
409	0.6	0.1	0.4	7	0.216	0.937
410	0.6	0.1	0.5	9	0.18	0.919
411	0.6	0.1	0.6	10	0.144	0.866
412	0.6	0.1	0.7	13	0.108	0.780
413	0.6	0.1	0.8	16	0.072	0.654
414	0.6	0.1	0.9	21	0.036	0.471
415	0.6	0.2	0.1	4	0.288	0.718
416	0.6	0.2	0.2	5	0.256	0.830
417	0.6	0.2	0.3	6	0.224	0.885
418	0.6	0.2	0.4	7	0.192	0.899
419	0.6	0.2	0.5	9	0.16	0.880
420	0.6	0.2	0.6	10	0.128	0.830
421	0.6	0.2	0.7	13	0.096	0.748
422	0.6	0.2	0.8	16	0.064	0.629
423	0.6	0.2	0.9	21	0.032	0.454
424	0.6	0.3	0.1	5	0.252	0.708
425	0.6	0.3	0.2	6	0.224	0.801
426	0.6	0.3	0.3	7	0.196	0.848
427	0.6	0.3	0.4	8	0.168	0.859
428	0.6	0.3	0.5	10	0.14	0.839
429	0.6	0.3	0.6	11	0.112	0.792
430	0.6	0.3	0.7	14	0.084	0.715
431	0.6	0.3	0.8	17	0.056	0.603
432	0.6	0.3	0.9	22	0.028	0.437
433	0.6	0.4	0.1	5	0.216	0.684
434	0.6	0.4	0.2	6	0.192	0.765
435	0.6	0.4	0.3	7	0.168	0.805
436	0.6	0.4	0.4	8	0.144	0.815
437	0.6	0.4	0.5	10	0.12	0.797
438	0.6	0.4	0.6	11	0.096	0.752
439	0.6	0.4	0.7	14	0.072	0.681
440	0.6	0.4	0.8	17	0.048	0.576
441	0.6	0.4	0.9	22	0.024	0.419
442	0.6	0.5	0.1	6	0.18	0.648
443	0.6	0.5	0.2	7	0.16	0.721
444	0.6	0.5	0.3	8	0.14	0.758
445	0.6	0.5	0.4	9	0.12	0.768
446	0.6	0.5	0.5	11	0.1	0.752
447	0.6	0.5	0.6	12	0.08	0.712

448	0.6	0.5	0.7	15	0.06	0.646
449	0.6	0.5	0.8	18	0.04	0.549
450	0.6	0.5	0.9	23	0.02	0.401
451	0.6	0.6	0.1	7	0.144	0.602
452	0.6	0.6	0.2	8	0.128	0.671
453	0.6	0.6	0.3	9	0.112	0.707
454	0.6	0.6	0.4	10	0.096	0.718
455	0.6	0.6	0.5	12	0.08	0.705
456	0.6	0.6	0.6	13	0.064	0.670
457	0.6	0.6	0.7	16	0.048	0.611
458	0.6	0.6	0.8	19	0.032	0.521
459	0.6	0.6	0.9	24	0.016	0.382
460	0.6	0.7	0.1	8	0.108	0.544
461	0.6	0.7	0.2	9	0.096	0.613
462	0.6	0.7	0.3	10	0.084	0.651
463	0.6	0.7	0.4	11	0.072	0.665
464	0.6	0.7	0.5	13	0.06	0.657
465	0.6	0.7	0.6	14	0.048	0.627
466	0.6	0.7	0.7	17	0.036	0.574
467	0.6	0.7	0.8	20	0.024	0.492
468	0.6	0.7	0.9	25	0.012	0.363
469	0.6	0.8	0.1	9	0.072	0.476
470	0.6	0.8	0.2	10	0.064	0.549
471	0.6	0.8	0.3	11	0.056	0.591
472	0.6	0.8	0.4	12	0.048	0.609
473	0.6	0.8	0.5	14	0.04	0.606
474	0.6	0.8	0.6	15	0.032	0.583
475	0.6	0.8	0.7	18	0.024	0.537
476	0.6	0.8	0.8	21	0.016	0.463
477	0.6	0.8	0.9	26	0.008	0.343
478	0.6	0.9	0.1	12	0.036	0.396
479	0.6	0.9	0.2	13	0.032	0.478
480	0.6	0.9	0.3	14	0.028	0.527
481	0.6	0.9	0.4	15	0.024	0.551
482	0.6	0.9	0.5	17	0.02	0.554
483	0.6	0.9	0.6	18	0.016	0.537
484	0.6	0.9	0.7	21	0.012	0.498
485	0.6	0.9	0.8	24	0.008	0.432
486	0.6	0.9	0.9	29	0.004	0.322
487	0.7	0.1	0.1	4	0.243	0.681
488	0.7	0.1	0.2	5	0.216	0.825

489	0.7	0.1	0.3	6	0.189	0.902
490	0.7	0.1	0.4	7	0.162	0.932
491	0.7	0.1	0.5	9	0.135	0.923
492	0.7	0.1	0.6	10	0.108	0.881
493	0.7	0.1	0.7	13	0.081	0.803
494	0.7	0.1	0.8	16	0.054	0.683
495	0.7	0.1	0.9	21	0.027	0.499
496	0.7	0.2	0.1	4	0.216	0.690
497	0.7	0.2	0.2	5	0.192	0.811
498	0.7	0.2	0.3	6	0.168	0.875
499	0.7	0.2	0.4	7	0.144	0.899
500	0.7	0.2	0.5	9	0.12	0.888
501	0.7	0.2	0.6	10	0.096	0.846
502	0.7	0.2	0.7	13	0.072	0.771
503	0.7	0.2	0.8	16	0.048	0.657
504	0.7	0.2	0.9	21	0.024	0.480
505	0.7	0.3	0.1	5	0.189	0.685
506	0.7	0.3	0.2	6	0.168	0.788
507	0.7	0.3	0.3	7	0.147	0.843
508	0.7	0.3	0.4	8	0.126	0.862
509	0.7	0.3	0.5	10	0.105	0.851
510	0.7	0.3	0.6	11	0.084	0.810
511	0.7	0.3	0.7	14	0.063	0.739
512	0.7	0.3	0.8	17	0.042	0.629
513	0.7	0.3	0.9	22	0.021	0.461
514	0.7	0.4	0.1	5	0.162	0.669
515	0.7	0.4	0.2	6	0.144	0.758
516	0.7	0.4	0.3	7	0.126	0.806
517	0.7	0.4	0.4	8	0.108	0.822
518	0.7	0.4	0.5	10	0.09	0.811
519	0.7	0.4	0.6	11	0.072	0.772
520	0.7	0.4	0.7	14	0.054	0.705
521	0.7	0.4	0.8	17	0.036	0.601
522	0.7	0.4	0.9	22	0.018	0.442
523	0.7	0.5	0.1	6	0.135	0.640
524	0.7	0.5	0.2	7	0.12	0.720
525	0.7	0.5	0.3	8	0.105	0.764
526	0.7	0.5	0.4	9	0.09	0.779
527	0.7	0.5	0.5	11	0.075	0.768
528	0.7	0.5	0.6	12	0.06	0.732
529	0.7	0.5	0.7	15	0.045	0.669

530	0.7	0.5	0.8	18	0.03	0.572
531	0.7	0.5	0.9	23	0.015	0.421
532	0.7	0.6	0.1	7	0.108	0.601
533	0.7	0.6	0.2	8	0.096	0.675
534	0.7	0.6	0.3	9	0.084	0.716
535	0.7	0.6	0.4	10	0.072	0.731
536	0.7	0.6	0.5	12	0.06	0.722
537	0.7	0.6	0.6	13	0.048	0.690
538	0.7	0.6	0.7	16	0.036	0.632
539	0.7	0.6	0.8	19	0.024	0.541
540	0.7	0.6	0.9	24	0.012	0.399
541	0.7	0.7	0.1	8	0.081	0.550
542	0.7	0.7	0.2	9	0.072	0.622
543	0.7	0.7	0.3	10	0.063	0.663
544	0.7	0.7	0.4	11	0.054	0.679
545	0.7	0.7	0.5	13	0.045	0.673
546	0.7	0.7	0.6	14	0.036	0.645
547	0.7	0.7	0.7	17	0.027	0.592
548	0.7	0.7	0.8	20	0.018	0.509
549	0.7	0.7	0.9	25	0.009	0.377
550	0.7	0.8	0.1	9	0.054	0.486
551	0.7	0.8	0.2	10	0.048	0.560
552	0.7	0.8	0.3	11	0.042	0.603
553	0.7	0.8	0.4	12	0.036	0.622
554	0.7	0.8	0.5	14	0.03	0.620
555	0.7	0.8	0.6	15	0.024	0.597
556	0.7	0.8	0.7	18	0.018	0.551
557	0.7	0.8	0.8	21	0.012	0.475
558	0.7	0.8	0.9	26	0.006	0.353
559	0.7	0.9	0.1	12	0.027	0.406
560	0.7	0.9	0.2	13	0.024	0.487
561	0.7	0.9	0.3	14	0.021	0.536
562	0.7	0.9	0.4	15	0.018	0.560
563	0.7	0.9	0.5	17	0.015	0.563
564	0.7	0.9	0.6	18	0.012	0.546
565	0.7	0.9	0.7	21	0.009	0.506
566	0.7	0.9	0.8	24	0.006	0.439
567	0.7	0.9	0.9	29	0.003	0.327
568	0.8	0.1	0.1	5	0.162	0.655
569	0.8	0.1	0.2	6	0.144	0.809
570	0.8	0.1	0.3	7	0.126	0.894

571	0.8	0.1	0.4	8	0.108	0.933
572	0.8	0.1	0.5	10	0.09	0.934
573	0.8	0.1	0.6	11	0.072	0.900
574	0.8	0.1	0.7	14	0.054	0.828
575	0.8	0.1	0.8	17	0.036	0.712
576	0.8	0.1	0.9	22	0.018	0.526
577	0.8	0.2	0.1	5	0.144	0.670
578	0.8	0.2	0.2	6	0.128	0.799
579	0.8	0.2	0.3	7	0.112	0.872
580	0.8	0.2	0.4	8	0.096	0.904
581	0.8	0.2	0.5	10	0.08	0.902
582	0.8	0.2	0.6	11	0.064	0.867
583	0.8	0.2	0.7	14	0.048	0.797
584	0.8	0.2	0.8	17	0.032	0.685
585	0.8	0.2	0.9	22	0.016	0.506
586	0.8	0.3	0.1	6	0.126	0.671
587	0.8	0.3	0.2	7	0.112	0.782
588	0.8	0.3	0.3	8	0.098	0.845
589	0.8	0.3	0.4	9	0.084	0.871
590	0.8	0.3	0.5	11	0.07	0.866
591	0.8	0.3	0.6	12	0.056	0.831
592	0.8	0.3	0.7	15	0.042	0.764
593	0.8	0.3	0.8	18	0.028	0.656
594	0.8	0.3	0.9	23	0.014	0.485
595	0.8	0.4	0.1	6	0.108	0.660
596	0.8	0.4	0.2	7	0.096	0.757
597	0.8	0.4	0.3	8	0.084	0.812
598	0.8	0.4	0.4	9	0.072	0.834
599	0.8	0.4	0.5	11	0.06	0.828
600	0.8	0.4	0.6	12	0.048	0.794
601	0.8	0.4	0.7	15	0.036	0.729
602	0.8	0.4	0.8	18	0.024	0.626
603	0.8	0.4	0.9	23	0.012	0.463
604	0.8	0.5	0.1	7	0.09	0.638
605	0.8	0.5	0.2	8	0.08	0.725
606	0.8	0.5	0.3	9	0.07	0.773
607	0.8	0.5	0.4	10	0.06	0.793
608	0.8	0.5	0.5	12	0.05	0.786
609	0.8	0.5	0.6	13	0.04	0.753
610	0.8	0.5	0.7	16	0.03	0.692
611	0.8	0.5	0.8	19	0.02	0.595

612	0.8	0.5	0.9	24	0.01	0.440
613	0.8	0.6	0.1	8	0.072	0.605
614	0.8	0.6	0.2	9	0.064	0.683
615	0.8	0.6	0.3	10	0.056	0.728
616	0.8	0.6	0.4	11	0.048	0.747
617	0.8	0.6	0.5	13	0.04	0.740
618	0.8	0.6	0.6	14	0.032	0.710
619	0.8	0.6	0.7	17	0.024	0.653
620	0.8	0.6	0.8	20	0.016	0.562
621	0.8	0.6	0.9	25	0.008	0.416
622	0.8	0.7	0.1	9	0.054	0.558
623	0.8	0.7	0.2	10	0.048	0.633
624	0.8	0.7	0.3	11	0.042	0.676
625	0.8	0.7	0.4	12	0.036	0.695
626	0.8	0.7	0.5	14	0.03	0.690
627	0.8	0.7	0.6	15	0.024	0.663
628	0.8	0.7	0.7	18	0.018	0.610
629	0.8	0.7	0.8	21	0.012	0.526
630	0.8	0.7	0.9	26	0.006	0.390
631	0.8	0.8	0.1	10	0.036	0.497
632	0.8	0.8	0.2	11	0.032	0.572
633	0.8	0.8	0.3	12	0.028	0.616
634	0.8	0.8	0.4	13	0.024	0.636
635	0.8	0.8	0.5	15	0.02	0.634
636	0.8	0.8	0.6	16	0.016	0.611
637	0.8	0.8	0.7	19	0.012	0.565
638	0.8	0.8	0.8	22	0.008	0.488
639	0.8	0.8	0.9	27	0.004	0.363
640	0.8	0.9	0.1	13	0.018	0.415
641	0.8	0.9	0.2	14	0.016	0.496
642	0.8	0.9	0.3	15	0.014	0.545
643	0.8	0.9	0.4	16	0.012	0.569
644	0.8	0.9	0.5	18	0.01	0.572
645	0.8	0.9	0.6	19	0.008	0.554
646	0.8	0.9	0.7	22	0.006	0.514
647	0.8	0.9	0.8	25	0.004	0.446
648	0.8	0.9	0.9	30	0.002	0.333
649	0.9	0.1	0.1	6	0.081	0.639
650	0.9	0.1	0.2	7	0.072	0.802
651	0.9	0.1	0.3	8	0.063	0.895
652	0.9	0.1	0.4	9	0.054	0.942

653	0.9	0.1	0.5	11	0.045	0.951
654	0.9	0.1	0.6	12	0.036	0.923
655	0.9	0.1	0.7	15	0.027	0.856
656	0.9	0.1	0.8	18	0.018	0.741
657	0.9	0.1	0.9	23	0.009	0.552
658	0.9	0.2	0.1	6	0.072	0.659
659	0.9	0.2	0.2	7	0.064	0.797
660	0.9	0.2	0.3	8	0.056	0.877
661	0.9	0.2	0.4	9	0.048	0.916
662	0.9	0.2	0.5	11	0.04	0.920
663	0.9	0.2	0.6	12	0.032	0.890
664	0.9	0.2	0.7	15	0.024	0.824
665	0.9	0.2	0.8	18	0.016	0.713
666	0.9	0.2	0.9	23	0.008	0.530
667	0.9	0.3	0.1	7	0.063	0.665
668	0.9	0.3	0.2	8	0.056	0.784
669	0.9	0.3	0.3	9	0.049	0.853
670	0.9	0.3	0.4	10	0.042	0.886
671	0.9	0.3	0.5	12	0.035	0.886
672	0.9	0.3	0.6	13	0.028	0.855
673	0.9	0.3	0.7	16	0.021	0.791
674	0.9	0.3	0.8	19	0.014	0.683
675	0.9	0.3	0.9	24	0.007	0.508
676	0.9	0.4	0.1	7	0.054	0.660
677	0.9	0.4	0.2	8	0.048	0.763
678	0.9	0.4	0.3	9	0.042	0.823
679	0.9	0.4	0.4	10	0.036	0.851
680	0.9	0.4	0.5	12	0.03	0.848
681	0.9	0.4	0.6	13	0.024	0.817
682	0.9	0.4	0.7	16	0.018	0.755
683	0.9	0.4	0.8	19	0.012	0.651
684	0.9	0.4	0.9	24	0.006	0.484
685	0.9	0.5	0.1	8	0.045	0.643
686	0.9	0.5	0.2	9	0.04	0.734
687	0.9	0.5	0.3	10	0.035	0.787
688	0.9	0.5	0.4	11	0.03	0.810
689	0.9	0.5	0.5	13	0.025	0.807
690	0.9	0.5	0.6	14	0.02	0.776
691	0.9	0.5	0.7	17	0.015	0.716
692	0.9	0.5	0.8	20	0.01	0.618
693	0.9	0.5	0.9	25	0.005	0.459

694	0.9	0.6	0.1	9	0.036	0.613
695	0.9	0.6	0.2	10	0.032	0.695
696	0.9	0.6	0.3	11	0.028	0.744
697	0.9	0.6	0.4	12	0.024	0.764
698	0.9	0.6	0.5	14	0.02	0.760
699	0.9	0.6	0.6	15	0.016	0.731
700	0.9	0.6	0.7	18	0.012	0.674
701	0.9	0.6	0.8	21	0.008	0.582
702	0.9	0.6	0.9	26	0.004	0.432
703	0.9	0.7	0.1	10	0.027	0.570
704	0.9	0.7	0.2	11	0.024	0.646
705	0.9	0.7	0.3	12	0.021	0.692
706	0.9	0.7	0.4	13	0.018	0.711
707	0.9	0.7	0.5	15	0.015	0.708
708	0.9	0.7	0.6	16	0.012	0.681
709	0.9	0.7	0.7	19	0.009	0.628
710	0.9	0.7	0.8	22	0.006	0.543
711	0.9	0.7	0.9	27	0.003	0.403
712	0.9	0.8	0.1	11	0.018	0.509
713	0.9	0.8	0.2	12	0.016	0.585
714	0.9	0.8	0.3	13	0.014	0.629
715	0.9	0.8	0.4	14	0.012	0.650
716	0.9	0.8	0.5	16	0.01	0.649
717	0.9	0.8	0.6	17	0.008	0.626
718	0.9	0.8	0.7	20	0.006	0.578
719	0.9	0.8	0.8	23	0.004	0.500
720	0.9	0.8	0.9	28	0.002	0.372
721	0.9	0.9	0.1	14	0.009	0.425
722	0.9	0.9	0.2	15	0.008	0.506
723	0.9	0.9	0.3	16	0.007	0.554
724	0.9	0.9	0.4	17	0.006	0.578
725	0.9	0.9	0.5	19	0.005	0.581
726	0.9	0.9	0.6	20	0.004	0.563
727	0.9	0.9	0.7	23	0.003	0.522
728	0.9	0.9	0.8	26	0.002	0.453
729	0.9	0.9	0.9	31	0.001	0.338

Lampiran D. Algoritma ECDSA

ECDSA merupakan algoritma *digital signature* yang dianalogikan menyerupai kurva elips. ECDSA dikembangkan oleh National Institute of Standards and Technology (NIST) sejak tahun 1992 (Johnson et al. 2001). Proses yang dilewati pada ECDSA terbagi menjadi tiga, yaitu pembangkitan kunci, pembuatan signature pada sisi pengirim dan verifikasi signature pada sisi penerima.

A.1 Pembangkitan Kunci

Pasangan kunci berasosiasi dengan kelompok kurva elips dengan parameter domain *D = (p,A,B,G,n,h)* terdiri dari integer *p* yang spesifik pada Fp, dua elemen $A, B \in F_p$ dan memenuhi kurva elips $E(F_p)$ pada persamaan (Johnson et al. 2001).

$$\mathrm{E}: y^2 \equiv x^3 + \mathrm{ax} + \mathrm{b} \ (\mathrm{mod\ p})$$

Base point dari *G* terdiri dari koordinat titik $G = (x_G, y_G)$ yang terletak pada kurva $E(F_p)$. Tupple kelima terdapat variabel *n* yang merupakan *order* dari *base point G* yang terdapat pada kurva elips. Sedangkan *h* adalah kofaktor yang dirumuskan sebagai $h = E(F_p)/n$.

Ukuran signature yang dihasilkan bergantung pada panjang kunci. Sedangkan ukuran signature sendiri mempengaruhi beban komunikasi yang berdampak pada kebutuhan *bandwidth*. Seperti yang telah diketahui, jaringan sensor memiliki keterbatasan dalam *bandwidth* maupun ukuran paket yang dipertukarkan. Oleh karena itu, ukuran signature dibuat serandah mungkin tetapi dengan keamanan yang tetap terjamin. Hingga saat ini, pada penelitian yang memanfaatkan ECDSA ukuran signature yang paling sesuai digunakan untuk jaringan sensor adalah 40 byte. Sehingga panjang kunci yang akan digunakan adalah 160 bits. Mengingat dibutuhkan dua bilangan prima dengan range $2^n - 1 = 2^{160} - 1$, maka dibutuhkan 320 bits untuk menampung kedua bilangan tersebut sehingga signature yang dihasilkan adalah 320 bits atau 40 byte. Dari

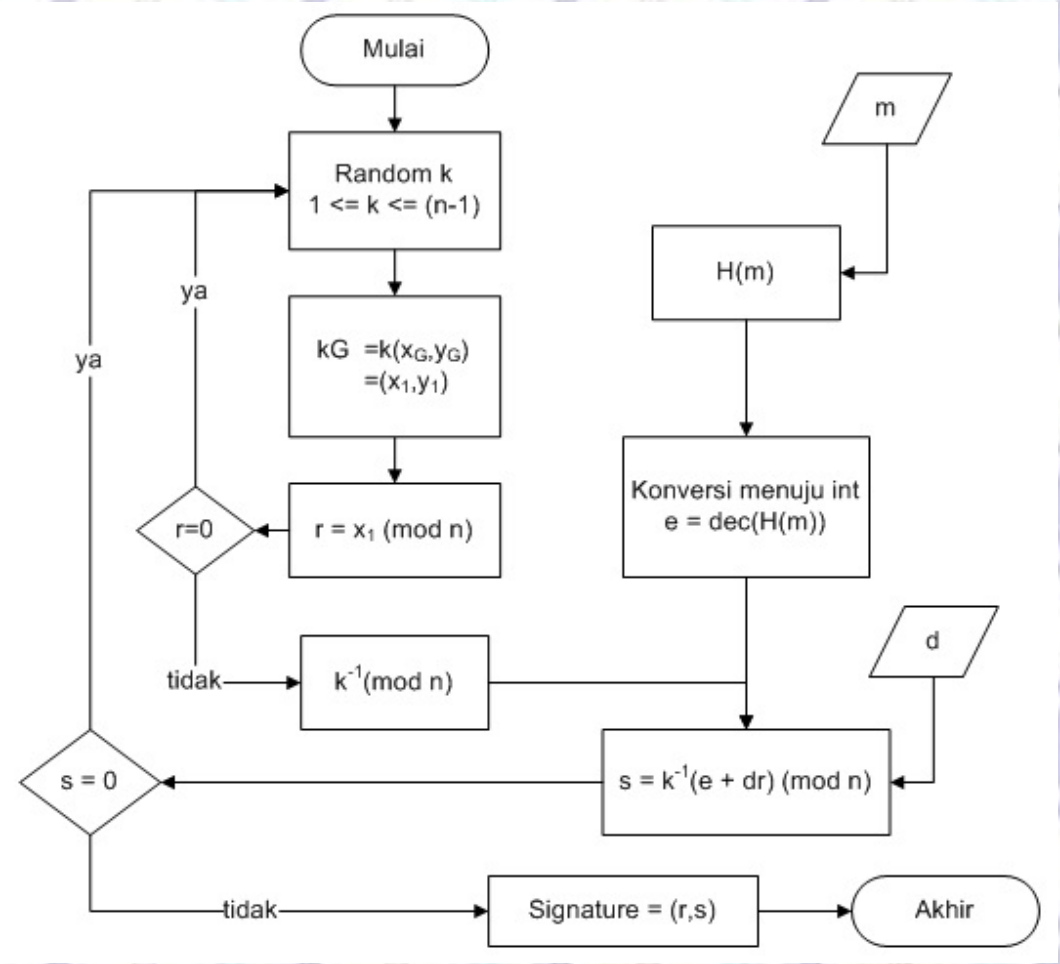
pilihan standar yang ada, penelitian ini menggunakan RFC5639 tepatnya brainpoolP160r1, dengan nilai sbb (Merkle & Lochter 2010) :

- p = E95E4A5F737059DC60DFC7AD95B3D8139515620F
- A = 340E7BE2A280EB74E2BE61BADA745D97E8F7C300
- B = 1E589A8595423412134FAA2DBDEC95C8D8675E58
- x_G = BED5AF16EA3F6A4F62938C4631EB5AF7BDBCDBC3
- y_G = 1667CB477A1A8EC338F94741669C976316DA6321
- n = E95E4A5F737059DC60DF5991D45029409E60FC09
- h = 1

Dari domain parameter tersebut, pengirim memilih bilangan random integer *d* antara 1 hingga (*n*-1) yang selanjutnya akan digunakan sebagai *private key*. Kemudian dijalankan operasi perkalian skalar antara *private key* dengan *base point* yang hasilnya merupakan *public key* $Q = (Q_x, Q_y)$.

Pembuatan Signature

Pada proses pembuatan signature di sisi penerima terdapat beberapa operasi utama yaitu pembangkitan bilangan random, operasi hash khususnya menggunakan SHA-1, perkalian skalar dan invers. Proses ini diawali dengan pembangkitan bilangan random *k* yang memiliki nilai berkisar antara 1 sampai (*n*-1). Kemudian pada nilai tersebut akan dijalankan perkalian skalar dengan base point *G*. Dari perkalian tersebut dihasilkan nilai x_1 dany_1. Variabel x_1 kemudian di-moduluskan dengan *n* untuk mendapatkan nilai *r*. Jika *r* bernilai 0 maka proses pembuatan signature harus diulang dari awal. Namun jika tidak, proses dapat diteruskan. Proses dilanjutkan dengan perhitungan nilai *s*. Langkah pertama yaitu invers-kan nilai *k* mod *n*. Kemudian hashing pesan menggunakan SHA-1 dan konversikan ke dalam bilangan desimal yang selanjutnya disimpan ke dalam variabel *e*. Dari kedua hasil tersebut, hitung nilai *s* dengan cara mengalikan invers dari *k* dengan (*e*+*d.r*) mod *n*. Dari hasil yang di dapat, cek apakah *s* bernilai 0.

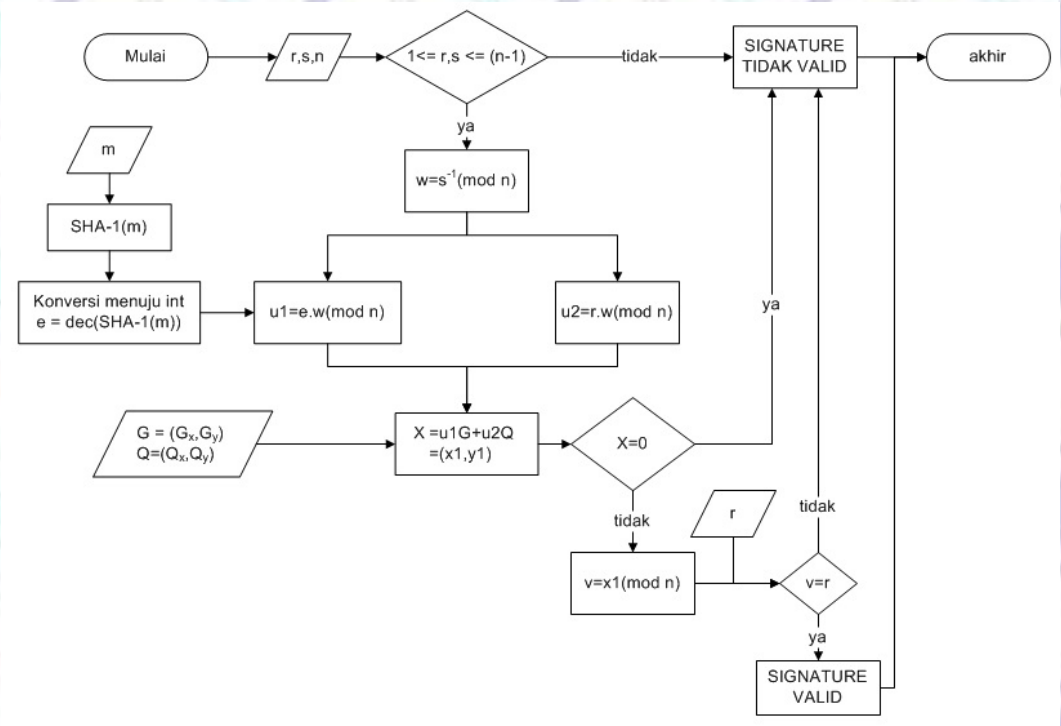


Gambar A.1 Pembuatan Signature pada ECDSA

Pengecekan tersebut dijalankan karena nilai dari *r* dan *s* harus berkisar antara 1 hingga (2^{160}-1). Jika *s* bernilai 0, ulangi proses dari awal. Namun jika tidak, signature terbentuk dari nilai *r* dan *s*. Ilustrasi dari pembentukan signature dapat dilihat pada Gambar A.1.

Verifikasi Signature

Proses inisialisasi diperlukan sebagai awalan verifikasi signature untuk menyimpan domain parameter kurva elips di sisi penerima. Variabel tersebut yaitu kunci publik dari pengirim *Q*, base point kurva elips *G* pada F_p dan order *n* dari base point *G*. Saat penerima menerima paket dari pengirim, minimal harus terdapat dua komponen utama yaitu signature yang terdiri dari *r*, *s* dan

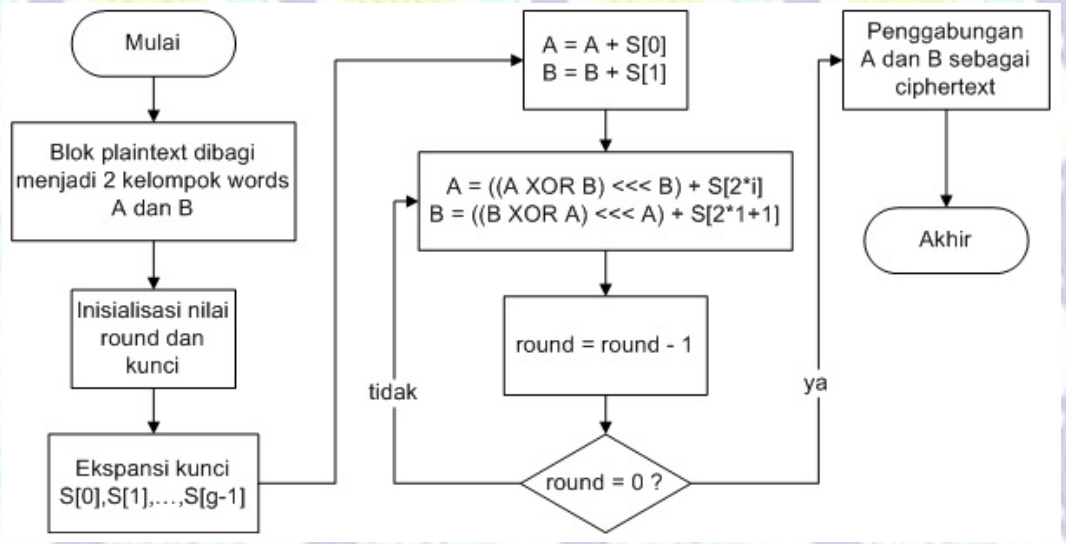


Gambar A.2 Verifikasi signature pada ECDSA

pesan *m*. Ilustrasi dari verifikasi signature dapat dilihat pada Gambar A.2. Proses verifikasi signature diawali dengan pengecekan nilai *r* dan *s* yang diharuskan berkisar antara 1 hingga (2^{160}-1). Jika tidak memenuhi, dapat dipastikansignature tidak valid. Dilanjutkan dengan perhitungan variabel *w*, yang dihasilkan dari proses invers nilai *s* mod *n*. Nilai ini kemudian digunakan untuk menentukan nilai u_1 dan u_2. Nilai u_1 merupakan hasil perkalian antara *e* dengan *w* mod *n*. Sama dengan proses pembentukan signature pada penjelesan sebelumnya, *e* merupakan nilai desimal dari hashing pesan yang diterima. Kemudian, nilai u_2 diperoleh dari perkalian antara bagian signature *r* dengan *w* mod *n*. Kedua nilai tersebut kemudian dikalikan dengan base point juga kunci publik sehingga didapatkan nilai $X= u_1.G+ u_2.Q$. Karena *G* dan *Q* merupakan titik yang terdiri dari koordinat sumbu *x* dan *y* sedangkan u_1 dan u_2 adalah besaran skalar, maka *X* juga terdiri dari dua komponen yaitu x_1 yang mewakili sumbu *x* dan y_1 yang mewakili sumbu *y*. Jika *X* bernilai 0 maka signature menjadi tidak valid. Akan tetapi jika sebaliknya, perlu dilakukan pengecekan apakah nilai x_1 mod *n* sama dengan nilai *r*. Jika ya maka signature valid dan berlaku sebaliknya.

Lampiran E. Algoritma RC5

RC5 merupakan algoritma enkripsi yang termasuk dalam simetrik kriptografi. Hal tersebut dikarenakan kunci yang digunakan untuk enkripsi maupun dekripsi adalah sama. Selain itu, berdasarkan mekanisme pembangunannya RC5 termasuk dalam block cipher. Di dalam mekanisme pengelolaan plaintext menjadi ciphertext, RC5 membagi-bagi menjadi blok-blok yang sama besar. Masing masing blok tersebut akan dienkripsi menggunakan kunci yang telah didefinisikan sebelumnya. Pada tahap inisialisasi parameter yang perlu didefinisikan yaitu word (*w*), round (*r*) dan panjang kunci (*b*) sehingga terbentuk RC5 *r/w/b*. Detail implementasi enkripsi dari RC5 dijelaskan pada Gambar B.1. Proses enkripsi pada RC5 diawali dengan pembagian pesan menjadi 2 kelompok. Masing masing kelompok ditambahkan dengan bagian dari kunci. Kemudian dijalankan iterasi XOR, penggeseran bit dan penambahan dengan sisa kunci sebanyak jumlah round yang telah didefinisikan sebelumnya. Simbol <<< *A* merupakan operasi penggeseran bit sebanyak *A* kali. Proses ini diakhiri dengan penggabungan kelompok *A* dan *B* sebagai hasil enkripsi dari pesan yang akan dikirim.



Gambar B.1 Proses enkripsi dengan metode RC5

Halaman ini sengaja dikosongkan

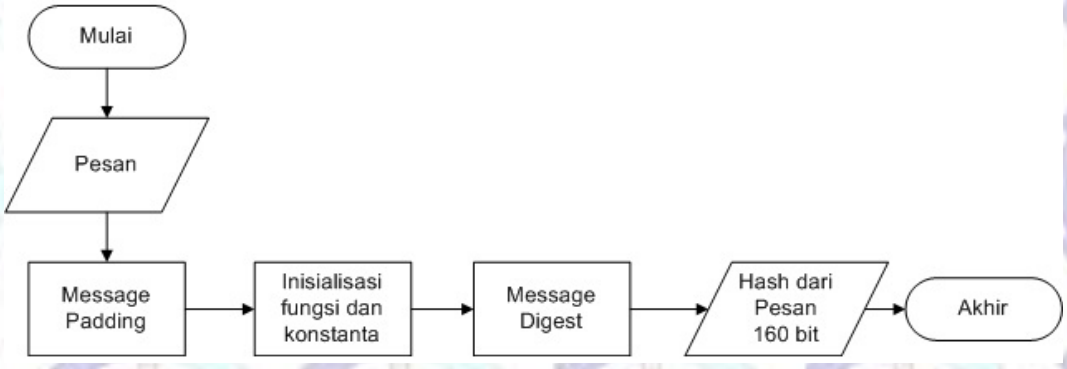
Lampiran F. Algoritma SHA1

SHA1 dipublikasikan oleh NIST sejak tahun 1995 (Eastlake & Jones 2001). Fungsi ini merupakan salah satu operasi hash yang paling banyak digunakan. Hal tersebut dikarenakan ukuran luarannya yang kecil yaitu sebesar 160 bit atau 20 byte. Dibandingkan dengan fungsi hash lainnya yang lebih besar, fungsi ini sesuai dengan karakteristik pada mesin dengan sumber daya rendah. Proses yang harus dijalankan pada pembangkitan nilai SHA1 dari sebuah pesan dijelaskan pada Gambar C.1.

Pembangkitan SHA1 diawali dengan pre-processing dari pesan agar sesuai dengan blok yang ditentukan yaitu kelipatan 512. Tahap ini disebut dengan message padding yaitu penambahan pesan hingga memenuhi blok minimum dari proses pembangkitan SHA1. Detail message padding dijalankan dengan cara (Eastlake & Jones 2001):

a. Penambahan satu bit bernilai '1' di belakang pesan yang kurang dari 512
b. Penambahan sejumlah bit bernilai '0' hingga tersisa 64 bit kosong di akhir blok
c. Penambahan 64 bit yang bernilai integer dari panjang pesan asli

Setelah memenuhi blok pesan, tahap selanjutnya yaitu inisialisasi fungsi dan konstanta yang akan digunakan. Pertama yaitu pendefinisian fungsi lojik sebanyak 80 secara berurutan dengan format $f(t;B,C,D)$ dengan t merupakan urutan dari fungsi lojik yang bernilai $0 \leq t \leq 79$ dan B,C,D merupakan words dengan panjang 32-bit. Luaran dari fungsi lojik juga memiliki panjang 32-bit words. Fungsi lojik tersebut terdiri dari (Eastlake & Jones 2001) :



Gambar C.1 Proses pembangkitan fungsi SHA-1

a. $f(t;B,C,D) = (B \text{ AND } C) \text{ OR } ((\text{NOT } B) \text{ AND } D) \quad (0 \leq t \leq 19)$

b. $f(t;B,C,D) = B \text{ XOR } C \text{ XOR } D \quad (20 \leq t \leq 39)$

c. $f(t;B,C,D) = (B \text{ AND } C) \text{ OR } (B \text{ AND } D) \text{ OR } (C \text{ AND } D) \quad (40 \leq t \leq 59)$

d. $f(t;B,C,D) = B \text{ XOR } C \text{ XOR } D \quad (60 \leq t \leq 79)$

Inisialisasi kedua berguna untuk words sebanyak 80 dengan nilai secara berurutan didefinisikan sebagai (Eastlake & Jones 2001):

a. $K(t)$ = 5A827999 $(0 \leq t \leq 19)$

b. $K(t)$ = 6ED9EBA1 $(20 \leq t \leq 39)$

c. $K(t)$ = 8F1BBCDC $(40 \leq t \leq 59)$

d. $K(t)$ = CA62C1D6 $(60 \leq t \leq 79)$

Inisialisasi ketiga diperlukan untuk nilai awal dari setiap blok yang akan digunakan untuk proses selanjutnya. Nilai tersebut didefinisikan pada (Eastlake & Jones 2001) :

a. *H0* = 67452301

b. *H1* = EFCDAB89

c. *H2* = 98BADCFE

d. *H3* = 10325476

e. *H4* = C3D2E1F0

Langkah terakhir pada tahap ini adalah memasukkan pesan yang sudah melewati proses padding sebelumnya pada 16 blok 32-bit words. Variabel yang digunakan untuk menampungnya adalah *M(1), M(2),..., M(n)*

Proses utama dari pembangkitan fungsi hash yaitu Message Digest. Proses ini dijalankan untuk setiap word dari pesan *M(i)*. Langkah yang harus dijalankan pada tiap iterasi dijelaskan sebagai berikut (Eastlake & Jones 2001):

a. Pembagian *M(i)* pada 16 words *W*(0), *W*(1), ... , *W*(15) dengan *W*(0) adalah word paling kiri.

b. Untuk *t*=16 hingga 79 jalankan $W(t) = S^1(W(t\text{-}3) \text{ XOR } W(t\text{-}8) \text{ XOR } W(t\text{-}14) \text{ XOR } W(t\text{-}16))$

c. Penugasan nilai pada variabel berikut *A = H0, B = H1, C = H2, D = H3, E = H4*

d. Untuk *t*=0 hingga 79 jalankan

$$TEMP = S^5(A) + f(t;B,C,D) + E + W(t) + K(t)$$

$$E = D;\ D = C;\ C = S^{30}(B);\ B = A;\ A = TEMP;$$

e. Penugasan nilai pada variabel berikut

H0 = H0 + A, H1 = H1 + B, H2 = H2 + C, H3 = H3 + D, H4 = H4 + E

Langkah tersebut diulang hingga *M*(*n*). Hasil akhir dari message digest ada 160 bit string yang direpresentasikan dengan penggabungan kelima words *H0,H1,H2,H3,H4*.

Halaman ini sengaja dikosongkan

Lampiran G. Kode program simulasi skema puzzle dinamis pada NS3

Skema M-DCP menggunakan TinySet pada NS3

```
#include <fstream>
#include <iostream>
#include <string>
#include <cstdlib>
#include <vector>
#include <bitset>
#include <ctime>
#include <ostream>
#include <ctime>
#include "ns3/hmac_sha1.h"
#include <memory>
#include <ns3/lr-wpan-module.h>
#include <ns3/mobility-module.h>
#include "ns3/core-module.h"
#include "ns3/internet-module.h"
#include "ns3/csma-module.h"
#include "ns3/csma-helper.h"
#include "ns3/node.h"
#include "ns3/applications-module.h"
#include "ns3/ipv6-static-routing-helper.h"
#include "ns3/simulator.h"
#include "ns3/ipv6-routing-table-entry.h"
#include "ns3/sixlowpan-module.h"
#include "ns3/netanim-module.h"
#include "ns3/address-utils.h"
#include "ns3/sha1.h"
#include "ns3/md5.h"
#include "ns3/system-wall-clock-ms.h"
#include "ns3/flow-monitor-module.h"
#include "ns3/flow-monitor-helper.h"
#include "ns3/data-rate.h"
#include "ns3/ecdsa.h"
#include "ns3/numbertheory.h"
#include "ns3/bigInt.h"
#include "ns3/BigIntegerLibrary.h"
#include "ns3/ellipticcurve.h"
#include "ns3/der.h"
#include "ns3/curves.h"
#include "ns3/util.h"
#include "ns3/keys.h"
#include "ns3/rc5.h"
#include "ns3/der.h"
#include "ns3/TinyTable.h"
#include <ctime>

using namespace ns3;
struct timespec start, finish,start2,finish2,start3,finish3;
```

```
double elapsed,elapsed2;

int pesanke = 0;
const uint16_t nl1 = 3;
const uint16_t nl2 = 2;
const uint16_t nl3 = 2;
uint16_t NodesCount =  nl1 + (nl1*nl2) + (nl1*nl2*nl3);
Ptr<Node> sink = CreateObject<Node> ();
Ptr<NetDevice> nd_sink;
NodeContainer AllNodes;
NodeContainer nodeCL1[nl1];
NetDeviceContainer ndc2;
NetDeviceContainer ndc3;
NetDeviceContainer ndc33;
NetDeviceContainer ndc4[nl1];
NetDeviceContainer ndc5[nl1*nl2];
uint16_t port = 22227;
uint32_t ipv6Tclass = 0;
bool ipv6RecvTclass = true;
uint32_t ipv6Hoplimit = 0;
bool ipv6RecvHoplimit = true;
int lmax=24;
uint16_t takhir = 10*30*NodesCount*30; //banyak pesan yang dkirim
std::string keysimpan ="";
unsigned int idxsimpan ;
std::string pubstring ="";
static uint32_t m_bytesTotal;
int tver=0;
unsigned long int nhash;
int rounds_rc5 = 12;
int tenc_rc = KR(rounds_rc5);
unsigned int lprev_node=0;
ofstream myfilerec;
int fingerprintSize = 13;
int bucketCapacity = 30;
int nrBuckets = 3;
TinyTable tt(fingerprintSize, bucketCapacity, nrBuckets);
const int jmlus=30;
SigningKey uspriv[jmlus];
VerifyingKey uspub[jmlus] ;
string usid[jmlus],upk[jmlus];
///////////////
std::string print_timediff(const char* prefix, const struct timespec& start, const struct
timespec& end)
{
    std::string aa;
    std::ostringstream strs;
    double milliseconds = (end.tv_nsec - start.tv_nsec) / 1e6 + (end.tv_sec -
        start.tv_sec) * 1e3;
    strs << milliseconds;
```

```
    printf("%s: %lf milliseconds\n", prefix, milliseconds);
    aa = strs.str() +" milliseconds";
    return aa;
}

NS_LOG_COMPONENT_DEFINE ("ExampleSixlowpan");

class MyApp : public Application
{
public:
    MyApp ();
    virtual ~MyApp ();
    void Setup ( Address address, uint32_t packetSize, uint32_t nPackets, uint16_t
        port, Ptr<NetDevice> nd, DataRate m_dr);
    void SetFill (std::string fill);
protected:
    virtual void DoDispose (void);
private:
    virtual void StartApplication (void);
    virtual void StopApplication (void);
    void ScheduleTx (void);
    void Send (void);
    void ScheduleTransmit (Time dt);
    void ReadPacket (Ptr<Socket> socket);
    uint32_t m_count;
   Time m_interval; //!< Packet inter-send time
   uint32_t m_size; //!< Size of the sent packet
   uint32_t m_dataSize; //!< packet payload size (must be equal to m_size)
   uint8_t *m_data; //!< packet payload data
   uint32_t m_sent; //!< Counter for sent packets
   Ptr<Socket> m_socket; //!< Socket
   Address m_peerAddress; //!< Remote peer address
   uint16_t m_peerPort; //!< Remote peer port
   EventId m_sendEvent; //!< Event to send the next packet
   Ptr<NetDevice> m_nd;
   DataRate m_datarate;
  TracedCallback<Ptr<const Packet> > m_txTrace;
};

MyApp::MyApp ()
{
    NS_LOG_FUNCTION (this);
    m_sent = 0;
    m_socket = 0;
    m_sendEvent = EventId ();
    m_data = 0;
    m_dataSize = 0;
}

MyApp::~MyApp ()
```

```
{
      NS_LOG_FUNCTION (this);
      m_socket = 0;
      delete [] m_data;
      m_data = 0;
      m_dataSize = 0;
}

Void MyApp::Setup ( Address address, uint32_t packetSize, uint32_t nPackets,
    uint16_t port, Ptr<NetDevice> nd, DataRate m_dr)
{
       m_peerAddress = address;
       m_peerPort = port;
       m_size = packetSize;
       m_count = nPackets;
       m_nd = nd;
       m_datarate = m_dr;
}

Void MyApp::StartApplication (void)
{
 NS_LOG_FUNCTION (this);
 if (m_socket == 0)
   {
    TypeId tid = TypeId::LookupByName ("ns3::UdpSocketFactory");
    m_socket = Socket::CreateSocket (GetNode (), tid);
    if (Ipv4Address::IsMatchingType(m_peerAddress) == true)
     {
      m_socket->Bind();
       m_socket->Connect      (InetSocketAddress(Ipv4Address::ConvertFrom(m_peerAddress),
       m_peerPort));
     }
    else
     {
      if (ipv6Tclass != 0)
         {
              m_socket->SetIpv6Tclass (ipv6Tclass);
         }
      if (ipv6Hoplimit > 0)
         {
             m_socket->SetIpv6HopLimit (ipv6Hoplimit);
         }
      m_socket->SetRecvPktInfo(true);
      m_socket->SetIpv6RecvHopLimit(true);
      m_socket->Connect
        (Inet6SocketAddress(Ipv6Address::ConvertFrom(m_peerAddress), m_peerPort));
      m_socket->BindToNetDevice(m_nd);
     }
   }
 m_socket->SetRecvCallback (MakeCallback (&MyApp::ReadPacket, this));
```

```
  ScheduleTransmit (Seconds (0.));
}

Void MyApp::StopApplication ()
{
  NS_LOG_FUNCTION (this);
  if (m_socket != 0)
    {
      m_socket->Close ();
      m_socket->SetRecvCallback (MakeNullCallback<void, Ptr<Socket> > ());
      m_socket = 0;
    }
  Simulator::Cancel (m_sendEvent);
}

Void MyApp::SetFill (std::string fill)
{
  NS_LOG_FUNCTION (this << fill);
  uint32_t dataSize = fill.size () + 1;
  if (dataSize != m_dataSize)
    {
      delete [] m_data;
      m_data = new uint8_t [dataSize];
      m_dataSize = dataSize;
    }

  memcpy (m_data, fill.c_str (), dataSize);
  m_size = dataSize;
}

Void MyApp::Send (void)
{
  NS_LOG_FUNCTION (this);
  NS_ASSERT (m_sendEvent.IsExpired ());
  Ptr<Packet> p;
  if (m_dataSize)
    {
      NS_ASSERT_MSG (m_dataSize == m_size, "MyApp::Send(): m_size and
          m_dataSize inconsistent");
      NS_ASSERT_MSG (m_data, "MyApp::Send(): m_dataSize but no m_data");
      p = Create<Packet> (m_data, m_dataSize);
    }
  else
    {
      p = Create<Packet> (m_size);
    }
  m_txTrace (p);
  m_socket->Send (p);
  ++m_sent;
  if (Ipv4Address::IsMatchingType (m_peerAddress))
```

```
  {
    NS_LOG_INFO ("At time " << Simulator::Now ().GetSeconds () << "s client
        sent " << m_size << " bytes to " << Ipv4Address::ConvertFrom
        (m_peerAddress) << " port " << m_peerPort);
  }
 else
  {
    NS_LOG_INFO ("At time " << Simulator::Now ().GetSeconds () << "s client
        sent " << m_size << " bytes to " << Ipv6Address::ConvertFrom
        (m_peerAddress) << " port " << m_peerPort);
  }
 if (m_sent < m_count)
  {
    ScheduleTransmit (m_interval);
  }
}

void
MyApp::ScheduleTransmit (Time dt)
{
 NS_LOG_FUNCTION (this << dt);
 Time tNext (Seconds (m_size * 8 / static_cast<double> (m_datarate.GetBitRate
    ())));
 std::cout<<dt<<"      TRANSMITTTT      ke      "<<Ipv6Address::ConvertFrom
    (m_peerAddress)<<" bit rate "<<m_datarate.GetBitRate ()<<std::endl;
 m_sendEvent = Simulator::Schedule (dt, &MyApp::Send, this);
}

Void MyApp::DoDispose (void)
{
 NS_LOG_FUNCTION (this);
 Application::DoDispose ();
}

Void MyApp::ReadPacket (Ptr<Socket> socket)
{
 NS_LOG_FUNCTION (this << socket);
 Ptr<Packet> packet;
 Address from;
 while ((packet = socket->RecvFrom (from)))
 {
   if (InetSocketAddress::IsMatchingType (from))
    {
      NS_LOG_INFO ("At time " << Simulator::Now ().GetSeconds () << "s client
          received " << packet->GetSize () << " bytes from " <<
          InetSocketAddress::ConvertFrom (from).GetIpv4 () << " port " <<
          InetSocketAddress::ConvertFrom (from).GetPort ());
    }
   else if (Inet6SocketAddress::IsMatchingType (from))
    {
```

```
        NS_LOG_INFO ("At time " << Simulator::Now ().GetSeconds () << "s client
              received " << packet->GetSize () << " bytes from " <<
              Inet6SocketAddress::ConvertFrom (from).GetIpv6 () << " port " <<
              Inet6SocketAddress::ConvertFrom (from).GetPort ());
      }
    }
}

void kirim_data(Ptr<Node> node, Address address, Ptr<NetDevice> nd, DataRate
      m_dr,std::string p, int t)
{
    uint32_t MaxPacketSize = 100;
    uint32_t maxPacketCount = 1;
    if(p.substr(0,8) == "00000000")
            p.substr(0,8) == "11111111";

    Ptr<MyApp> apps2 = CreateObject<MyApp> ();
    apps2->Setup (address,MaxPacketSize,maxPacketCount,port ,nd, m_dr);

    //ENCODE
    p=bin2sym(p);
    p=topem(p);
    std::cout<<"KIRIM topem p "<<p<<" panjang = "<<p.length()<<std::endl;
    apps2->SetFill(p);
    node->AddApplication (apps2);
    apps2->SetStartTime (Seconds (t));
    apps2->SetStopTime (Seconds (t+10));

}


class MyAppRec : public Application
{
public:
  MyAppRec ();
  virtual ~MyAppRec ();
  void Setup (Address address, uint16_t port, Ptr<NetDevice> nd);
  void kirim_data(Ptr<Node> node, Address address, Ptr<NetDevice> nd,   DataRate
       m_dr,std::string p, int t);

private:
  virtual void StartApplication (void);
  virtual void StopApplication (void);
  void ReadPacket (Ptr<Socket> socket);
  uint16_t m_port; //!< Port on which we listen for incoming packets.
  Ptr<Socket> m_socket; //!< IPv4 Socket
  Ptr<Socket> m_socket6; //!< IPv6 Socket
  Address m_local; //!< local multicast address
  Ptr<NetDevice> m_nd;
};
```

```
void MyAppRec::kirim_data(Ptr<Node> node, Address address, Ptr<NetDevice> nd,
    DataRate m_dr,std::string p, int t)
{
     uint32_t MaxPacketSize = 50;
     uint32_t maxPacketCount = 1;
     std::cout<<"myapprec kirim data11 "<<p<<std::endl;

     if(p.substr(0,8) == "00000000")
                        p.substr(0,8) == "11111111";
     Ptr<MyApp> apps2 = CreateObject<MyApp> ();
     apps2->Setup (address,MaxPacketSize,maxPacketCount,port ,nd, m_dr);

     //ENCODE
     p=bin2sym(p);
     p=topem(p);
     std::cout<<"KIRIM topem p "<<p<<" panjang = "<<p.length()<<std::endl;
     apps2->SetFill(p);
     node->AddApplication (apps2);
     apps2->SetStartTime (Seconds (t));
     apps2->SetStopTime (Seconds (t+10));

}


MyAppRec::MyAppRec ()
{
     NS_LOG_FUNCTION (this);
}

MyAppRec::~MyAppRec ()
{
       NS_LOG_FUNCTION (this);
       m_socket = 0;
       m_socket6 = 0;
}

Void MyAppRec::Setup (Address address, uint16_t port, Ptr<NetDevice> nd)
{
 m_local = address;
 m_port = port;
 m_nd = nd;
}

Void MyAppRec::StartApplication (void)
{
 NS_LOG_FUNCTION (this);
 if (m_socket == 0)
   {
    TypeId tid = TypeId::LookupByName ("ns3::UdpSocketFactory");
```

```
    m_socket = Socket::CreateSocket (GetNode (), tid);
    InetSocketAddress local = InetSocketAddress (Ipv4Address::GetAny (), m_port);
    m_socket->Bind (local);
    if (addressUtils::IsMulticast (m_local))
      {
        Ptr<UdpSocket> udpSocket = DynamicCast<UdpSocket> (m_socket);
        if (udpSocket)
          {
            udpSocket->MulticastJoinGroup (0, m_local);
          }
        else
          {
            NS_FATAL_ERROR ("Error: Failed to join multicast group");
          }
      }
  }

 if (m_socket6 == 0)
   {
    TypeId tid = TypeId::LookupByName ("ns3::UdpSocketFactory");
    m_socket6 = Socket::CreateSocket (GetNode (), tid);
    Inet6SocketAddress local6 = Inet6SocketAddress (Ipv6Address::GetAny (), port);
           // unicast
    m_socket6->Bind (local6);
    m_socket6->BindToNetDevice(m_nd);
    if (addressUtils::IsMulticast (local6))
      {
            Ptr<UdpSocket> udpSocket = DynamicCast<UdpSocket> (m_socket6);
            if (udpSocket)
                udpSocket->MulticastJoinGroup (0, local6);
            else
                NS_FATAL_ERROR ("Error: Failed to join multicast group");
      }
   }
 m_socket->SetRecvCallback (MakeCallback (&MyAppRec::ReadPacket, this));
 m_socket6->SetRecvCallback (MakeCallback (&MyAppRec::ReadPacket, this));
}

Void MyAppRec::StopApplication ()
{
 NS_LOG_FUNCTION (this);
 if (m_socket != 0)
   {
    m_socket->Close ();
    m_socket->SetRecvCallback (MakeNullCallback<void, Ptr<Socket> > ());
   }
 if (m_socket6 != 0)
   {
     m_socket6->Close ();
     m_socket6->SetRecvCallback (MakeNullCallback<void, Ptr<Socket> > ());
```

```
  }
}

Void MyAppRec::ReadPacket (Ptr<Socket> socket)
{
 NS_LOG_FUNCTION (this << socket);
 SystemWallClockMs bb;
 Ptr<Packet> packet;
 Address from;
 int diteruskan=0;
 pesanke = pesanke +1;
 bb.Start();
 while ((packet = socket->RecvFrom (from)))
  {
      std::cout << "Pesan diterima ke " << pesanke << " dari "<<
        Inet6SocketAddress::ConvertFrom (from).GetIpv6 () << " ke " <<
        Inet6SocketAddress::ConvertFrom (m_local).GetIpv6 ()<<std::endl;
      DataRate m_dr("250kbps");
      unsigned int l = 0;
      uint8_t *buffer = new uint8_t[packet->GetSize ()];
      memset(buffer, 0, packet->GetSize () );
      packet->CopyData(buffer, packet->GetSize ());
      std::string  isi = std::string((char*)buffer);
      std::string  isiN;
      if(lprev_node == 0)
      lprev_node=lmax;

       //DECODE
      isi = unpem(isi);
      isi = symtobin(isi);
      std::cout << "isi = "<< isi  << "ukuran "<< isi.length() << std::endl;
      isiN = isi;
      m_bytesTotal = m_bytesTotal + packet->GetSize ();
if(isi.length() == 384)
        {   pubstring = isi.substr(64,320);
            keysimpan = isi.substr(0,64);
            idxsimpan = 0;
          std::cout << "pub = "<< unhexlify(bin2hex(pubstring)) << std::endl;
        std::cout << "seskeykey_bin = "<< keysimpan << std::endl;
        std::cout << "seskeyhex = "<< bin2hex(keysimpan) << std::endl;
      }
     else if(isi.length() >= 632 and isi.length() <= 784)
     {
          clock_gettime(CLOCK_MONOTONIC, &start);
          clock_gettime(CLOCK_MONOTONIC, &start2);
          unsigned int  temptag;
          int tagvalid=0;
          std::string vtag;
          std::cout << "Proses verifikasi multiuser " <<  std::endl;
          l=lmax;
```

```
lprev_node = lmax;
//1. VERIFIKASI TAG
clock_gettime(CLOCK_MONOTONIC, &start3);
while((l>0) and (tagvalid == 0))
{
    temptag=(idxsimpan+1)+lprev_node+l;
    vtag=std::bitset< 8 >(temptag).to_string();
    vtag=hextobin(sha1(vtag));
    vtag=vtag.substr(0,24);
    if(vtag == isi.substr(isi.length()-24,24))
    {
        tagvalid=1;
    }
    else
    {
        l=l-1;
    }
 }
 clock_gettime(CLOCK_MONOTONIC, &finish3);
 double elapsed3;
 elapsed3 = (finish3.tv_sec - start3.tv_sec);
 elapsed3 += (finish3.tv_nsec - start3.tv_nsec) / 1000000000.0 ;
 if(tagvalid == 1)
 {   clock_gettime(CLOCK_MONOTONIC, &finish);
     std::cout<<"Tag valid"<<std::endl;
     //2. VERIFIKASI TINY TABLE
     std::string tempupk;
     tempupk = isi.substr(488,16)+isi.substr(320,168);
     tempupk = bin2sym(tempupk);
     if((tt.containItem(tempupk) == true) or (tt.containItem(tempupk) ==
             false))
     {

     if(tt.containItem(tempupk) == true)
     std::cout<<"bismillah user aktif TT OK"<<std::endl;
     else
     {
         std::cout<<"publick key not valid"<<std::endl;
         myfilerec <<pesanke<<" not valid"<<"\n";
     }
    std::string encpesan,puzpesan = isi.substr(504,isi.length()-528);
puzpesan = hextobin(md5(bin2sym(puzpesan)));
    encpesan = isi.substr(504,isi.length()-528);
    std::cout<<bin2sym(isi.substr(0,320)).length()<<"sig="<<
       isi.substr(0,320) <<std::endl;
    std::string sign_rec=realtopub(bin2sym(isi.substr(0,320)));
    std::string hsig;
    hsig = sha1(sign_rec);
    //3. VERIFIKASI PUZZLE SOLUTION
    if((hextobin(hsig)).substr(0,l)==puzpesan.substr(0,l))
```

```
{
   std::cout<<"Puzzle valid= "<<std::endl;
   // 4. VERIFIKASI SESSION KEY
   BYTE Key_rec[8] ;
   word32 *ky1;
   std::string ksym,hdek;
   ksym = bin2sym(keysimpan);
   copy( ksym.begin(), ksym.end(), Key_rec );
   Key_rec[ksym.length()] = 0;
   BYTE Key_rec2[8] ;
   ky1 = (word32 *) calloc(tenc_rc, sizeof(word32));
   copy( ksym.begin(), ksym.end(), Key_rec2 );
   Key_rec2[ksym.length()] = 0;
   RC5key(Key_rec, sizeof(Key_rec), ky1);
  hdek=decRC5(bin2sym(encpesan),ky1);
 std::string m2 = hdek.substr(0,hdek.length()-12);
 std::string psol = hdek.substr(hdek.length()-4,4);
 std::string sesk = hdek.substr(hdek.length()-12,8); // session key
 if(hextobin(sha1(sesk).substr(0,16))==keysimpan)
 {
     std::cout<<"session key valid GOOD FILTER"<<std::endl;
     VerifyingKey pub;
     curve cc1;
     pubstring = bin2sym(isi.substr(320,168));
     pubstring = bin2sym(hextobin(dekomprespubk(pubstring)));
     std::string m1;

         sign_rec=realtopub(bin2sym(isi.substr(0,320)));
         try
         {
         pub =pub.from_string(pubstring,cc1,"", true);
         if(pub.verify_modified(sign_rec,m2+usid[idxsimpan],"",&m1)
              == true)
           {
              std::cout << "GOOD SIGNATURE" << std::endl;
              clock_gettime(CLOCK_MONOTONIC, &finish2);
              keysimpan = symtobin(sesk);
              idxsimpan++;
              lprev_node=l; // jika tagging valid
           }
         }catch (std::runtime_error e)
         {
           std::cout << "BAD SIGNATURE" <<std::endl;
         }
 } //endif keychain
 else
 {
     std::cout<<"session key TIDAK valid "<<std::endl;
 }
} //endif puzzle
```

```
                else
                {
                    std::cout<<"Puzzle not VERIFIED"<<std::endl;
                }
            } //tiny table benar
            else
            {
                std::cout<<"Public key not VERIFIED"<<std::endl;
            } //tiny table salah
        } //if tagvalid ==1
        else
        {
            clock_gettime(CLOCK_MONOTONIC, &finish);
            std::cout << "TAGGING TIDAK VALID" <<  std::endl;
        }
        bb.End();
        clock_gettime(CLOCK_MONOTONIC, &finish2);
        elapsed = (finish.tv_sec - start.tv_sec);
        elapsed += (finish.tv_nsec - start.tv_nsec) / 1000000000.0 ; //second
        std::cout<<" elapsed for tagging = "<<(1000*elapsed)<<std::endl;
        elapsed2 = (finish2.tv_sec - start2.tv_sec);
        elapsed2 += (finish2.tv_nsec - start2.tv_nsec) / 1000000000.0 ;
        std::cout<<" elapsed TOTAL = "<<(1000*elapsed2)<<std::endl;
 }
 else //end else pesan tidak sama dengan 752
 {
    std::cout << "PESAN KORUP" <<  std::endl;
 }
 std::cout<<"waktu sistem = "<< bb.GetElapsedSystem() <<std::endl;
 std::cout<<"waktu real = "<< bb.GetElapsedReal() <<std::endl;
 std::cout<<"waktu user = "<< bb.GetElapsedUser() <<std::endl;
 tver = tver +  (1000*elapsed2);
 std::cout<<"waktu SEMUA user = "<< tver <<" ms"<<std::endl;
 myfilerec<<pesanke<<";;"<<(1000*elapsed2)<<";;"<<tver<<";"<<(1000*elapse
       d)<<"\n";
 if (InetSocketAddress::IsMatchingType (from))
 {
    NS_LOG_INFO ("At time " << Simulator::Now ().GetSeconds () << "s
        server received " << packet->GetSize () << " bytes from " <<
        InetSocketAddress::ConvertFrom (from).GetIpv4 () << " port "
        <<InetSocketAddress::ConvertFrom (from).GetPort ());
}
else if (Inet6SocketAddress::IsMatchingType (from))
{
    NS_LOG_INFO ("At time " << Simulator::Now ().GetSeconds () << "s
        server received " << packet->GetSize () << " bytes from "
        <<Inet6SocketAddress::ConvertFrom (from).GetIpv6 () << " port " <<
        Inet6SocketAddress::ConvertFrom (from).GetPort ());
}
packet->RemoveAllPacketTags ();
```

```
packet->RemoveAllByteTags ();
NS_LOG_LOGIC ("Echoing packet");
//  socket->SendTo (packet, 0, from);
if (diteruskan == 1) //2001:2::200:ff:fe00:5
{
 if (m_nd == ndc2.Get(1))
      kirim_data(sink, Ipv6Address ("FF02::02"), ndc3.Get(0), m_dr,isiN,
      Simulator::Now ().GetSeconds ());
 else if(Inet6SocketAddress::ConvertFrom (m_local).GetIpv6 () == Ipv6Address
 ("2001:4::200:ff:fe00:a"))
      kirim_data(nodeCL1[0].Get(0), Ipv6Address ("FF02::02"), ndc4[0].Get(0),
      m_dr,isiN, Simulator::Now ().GetSeconds ());
 else if(Inet6SocketAddress::ConvertFrom (m_local).GetIpv6 () == Ipv6Address
 ("2001:4::200:ff:fe00:d"))
      kirim_data(nodeCL1[1].Get(0), Ipv6Address ("FF02::02"), ndc4[1].Get(0),
      m_dr,isiN, Simulator::Now ().GetSeconds ());
 else if(Inet6SocketAddress::ConvertFrom (m_local).GetIpv6 () == Ipv6Address
 ("2001:4::200:ff:fe00:10"))
      kirim_data(nodeCL1[2].Get(0), Ipv6Address ("FF02::02"), ndc4[2].Get(0),
      m_dr,isiN, Simulator::Now ().GetSeconds ());
 else if(Inet6SocketAddress::ConvertFrom (m_local).GetIpv6 () == Ipv6Address
 ("2001:5::200:ff:fe00:13"))
      kirim_data(nodeCL1[0].Get(1), Ipv6Address ("FF02::02"), ndc5[0].Get(0),
      m_dr,isiN, Simulator::Now ().GetSeconds ());
 else if(Inet6SocketAddress::ConvertFrom (m_local).GetIpv6 () == Ipv6Address
 ("2001:5::200:ff:fe00:16"))
      kirim_data(nodeCL1[0].Get(2), Ipv6Address ("FF02::02"), ndc5[1].Get(0),
      m_dr,isiN, Simulator::Now ().GetSeconds ());
 else if(Inet6SocketAddress::ConvertFrom (m_local).GetIpv6 () == Ipv6Address
 ("2001:5::200:ff:fe00:19"))
      kirim_data(nodeCL1[1].Get(1), Ipv6Address ("FF02::02"), ndc5[2].Get(0),
      m_dr,isiN, Simulator::Now ().GetSeconds ());
 else if(Inet6SocketAddress::ConvertFrom (m_local).GetIpv6 () == Ipv6Address
 ("2001:5::200:ff:fe00:1c"))
      kirim_data(nodeCL1[1].Get(2), Ipv6Address ("FF02::02"), ndc5[3].Get(0),
      m_dr,isiN, Simulator::Now ().GetSeconds ());
 else if(Inet6SocketAddress::ConvertFrom (m_local).GetIpv6 () == Ipv6Address
 ("2001:5::200:ff:fe00:1f"))
      kirim_data(nodeCL1[2].Get(1), Ipv6Address ("FF02::02"), ndc5[4].Get(0),
      m_dr,isiN, Simulator::Now ().GetSeconds ());
 else if(Inet6SocketAddress::ConvertFrom (m_local).GetIpv6 () == Ipv6Address
 ("2001:5::200:ff:fe00:22"))
      kirim_data(nodeCL1[2].Get(2), Ipv6Address ("FF02::02"), ndc5[5].Get(0),
      m_dr,isiN, Simulator::Now ().GetSeconds ());
}
if (InetSocketAddress::IsMatchingType (from))
{
    NS_LOG_INFO ("At time " << Simulator::Now ().GetSeconds () << "s server
    sent " << packet->GetSize () << " bytes to " <<
```

```
            InetSocketAddress::ConvertFrom (from).GetIpv4 () << " port " <<
            InetSocketAddress::ConvertFrom (from).GetPort ());
        }
        else if (Inet6SocketAddress::IsMatchingType (from))
        {
            NS_LOG_INFO ("At time " << Simulator::Now ().GetSeconds () << "s server
            sent " << packet->GetSize () << " bytes to " <<
            Inet6SocketAddress::ConvertFrom (from).GetIpv6 () << " port " <<
            Inet6SocketAddress::ConvertFrom (from).GetPort ());
    }
   } //endwhile
}

std::string msp (std::string fp)
{
    std::string phash="";
    int benar =0;
    std::string p="",rnd;
    unsigned int l = 0;
    l=22;
    std::string lbin;
    float maxnhash;
    do
    {
        maxnhash=0.7448*exp(0.6994*l); //new3
        std::cout<<"l saat ini = "<<l<<std::endl;
        std::cout<<"maxnhash = "<<maxnhash<<std::endl;
        nhash = 0;
        lbin = genArray0(l);
        while((benar == 0) && (nhash <= maxnhash))
        {
            nhash++;
            rnd=genRandom(8);
            p = fp+hextobin(rnd); //32 bit = 8 hex x 4
            phash = hextobin(sha1(bin2sym(p)));
            if(phash.substr(0,l)==lbin)
            benar = 1;
        }
        if (benar == 0)
            l=l-1;
    } while ((l>0) and (benar == 0));
    if((benar == 1) && (nhash <= maxnhash))
        return p;
    else
        return "";
}

std::string cp (std::string h1, std::string sig, std::string k0, std::string k1,word32 *ky,
unsigned int *l) //semua format symbol
{
```

```
word32 *ky;
BYTE Key[8] ;
BYTE digest[20] ;
std::string sran=""; //random 32 bit = 8hex x 4;

int benar=0;
std::string data,hsig,binhasil2;
std::string hasil,hasil2;
CHMAC_SHA1 HMAC_SHA1 ;
hsig = bin2sym(hextobin(sha1(sig))); //output sha1=hexa
float maxnhash;
copy( k0.begin(), k0.end(), Key );
Key[k0.length()] = 0;
do
{
    //Q2 or median
    //maxnhash=pow(2,l); //new1
    //maxnhash=ceil(1.093e-19*pow(l,19.05)); //CPnew2salah data30
    //maxnhash=ceil(7.754e-09*pow(l,10.78)); //CPnew2
    //maxnhash=ceil(0.01047*exp(0.8989*l)); //CPnew3salah data30
    //maxnhash= ceil(20.86*exp(0.5282*l)); //CPnew3
    //**********************
    //Q1
    maxnhash =  ceil(pow(1.8786,double(*l))+1726); //Q1new1
    //maxnhash= ceil(2.412e-11*pow(double(*l),12.4)); //Q1new2
    //maxnhash= ceil(1.966*exp(0.5997*double(*l))); //Q1new3
    //**********************
    //Q3
    //maxnhash= ceil(pow(2.018,l)+(6.448e+04)); //Q3new1
    //maxnhash = ceil(3.147e-13*pow(l,14.35)); //Q3new2
    //maxnhash = ceil(1.533 *exp(0.688*l)); //Q3new3
    std::cout << "maxnhash " << maxnhash<<std::endl;
    nhash = 0;
    //std::cout << "h1 " << h1<<" panjang "<<h1.length()<<std::endl;
    //std::cout<<"l CP saat ini = "<<l<<std::endl;
    //std::cout<<"maxnhash = "<<maxnhash<<std::endl;
    while((benar == 0) && (nhash <= maxnhash))
    {
        nhash++;
        sran= bin2sym(hextobin(genRandom(8)));
        data=h1+sran+k1; //format symbol
        hasil=encRC5(data,ky);
        unsigned char datauc [] = "";
        copy( hasil.begin(), hasil.end(), datauc );
        datauc[hasil.length()] = 0;
        HMAC_SHA1.HMAC_SHA1(datauc, hasil.length(), Key, sizeof(Key), digest)
        ;
        std::string hmac(reinterpret_cast<char*>(digest));

        hmac=hmac.substr(0,20); //format symbol
```

```
            hasil2=hmac+hasil;
            binhasil2=(symtobin(hasil2)).substr(0,*l);
            if((symtobin(hsig)).substr(0,*l)==binhasil2)
                benar=1;
            else
                sran="";
        } //end while puzzle
        if (benar == 0)
            *l=*l-1;
            std::cout << " l turun " <<std::endl;
    } while ((*l>0) and (benar == 0)); //end do
    if((benar == 1) && (nhash <= maxnhash))
        return (hasil2);
    else
        return "";
}

std::string cptt (std::string h1, std::string sig, std::string k0, std::string k1,word32 *ky,
unsigned int *l) //semua format symbol
{
    BYTE Key[8] ;
    std::string sran=""; //random 32 bit = 8hex x 4;
    std::cout<<"signature utk puzzle= "<<symtobin(sig)<<std::endl;
    int benar=0;
    std::string data,hsig,binhasil2;
    std::string hasil,hasil2;
    int ltemp = lmax;
    CHMAC_SHA1 HMAC_SHA1 ;
    hsig = bin2sym(hextobin(sha1(sig))); //output sha1=hexa
    std::cout<<sig<<" binari HSIG = "<<symtobin(hsig)<<std::endl;
    float maxnhash;
    copy( k0.begin(), k0.end(), Key );
    Key[k0.length()] = 0;
    std::cout <<h1.length()<< " panjang pesan= " << h1<<std::endl;
    std::cout <<k1.length()<< " panjang kunci k1= " << k1<<std::endl;
    do
    {
        //Q2 or median
        //maxnhash=pow(2,l); //new1
        //maxnhash=ceil(1.093e-19*pow(l,19.05)); //CPnew2salah data30
        //maxnhash=ceil(7.754e-09*pow(l,10.78)); //CPnew2
        //maxnhash=ceil(0.01047*exp(0.8989*l)); //CPnew3salah data30
        //maxnhash= ceil(20.86*exp(0.5282*l)); //CPnew3
        //**********************
        //Q1
        //maxnhash =  ceil(pow(1.8786,double(*l))+1726); //Q1new1
        //maxnhash= ceil(2.412e-11*pow(double(*l),12.4)); //Q1new2
        //maxnhash= ceil(1.966*exp(0.5997*double(*l))); //Q1new3
        //**********************
        //Q3
```

```
        //maxnhash= ceil(pow(2.018,l)+(6.448e+04)); //Q3new1
        //maxnhash = ceil(3.147e-13*pow(l,14.35)); //Q3new2
        //maxnhash = ceil(1.533 *exp(0.688*l)); //Q3new3
        //std::cout<<" lmax = "<<lmax<<std::endl;
        //std::cout<<"l = "<<*l<<std::endl;
        //int temp=(0.9999999/(pow(lmax,2))*(pow(( (*l) - (lmax+1) ),2)));
        //DETER
        //maxnhash=  ceil(log(1-(0.999999*(l-(lmaks+1))/-(lmaks)))/log(1-(pow(0.5,l))));
              //garis 24
        maxnhash=ceil(log10(1-(0.9999999/(pow(lmax,2))*(pow((ltemp-
              (lmax+1)),2))))/log10(1-(1/pow(2,ltemp)))); //kurva24
        std::cout << "maxnhash " << maxnhash<<std::endl;
        nhash = 0;
        while((benar == 0) && (nhash <= maxnhash))
        {
            nhash++;
            //std::cout <<maxnhash<< " nhash " << nhash<<std::endl;
            sran= bin2sym(hextobin(genRandom(8)));
            data=h1+k1+sran; //format symbol
            hasil=encRC5(data,ky);
            binhasil2 = hextobin(md5(hasil)); //output sha1=hexa
            if((symtobin(hsig)).substr(0,ltemp)==binhasil2.substr(0,ltemp))
                benar=1;
            else
                sran="";
        } //end while puzzle
        if (benar == 0)
            ltemp=ltemp-1;
        std::cout << " l turun " <<std::endl;
    } while ((ltemp>0) and (benar == 0)); //end do
    *l = ltemp;
    std::cout << "hdek= " << symtobin(data)<<std::endl;
    std::cout <<hasil.length()<< " encrypted pesan= "<<symtobin(hasil) <<std::endl; //
    std::cout          <<(symtobin(hsig)).substr(0,*l).length()<<          "          puzzlep=
        "<<(symtobin(hsig)).substr(0,*l) <<std::endl;
    if((benar == 1) && (nhash <= maxnhash))
        return (hasil);
    else
        return "";
}

std::string randommsp ()
{
    std::string phash="";
    int benar =0;
    std::string p="";
    unsigned int l = 0;
    std::string lbin = genArray0(l);
    while(benar == 0)
    {
```

```
            p = hextobin(genRandom(204)); //32 bit = 8 hex x 4
            phash = hextobin(sha1(bin2sym(p)));
            if(phash.substr(0,l)==lbin)
                benar = 1;
        }
        return p;
}

int main (int argc, char** argv)
{
        bool verbose = false;
        CommandLine cmd;
        cmd.AddValue ("verbose", "turn on some relevant log components", verbose);
        cmd.Parse (argc, argv);

        // Enable logging for UdpClient and UdpServer
        LogComponentEnable ("UdpClient", LOG_LEVEL_INFO);
        LogComponentEnable ("UdpServer", LOG_LEVEL_INFO);
        Packet::EnablePrinting ();
        Packet::EnableChecking ();

        NS_LOG_INFO ("Create nodes.");
        Ptr<Node> sta1 = CreateObject<Node> ();
        Ptr<Node> r1 = CreateObject<Node> ();
        Ptr<Node> r2 = CreateObject<Node> ();
        Ptr<Node> sta2 = CreateObject<Node> ();
        sink = sta2;
        NodeContainer net1 (sta1, r1, r2);
        NodeContainer net2 (r2, sta2);
        AllNodes.Create(nl1);
        NodeContainer sinksensor = NodeContainer (sta2, AllNodes);
        NodeContainer all (sta1, r1, r2, sta2, AllNodes);

        AnimationInterface::SetConstantPosition (sta1, 10, 30);
        AnimationInterface::SetConstantPosition (r1, 30, 35);
        AnimationInterface::SetConstantPosition (r2, 10, 45);
        AnimationInterface::SetConstantPosition (sta2, 10, 50);
        for(int i=0;i<nl1;i++) //NC menghubungkan level 1 dan 2
        {
            nodeCL1[i].Add(AllNodes.Get(i));
            nodeCL1[i].Create(nl2);
        }

        NodeContainer nodeCL2[nl1*nl2];
        int k=0;
        for(int i=0;i<nl1;i++) //NC menghubungkan level 1 dan 2
        {
            AnimationInterface::SetConstantPosition (AllNodes.Get(i), 10+(5*i), 55);
            for(int j=1;j<=nl2;j++) //NC menghubungkan level 2 dan 3
            {//j=jumlah nodeL2 = nl2
```

```
            AnimationInterface::SetConstantPosition  (nodeCL1[i].Get(j),  10+(6*i)+(j*4),
                65); //nl2
            nodeCL2[k].Add(nodeCL1[i].Get(j));//get(0)diisi oleh interface allNode
            nodeCL2[k].Create(nl3);
            all.Add(nodeCL1[i].Get(j));
            k++;
        }
    }

    k=0;
    for(int i=0;i<(nl1*nl2);i++) //NC menghubungkan level 1 dan 2
    { //i=jumlah nodeCL1 = nl1
        for(int j=1;j<=nl3;j++) //NC menghubungkan level 2 dan 3
        {//j=jumlah nodeL2 = nl2
            AnimationInterface::SetConstantPosition  (nodeCL2[i].Get(j),  10+(6*i)+(4*j),
                75); //nl3
            all.Add(nodeCL2[i].Get(j));
            k++;
        }
    }

    InternetStackHelper internetv6;
    internetv6.SetIpv4StackInstall (false);
    internetv6.Install (all);

    NS_LOG_INFO ("Create channels.");
    CsmaHelper csma;
    csma.SetChannelAttribute ("DataRate", DataRateValue (5000000));
    csma.SetChannelAttribute ("Delay", TimeValue (MilliSeconds (2)));
    // csma.SetDeviceAttribute ("Mtu", UintegerValue (150)); paket masimal 648
    NetDeviceContainer ndc11 = csma.Install (net1);
    NetDeviceContainer ndc22 = csma.Install (net2);
    ndc33 = csma.Install (sinksensor);
    NetDeviceContainer ndc44[nl1];
    for(int i=0;i<nl1;i++) //NC menghubungkan level 1 dan 2
    {
        ndc44[i]= csma.Install(nodeCL1[i]);
    }

    NetDeviceContainer ndc55[nl1*nl2];
    for(int i=0;i<(nl1*nl2);i++) //NC menghubungkan level 1 dan 2
    {
        ndc55[i]= csma.Install(nodeCL2[i]);
    }

    SixLowPanHelper sixlow;
    sixlow.SetDeviceAttribute ("ForceEtherType", BooleanValue (true) );
    NetDeviceContainer ndc1 = sixlow.Install (ndc11); //sta1/user-router
    ndc2 = sixlow.Install (ndc22);
    ndc3 = sixlow.Install (ndc33);
```

```
 nd_sink = ndc3.Get(0);

 // ndc4[nl1] = global variable
 for(int i=0;i<nl1;i++) //NC menghubungkan level 1 dan 2 allnode
 {
        ndc4[i]= sixlow.Install(ndc44[i]);
 }

 // ndc5[nl1*nl2] = global variable
for(int i=0;i<(nl1*nl2);i++) //NC menghubungkan level 1 dan 2
{
        ndc5[i]= sixlow.Install(ndc55[i]);
}

 NS_LOG_INFO ("Assign IPv6 Addresses.");
 Ipv6AddressHelper ipv6;
 ipv6.SetBase (Ipv6Address ("2001:1::"), Ipv6Prefix (64));
 Ipv6InterfaceContainer iic1 = ipv6.Assign (ndc1);
 iic1.SetForwarding (2, true);
 iic1.SetDefaultRouteInAllNodes (2);
 ipv6.SetBase (Ipv6Address ("2001:2::"), Ipv6Prefix (64));
 Ipv6InterfaceContainer iic2 = ipv6.Assign (ndc2);
 iic2.SetForwarding (1, true);
 iic2.SetForwarding (0, true);
 iic2.SetDefaultRouteInAllNodes (0);
 ipv6.SetBase (Ipv6Address ("2001:3::"), Ipv6Prefix (64));
 Ipv6InterfaceContainer iic3 = ipv6.Assign (ndc3);
 iic3.SetForwarding (0, true);
 iic3.SetDefaultRouteInAllNodes (0);
 ipv6.SetBase (Ipv6Address ("2001:4::"), Ipv6Prefix (64));
 Ipv6InterfaceContainer iic4[nl1];
 for(int i=0;i<nl1;i++) //NC menghubungkan level 1 dan 2
 {
        iic4[i] = ipv6.Assign (ndc4[i]);
        iic4[i].SetForwarding (0, true);
        iic4[i].SetDefaultRouteInAllNodes (0);
 }

 ipv6.SetBase (Ipv6Address ("2001:5::"), Ipv6Prefix (64));
 Ipv6InterfaceContainer iic5[nl1*nl2];
 for(int i=0;i<(nl1*nl2);i++) //NC menghubungkan level 2 dan 3
 {
    iic5[i] = ipv6.Assign (ndc5[i]);
    iic5[i].SetForwarding (0, true);
    iic5[i].SetDefaultRouteInAllNodes (0);
 }


 Ipv6StaticRoutingHelper routingHelper;
```

```
// manually inject a static route to the second router.
Ptr<Ipv6StaticRouting> routing2 = routingHelper.GetStaticRouting (r2->GetObject<Ipv6> ());
routing2->AddNetworkRouteTo(Ipv6Address ("2001:3::"), Ipv6Prefix (64), iic2.GetAddress (1, 1), iic3.GetInterfaceIndex (0));
routing2->AddNetworkRouteTo(Ipv6Address ("2001:4::"), Ipv6Prefix (64), iic2.GetAddress (1, 1), iic3.GetInterfaceIndex (0));
routing2->AddNetworkRouteTo(Ipv6Address ("2001:5::"), Ipv6Prefix (64), iic2.GetAddress (1, 1), iic3.GetInterfaceIndex (0));

Ptr<OutputStreamWrapper> routingStream = Create<OutputStreamWrapper> (&std::cout);
routingHelper.PrintRoutingTableAt (Seconds (2.0), r1, routingStream);
routingHelper.PrintRoutingTableAt (Seconds (90.0), sta1, routingStream);
routingHelper.PrintRoutingTableAt (Seconds (90.0), r2, routingStream);

// 1. TERIMA Create one udpServer applications on node one.
Ptr<MyAppRec> app = CreateObject<MyAppRec> ();

app->Setup(Inet6SocketAddress (iic1.GetAddress (0, 1),port),port,ndc1.Get(0));
    ///local , port, netdevice local
sta1->AddApplication (app); //2001:1::200:ff:fe00:1
app->SetStartTime (Seconds (0.0));
app->SetStopTime (Seconds (takhir));

Ptr<MyAppRec> appa = CreateObject<MyAppRec> ();
appa->Setup(Inet6SocketAddress (iic1.GetAddress (1, 1),port),port,ndc1.Get(1));
    ///local , port, netdevice local
r1->AddApplication (appa); //2001:1::200:ff:fe00:2
appa->SetStartTime (Seconds (0.0));
appa->SetStopTime (Seconds (takhir));

Ptr<MyAppRec> appb = CreateObject<MyAppRec> ();
appb->Setup(Inet6SocketAddress (iic1.GetAddress (2, 1),port),port,ndc1.Get(2));
    ///local , port, netdevice local
r2->AddApplication (appb); //2001:1::200:ff:fe00:3
appb->SetStartTime (Seconds (0.0));
appb->SetStopTime (Seconds (takhir));

Ptr<MyAppRec> appc = CreateObject<MyAppRec> ();
appc->Setup(Inet6SocketAddress (iic2.GetAddress (0, 1),port),port,ndc2.Get(0));
    ///local , port, netdevice local
r2->AddApplication (appc); //2001:2::200:ff:fe00:4
appc->SetStartTime (Seconds (0.0));
appc->SetStopTime (Seconds (takhir+100));

Ptr<MyAppRec> appz = CreateObject<MyAppRec> ();
appz->Setup(Inet6SocketAddress (iic2.GetAddress (1, 1),port),port,ndc2.Get(1));
    ///local , port, netdevice local
sta2->AddApplication (appz);
```

```
appz->SetStartTime (Seconds (0.0));
appz->SetStopTime (Seconds (takhir+100));

for (uint16_t i = 0; i < (nl1*nl2); i++)
{
    for (uint16_t j = 0; j <= nl3; j++)
    {
        Ptr<MyAppRec> app2 = CreateObject<MyAppRec> ();
        app2->Setup(Inet6SocketAddress (iic5[i].GetAddress (j,
            1),port),port,ndc5[i].Get(j)); ///local , port, netdevice local
        nodeCL2[i].Get(j)->AddApplication (app2);
        app2->SetStartTime (Seconds (0.0));
        app2->SetStopTime (Seconds (takhir+100));
    }
}
for (uint16_t i = 0; i < nl1; i++)
{
    for (uint16_t j = 0; j <= nl2; j++)
    {
        Ptr<MyAppRec> app2 = CreateObject<MyAppRec> ();
        app2->Setup(Inet6SocketAddress (iic4[i].GetAddress (j,
            1),port),port,ndc4[i].Get(j)); ///local , port, netdevice local
        nodeCL1[i].Get(j)->AddApplication (app2);
        app2->SetStartTime (Seconds (0.0));
        app2->SetStopTime (Seconds (takhir+100));
    }
}

for (uint16_t i = 0; i <= nl1; i++)
{
    Ptr<MyAppRec> app2 = CreateObject<MyAppRec> ();
    app2->Setup(Inet6SocketAddress (iic3.GetAddress (i, 1),port),port,ndc3.Get(i));
        ///local , port, netdevice local
    if(i > 0)
        AllNodes.Get(i-1)->AddApplication (app2);
    else
        sta2->AddApplication (app2);
    app2->SetStartTime (Seconds (0.0));
    app2->SetStopTime (Seconds (takhir+100));
}

myfilerec.open ("catwaktunuserTTCPrec.txt");

// 1. KIRIM Create one UdpClient application to send UDP datagrams from sta1 to
    AllNode

DataRate m_dr("250kbps");
int t=4*NodesCount;
int tkirim;
// Generate Key Chain
```

```
int panjangkey = 710; //banyaknya key pada keychain
std::string* key;
key = genKeyChain(panjangkey); //output key (hexa)
std::string pubreal,keysym=bin2sym(hextobin(key[0]));

SigningKey priv;
curve cc;
priv.generate(cc,"");

VerifyingKey pub,pub2 ;
pub = priv.get_verifying_key();
pubreal = pubtoreal(pub.to_string());
std::string keykirim = keysym+pubreal,komppub;
keykirim=symtobin(keykirim);
ofstream myfile;
myfile.open ("catwaktunuserTTCP.txt");
clock_gettime(CLOCK_MONOTONIC, &start2);
int j=0,ntry=0;
while (j<jmlus)
{  ntry++;
    uspriv[j].generate(cc,"");
    uspub[j] = uspriv[j].get_verifying_key();
    if(bin2sym(hextobin(dekomprespubk(komppub))) == uspub[j].to_string())
    {
        std::cout<<j<<"                    kompresi                    berhasil
            "<<bintohex(symtobin(uspub[j].to_string()))<<std::endl;
        clock_gettime(CLOCK_MONOTONIC, &start2);
        usid[j] = bin2sym(hextobin(genRandom(4))); //random 16 bit = 8hex x 4
            [output hexa]
        upk[j] = usid[j]+ komppub;
        tt.addItem(upk[j]);
        std::cout<<"user "<<j<<" is inserted, panjang= "<<upk[j].length()<<std::endl;
        j++;
        clock_gettime(CLOCK_MONOTONIC, &finish2);
        myfile <<"add TT = "<<print_timediff("chrono::system_clock::now", start2,
            finish2)<<"\n";
    }
    else
        std::cout<<j<<"Trying... "<<std::endl;
}
clock_gettime(CLOCK_MONOTONIC, &finish2);
std::cout<<"isi Tinytable = "<<(tt.getNrItems())<<std::endl;
myfile <<"fingerprintSize = "<<fingerprintSize<<"\n";
myfile <<"bucketCapacity = "<<bucketCapacity<<"\n";
myfile <<"nrBuckets = "<<nrBuckets<<"\n";
myfile <<"lmaks = "<<lmax<<"\n";
myfile <<"add TinyTable = "<<print_timediff("chrono::system_clock::now", start2,
    finish2)<<"\n";
myfile <<"isi Tinytable = "<<(tt.getNrItems())<<" dengan mencoba sebanyak=
    "<<ntry<<"\n";
```

```
kirim_data(sta1, Ipv6Address (iic2.GetAddress (1, 1)), ndc1.Get(0), m_dr,keykirim,
    5);

// 2. KIRIM Create UdpClient application to send UDP from sta1 to AllNode
unsigned int l = 0,lprev=lmax,lpesan=0; //,lmax=15
std::string mrand; //random 16 bit = 4hex x 4;
std::string data="bismillah saya bisa"; //maks = 30 recovery share send add makeno
std::string hasil,hasil2;
std::string enckey_hex = bin2sym(hextobin(key[0]));
int index = 0;
std::string idx, kidx,sig, M,tm2, MSig, p="";
std::string Sig,pesan,pot1,pot2,sigpesan,m2,puzpesan,tagpesan,kpesan;
myfile <<"L = "<<l<< " Writing this to a file. pjg pesan "<<data.length()<<". isi
   ="<<data<<"\n";
word32 *ky;
std::string k0;
BYTE enckey[8] ;
ky = (word32 *) calloc(tenc_rc, sizeof(word32));
index++; // commitment value idx = 0,
idx = std::bitset<16>(index).to_string();
k0=bin2sym(hextobin(key[index-1]));
copy( k0.begin(), k0.end(), enckey );
enckey[k0.length()] = 0;
RC5key(enckey, sizeof(enckey), ky);
std::cout<<bintosym(dectobin(*ky))<<" kunci ky "<<bin2sym(hextobin(key[index-
   1]))<<"\n";
tkirim=(index+1)*t;
SystemWallClockMs mspb;
mspb.Start();
clock_gettime(CLOCK_MONOTONIC, &start); //satuan detik
 /////// SIG MODIFY///////
lpesan = lmax;
pesan = data + usid[0]; //data maks 33
potongpesan(pesan,&pot1,&pot2); //pot2 maks 23
sigpesan= uspriv[0].sign_modified(pesan,"");
m2=pot2.substr(0,pot2.length()-2);//pesan2 tanpa uid, m2 maks 21
puzpesan = cptt (m2, sigpesan, bin2sym(hextobin(key[index-1])),
    bin2sym(hextobin(key[index])), ky, &lpesan); //puzpesan =
    m2+bin2sym(hextobin(key[index]))+puzzle
std::cout<<"lmax = "<<lmax<<" l= "<<lpesan<<" lprev= "<<lprev <<std::endl;
std::cout<<lpesan<<" =l, NHASH yang dijalanlan = "<< nhash <<std::endl;
tagpesan = buattag(index, lprev, lpesan); //dalam binary
std::cout<<tagpesan.length()<<" tagpesan "<<bin2sym(tagpesan).length()<<" =
    "<<bintosym(tagpesan)<<std::endl;
std::cout<<pubtoreal(sigpesan).length()<<" pubtoreal(sigpesan) =
    "<<pubtoreal(sigpesan)<<std::endl;
std::cout<<komprespubk(uspub[0].to_string()).length()<<" pubkey =
    "<<komprespubk(uspub[0].to_string())<<std::endl;
std::cout<<usid[0].length()<<" usid[0] = "<<usid[0]<<symtobin(usid[0])<<std::endl;
std::cout<<puzpesan.length()<<" enc puzpesan = "<<symtobin(puzpesan)<<std::endl;
```

```
kpesan=pubtoreal(sigpesan)+komprespubk(uspub[0].to_string())+usid[0]+puzpesan+
    bin2sym(tagpesan);
std::cout<<kpesan.length()<<" kpesan = "<<kpesan<<std::endl;
clock_gettime(CLOCK_MONOTONIC, &finish);
mspb.End();
elapsed = (finish.tv_sec - start.tv_sec);
elapsed += (finish.tv_nsec - start.tv_nsec) / 1000000000.0 ; //second
myfile <<lpesan<<";;"<<(elapsed*1000)<<";;"<<nhash<<"\n";
myfile <<"valid paket = "<<print_timediff("chrono::system_clock::now", start,
    finish)<<"\n";
std::cout<<"waktu user MSP = "<< mspb.GetElapsedUser() <<std::endl;
std::cout<<"waktu sistem MSP = "<< mspb.GetElapsedSystem()
    <<tkirim<<std::endl;
kpesan = symtobin(kpesan);
kirim_data(sta1, Ipv6Address (iic2.GetAddress (1, 1)), ndc1.Get(0), m_dr,kpesan,
    41);//kpesan
for(int i=1;i<8;i++) //jmlus
{   clock_gettime(CLOCK_MONOTONIC, &start2);
    index++; // commitment value idx = 0,
    idx = std::bitset<16>(index).to_string();
    k0=bin2sym(hextobin(key[index-1]));
    copy( k0.begin(), k0.end(), enckey );
    enckey[k0.length()] = 0;
    RC5key(enckey, sizeof(enckey), ky);
    lpesan = lmax;
    pesan = data + usid[i]; //data maks 33
    potongpesan(pesan,&pot1,&pot2); //pot2 maks 23
    sigpesan= uspriv[i].sign_modified(pesan,"");
    m2=pot2.substr(0,pot2.length()-2);//pesan2 tanpa uid, m2 maks 21
    puzpesan = cptt (m2, sigpesan, bin2sym(hextobin(key[index-1])),
        bin2sym(hextobin(key[index])), ky, &lpesan);
    std::cout<<"lmax = "<<lmax<<" l= "<<lpesan<<" lprev= "<<lprev <<std::endl;
    std::cout<<lpesan<<" =l, NHASH yang dijalanlan = "<< nhash <<std::endl;
    myfile <<"PRoSES puzzle "<<lpesan<<" =l, NHASH yang dijalanlan = "<<
        nhash <<"\n";
    tagpesan = buattag(index, lprev, lpesan); //dalam binary
    kpesan=pubtoreal(sigpesan)+komprespubk(uspub[i].to_string())+usid[i]+puzpes
        an+bin2sym(tagpesan);
    kpesan = symtobin(kpesan);
    clock_gettime(CLOCK_MONOTONIC, &finish2);
    myfile <<"PRoSES KIRIM = "<<print_timediff("chrono::system_clock::now",
        start2, finish2)<<"\n";
    kirim_data(sta1, Ipv6Address (iic2.GetAddress (1, 1)), ndc1.Get(0),
        m_dr,kpesan, 61+(5*i));
}
myfile.close();

//3. KIRIM SIGNATURE PALSU + KEY VALID + INDEX VALID
std::cout<<"PENGIRIMAN KEDUA= "<<std::endl;
for (int i=1;i<3;i++)
```

```
    {
        index++;
        tkirim=index*t;
        std::cout<<"tkirim = "<<tkirim<<std::endl;
        idx = std::bitset<16>(index).to_string();
        kidx = bin2sym(hextobin(key[index-1]));
        copy( kidx.begin(), kidx.end(), enckey );
        enckey[kidx.length()] = 0;
        RC5key(enckey, sizeof(enckey), ky);
            std::cout<<enckey<<"  ukuran  "<<sizeof(enckey)<<"  kunci  ky
            "<<*ky<<"\n";
        myfile.close();
    }

    AsciiTraceHelper ascii;
    csma.EnableAsciiAll (ascii.CreateFileStream ("pcap/cas_bu.tr"));
    csma.EnablePcapAll (std::string ("pcap/cas_bu"), true);

   // Flow monitor
   Ptr<FlowMonitor> flowMonitor;
   FlowMonitorHelper flowHelper;
   flowMonitor = flowHelper.InstallAll();

    Simulator::Stop (Seconds (takhir));
    AnimationInterface::SetNodeDescription (sta1, "sta1"); // Optional
    AnimationInterface::SetNodeDescription (sta2, "sta2"); // Optional
    AnimationInterface::SetNodeDescription (r1, "r1"); // Optional
    AnimationInterface::SetNodeDescription (r2, "r2"); // Optional

    AnimationInterface anim ("xml/cas_bu.xml"); // Mandatory
    //anim.EnablePacketMetadata (); // Optional

    NS_LOG_INFO ("Run Simulation.");
    Simulator::Run ();

    flowMonitor->SerializeToXmlFile("xml/FMcas_bu.xml", true, true);

    Simulator::Destroy ();
    myfilerec.close();
    NS_LOG_INFO ("Done.");
}
```

Halaman ini sengaja dikosongkan

Lampiran H. Kode program testbed skema puzzle dinamis pada Arduino

Skema DCP pada Arduino

```
#include <sha1.h>
#include <rc5.h>
#include "BigNumber.h"
#include "limits.h"
#include <Arduino.h>
#include <LiquidCrystal.h>
#include "uECC.h"
#include "BigNumber.h"

extern "C" {
static int RNG(uint8_t *dest, unsigned size) {
  while (size) {
    uint8_t val = 0;
    for (unsigned i = 0; i < 8; ++i) {
      int init = analogRead(0);
      int count = 0;
      while (analogRead(0) == init) {
        ++count;
      }
      if (count == 0) {
        val = (val << 1) | (init & 0x01);
      } else {
        val = (val << 1) | (count & 0x01);
      }
    }
    *dest = val;
    ++dest;
    --size;
  }
  return 1;
}
}

char ch;
int Contrast=15;
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setupwifi()
{
  Serial2.print("AT+CIPMUX=1\r\n");
  delay(1000);
  Serial2.print("AT+CIPSERVER=1,8100\r\n"); //create server
  delay(1000);
  Serial2.print("AT+CWQAP\r\n");
  delay(1000);
```

```
  Serial2.print("AT+CIPSTA=\"10.122.69.235\",\"10.122.69.1\",\"255.255.255.0\"\r\n");
//LABB304
 // Serial2.print("AT+CIPSTA=\"10.122.14.241\",\"10.122.8.1\",\"255.255.248.0\"\r\n");
//JTE-WIFI
   delay(1000);
  Serial2.print("AT+CIFSR\r\n");
  delay(1000);
  Serial2.print("AT+CWJAP=\"Lab B304\",\"secret304\"\r\n");
  //Serial2.print("AT+CWJAP=\"JTE-WIFI\",\"\"\r\n");
  delay(1000);
  Serial2.print("AT+CIPDINFO=1");
  delay(1000);
}

void kirimdata(String data)
{
  //Serial2.println("AT+CIPSTART=1,\"TCP\",\"10.122.69.202\",8899");
  delay(1000);
  Serial2.print("AT+CIPSEND=1,");
  Serial2.println(data.length());
  delay(1000);

  String response;
  if(Serial2.available())
  {
    while(Serial2.available()) //baca dari ESP
    {
      response = Serial2.readStringUntil('\n');
    }
    Serial.print("fungsi kirim Serial2.avail ");
    Serial.println(response);
    if(response=">")
      Serial2.print(data);
    else
      Serial.println("pesan tidak terkirim");
  }
}

String resSHA1(String isi) {
  uint8_t *hash;
  Sha1.init();
  Sha1.print(isi);
  hash = Sha1.result();
  int i;
  String jj;
  for (i=0; i<20; i++) {
    jj = jj  + "0123456789abcdef"[hash[i]>>4];
    jj = jj + "0123456789abcdef"[hash[i]&0xf];
  }
  return jj;
```

```
}


String reshmac(String isi,String comkey) {
  uint8_t *hash;
  //uint8_t *key= (unsigned char*)comkey;
  uint8_t key[8];
  comkey.getBytes(key,9);
  Sha1.initHmac(key,8);
  Sha1.print(isi);
  hash = Sha1.resultHmac();
  int i;
  String jj;
  for (i=0; i<20; i++) {
    jj = jj  + "0123456789abcdef"[hash[i]>>4];
    jj = jj + "0123456789abcdef"[hash[i]&0xf];
  }
  return jj;
}

int i, c;

String dekripsiakhir(String comkey, String recencm){
    char enckey[8];
    String hasildec;
    comkey.getBytes(enckey, 9);
    enckey[comkey.length()] = 0;
    rc5::word32 *ky = new unsigned long int[20]();
    rc5::RC5key(enckey, 1, ky);
    hasildec=rc5::decRC5(rc5::bintohex(rc5::symtobin(recencm)), ky);
          delete(ky);
    return (hasildec);
}

void DCP(String comkey1, String recpaket1, String pub11,int lmaks)
{
  String recsig="",recmac="",recl="",recencm="",hmaccek="";
  //int lmaks = 24;
  int lprev,l,index, templ, ver_tag=0;
  lprev = lmaks;
  l = lmaks;
  recsig = recpaket1.substring(0,80);
  recmac = recpaket1.substring(80,120);


  while((l>0) and (ver_tag == 0))
  {
    Serial.println("iterasi");
    templ = (index+1)+ lprev + l;
    recl=(rc5::reversebin(rc5::dectobin(templ))).substring(24,32);
```

```
  recl = resSHA1(recl);
  if(rc5::hextobin(recl).substring(0,l) == rc5::hextobin(recmac).substring(0,l))
     ver_tag = 1;
  else
     l=l-1;
 }


 if(ver_tag == 1)
 {
     Serial.println("TAG valid");
     String hsig, hmacrecfull;
     recencm = recpaket1.substring(120,recpaket1.length());
     recencm = rc5::bintosym(rc5::hextobin(recencm));
     comkey1= rc5::bintosym(rc5::hextobin(comkey1));
      //HMAC arduino
     hmaccek=reshmac(recencm,comkey1);
     //HMAC dari paket
     hsig = resSHA1(rc5::bintosym(rc5::hextobin(recsig)));
     hmacrecfull =
rc5::hextobin(hsig).substring(0,l)+rc5::hextobin(recmac).substring(l,recmac.length()*4);
     hmacrecfull = rc5::bintohex(hmacrecfull);
     if(hmacrecfull == hmaccek)
     {

         Serial.println("HMAC valid");
         String hasildec,isi,k1;
         hasildec = dekripsiakhir( comkey1, recencm);
         k1=hasildec.substring(hasildec.length()-8,hasildec.length());
         isi = hasildec.substring(0,hasildec.length()-12);
         if(resSHA1(k1).substring(0,16) == rc5::bintohex(rc5::symtobin(comkey1)))
         {
             Serial.println("session key valid");
             const struct uECC_Curve_t * curve = uECC_secp160r1();
             uint8_t public1[40];
             uint8_t hash[21] = {0};
             uint8_t sig[64] = {0};
             recsig=rc5::bintosym(rc5::hextobin(recsig));
             pub11 = rc5::bintosym(rc5::hextobin(pub11)) + isi;
             recsig.getBytes(sig,41);
             pub11.getBytes(public1,49+1);
             isi.getBytes(hash,isi.length()+1);

             if (!uECC_verify(public1, hash, sizeof(hash), sig, curve))
             {
                 Serial.println("uECC_verify() ASLI failed");
             }
             else
             {
                 Serial.println("uECC_verify() ASLI ISO");
```

```
            } // end if verifikasi
          } // end if seskey
        } // end if verhmac
  } // end if vertag
  else
  {
    Serial.println("vertag tak valid");
  }
}


void setup()
{
  Serial2.begin(115200);
  Serial.begin(115200);
  setupwifi();  //  TERHUBUNG WIFI
  pinMode(13,OUTPUT);  // Pendefinisian LCD
  analogWrite(6,Contrast);
  lcd.begin(16, 2);  // set up the LCD's number of columns and rows:
  String comkey = "af17526d66fd46c9";    // KEYCHAIN
  String pub1 =
"16713a64205f94097d412580af2dbfce4fbcd53d21ba9ec99dbcd53bf13e319e39b7e78c8f
0f81c6";
  unsigned long ms;
  ms = micros();
  DCP(comkey, recpaket, pub1,24);
  Serial.println(" ");
  Serial.print((micros() - ms));
  Serial.println(" micros");
  kirimdata(isi);
  Serial.println("Sudah Kirim");
}

String comkey = "af17526d66fd46c9";

  String pub1 =
"16713a64205f94097d412580af2dbfce4fbcd53d21ba9ec99dbcd53bf13e319e39b7e78c8f
0f81c6";

const char EOL = '\n'; // End of line
void loop()
{
  int responseCounter = 0;
  if(Serial2.available())
  {
    String data_rec;
    while(Serial2.available()) //baca dari ESP
    {
      String response = Serial2.readStringUntil('\r'); //'\n'
      if (responseCounter == 0)
```

```
{
  Serial.println("Response Received:");
}
Serial2.setTimeout(10000);
String pieces[6];
  if(response.substring(1,5) == "+IPD")
  {
   //Serial.println("masuk IPD");
    char buf[response.length()];
    response.toCharArray(buf, sizeof(buf)+1);
    char *p = buf;
    char *str,*str2;
    char *pn = buf;
    int i=0;
    while ((str = strtok_r(p, ":", &p)) != NULL) // delimiter is the semicolon
      {
        pieces[i]=str;
        data_rec= pieces[1];
        pieces[i]="";
        i++;
      }
    Serial.print("isi data = ");
    Serial.println(data_rec);
    i=0;
    while ((str2 = strtok_r(pn, ",", &pn)) != NULL) // delimiter is the semicolon
      {
        pieces[i]=str2;
        i++;
      }
    Serial.print("IP = ");
    Serial.println(pieces[3]);
    Serial.print("panjang pesan = ");
    Serial.println(pieces[2]);

    if(data_rec.length()==pieces[2].toInt()){

          Serial.println("panjang pesan COCOK ");
          Serial.println(comkey);
          Serial.println(data_rec);
          Serial.println(pub1);
          DCP(comkey, data_rec, pub1,24);

      }
    else
    {
      Serial.println("panjang pesan tidak sesuai ");
      }
    analogWrite(9,28836);
    digitalWrite(13,LOW);
    digitalWrite(13,HIGH);
```

```
      lcd.setCursor(0, 1);
      lcd.print(data_rec);          // Print a message to the LCD.
    }
    else
    {

     Serial.println(response);
    }

    responseCounter++;
    Serial.println();
    Serial.println("============");
    Serial.println();
  }
}
if(Serial.available())
{
  String command="";
  char c;
    while(Serial.available() && c!=EOL)
    {
      // membaca satu karakter
      c = Serial.read();
      command+=c;
    }

    if (c==EOL) {
     if(command.substring(0,2) == "AT")
       Serial2.println(command);
     else if(command.substring(0,4) == "saya"){
       kirimdata("saya");
     }
     else
     {
       Serial.print("cc= ");
       Serial.println(command);
       prinLCD(command);
     }
    }
} //endif serial avail

 digitalWrite(13,LOW);
 delay(1000);
 digitalWrite(13,HIGH);
// set the cursor to column 0, line 1
// (note: line 1 is the second row, since counting begins with 0):
lcd.setCursor(0, 1);
// print the number of seconds since reset:
lcd.print(millis()/1000);
```

```
}

void prinLCD(String ch)
{
    Serial.println("Current contrast");
   Serial.println(Contrast);
   if(ch.substring(0,1)=="C" && Contrast<255)
   {
    Contrast=Contrast+1;
   }
   if(ch.substring(0,1)=="D" && Contrast>0)
   {
    Contrast=Contrast-1;
   }

   if(ch.substring(0,1)=="N")
   {
    analogWrite(9,28836);
   }
   if(ch.substring(0,1)=="F")
   {
    analogWrite(9,0);
   }
   analogWrite(6,Contrast);
   Serial.println("Current contrast");
   Serial.println(Contrast);

}
```

BIOGRAFI PENULIS

Nama	:	Farah Afianti
Tempat / Tanggal Lahir	:	Surabaya / 24-10-1986
Pekerjaan	:	Dosen
Alamat Rumah	:	Permata Buah Batu Blok G no 35 Bojongsoang Bandung
Email	:	farah.afianti16@mhs.ee.its.ac.id farah.aphie@gmail.com

A. Riwayat Pendidikan

- SDN Tanjungsari 97 Surabaya
- SLTP Negeri 1 Surabaya
- SMU Negeri 6 Surabaya
- S1 Teknik Informatika IT Telkom
- S2 Teknik Informatika Universitas Telkom

B. Riwayat Pekerjaan

- Instruktur Program Profesional D1 pada Sekolah Tinggi Teknologi Telkom jurusan Database Management System 2007-2008
- System Support pada kantor pusat PT Pelabuhan Indonesia II Masa kerja 2008-2012
- Dosen luar biasa Fakultas Teknik Informatika Universitas Telkom Masa kerja 2014-2015

C. Kegiatan Pelatihan

- ✓ **Kegiatan Pelatihan**
  - Cisco Network Academy Program semester 1-4 di Fakultas Teknik - Universitas Pancasila Jakarta (2010)

- Manajemen keamanan informasi berdasarkan ISO/IEC 27000 oleh PT LAPI Ganeshatama Consulting (2011)
- Manjemen resiko teknologi informasi berdasarkan RISKIT oleh Transforma Bandung (2011)
- *Procurement* dan aspek legal pada *Information and Communication Technology* oleh Sharing Vision (2012)

D. Publikasi Ilmiah selama studi program Doktor

✓ **Seminar Internasional**

- Afianti, F., Wirawan, & Suryani, T. (2017, November). *Filtering methods for broadcast authentication against PKC-based denial of service in WSN: A survey. In Fifth International Conference on Wireless and Optical Communications* (Vol. 10465, p. 1046503).

✓ **Jurnal Internasional**

- Afianti, F., Wirawan, & Suryani, T. (2018). *Dynamic Cipher Puzzle for Efficient Broadcast Authentication in Wireless Sensor Networks*. Sensors, 18(11), 4021.
- Afianti, F., Wirawan, & Suryani, T. (2019). *Dynamic Message Puzzle as Pre-Authentication Scheme in Wireless Sensor Networks*. International Journal on Advanced Science, Engineering and Information Technology, 9(1), 204-212.
- Afianti, F., Wirawan, & Suryani, T. (2019). *Lightweight and DoS Resistant Multiuser Authentication in Wireless Sensor Networks for Smart Grid Environments*. IEEE Access, 7(1), hal. 67107–67122.