BAB 2 LANDASAN KEPUSTAKAAN

2.1 Kajian Pustaka

Terdapat beberapa penelitian yang sebelumnya telah dilakukan oleh peneliti lainnya dalam bentuk jurnal serta literatur yang berkaitan dengan penelitian yang akan dilakukan. Beberapa penelitian tersebut kemudian dijadikan sebagai referensi dalam melaksanakan penelitian ini, jurnal serta literatur tersebut berisi tentang perkembangan teknologi *Internet of Things* (IoT), protokol yang digunakan khususnya *Message Queuing Telemetry Transport* (MQTT), implementasi *Access Control List* (ACL) sebagai metode otorisasi dan NodeMCU yang digunakan sebagai perangkat yang banyak digunakan dalam IoT. Kajian pustaka akan disajikan dalam bentuk tabel yang membahas perbandingan antara penelitian sebelumnya dan penelitian yang akan dilakukan dalam bentuk tabel sebagai berikut.

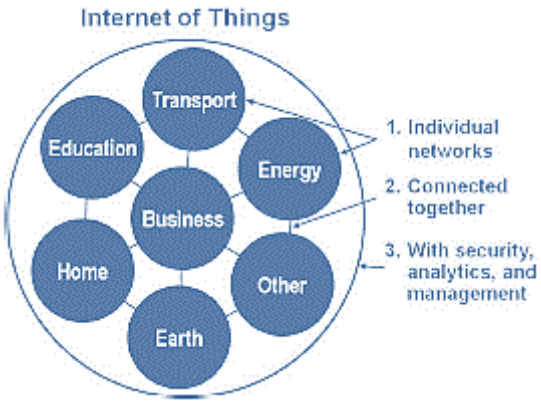
Tabel 2.1 Kajian Pustaka

No.	Nama Penulis, Tahun, Judul	Persamaan Penelitian	Perbedaan Penelitian	
			Penelitian Sebelumnya	Penelitian yang Akan Dilakukan
1	S. Gusmeroli, S. Piccione & D. Rotondi, 2012, *IoT Access Control Issues: a Capability Based Approach* (Gusmeroli dkk., 2012)	Penggunaan ACL sebagai mekanisme otorisasi dalam sebuah aplikasi IoT.	ACL didesain berdasarkan kapabilitas tiap pengguna, sehingga manajemen ACL menjadi lebih rumit.	ACL didesain berdasarkan sebuah entitas peran, dimana tiap pengguna memiliki suatu peran. Hal ini memudahkan manajemen ACL jika terjadi perubahan.
2	Y. Upadhyay, A. Borole & D. Dileepan, 2016, *MQTT Based Secured Home Automation System* (Upadhyay dkk., 2016)	Menerapkan mekanisme pengamanan pada protokol MQTT.	Mekanisme pengamanan dilakukan menggunakan enkripsi payload, meskipun ACL telah diterapkan namun penentuan hak akses pengguna tidak didefinisikan.	Mekanisme pengamanan dilakukan pada ACL yang akan mengatur penentuan hak akses pengguna terhadap topik yang dapat diakses.
3	A.Niruntasukrat, C. Issariyapat & P. Pongpaibool, 2016, *Authorization Mechanism for*	Menerapkan mekanisme pengamanan menggunakan autentikasi dan	- Mekanisme autentikasi dan otorisasi dilakukan dengan	- Mekanisme autentikasi dan otorisasi dilakukan dengan menggunakan library yang

	MQTT-based Internet of Things (Niruntasukrat dkk., 2016)	otorisasi pada protokol MQTT.	memodifikasi Oauth1.0a - *Access Token* dari Oauth1.0a digunakan sebagai kredensial - Mekanisme otorisasi dilakukan dengan persetujuan pemilik aplikasi tiap terjadi proses otorisasi dan jika telah mendapatkan persetujuan pengguna tersebut akan mendapatkan hak akses secara penuh.	disediakan oleh broker - Kredensial yang digunakan berupa JSON Web Token (JWT) - Mekanisme Otorisasi dilakukan berdasarkan peran yang didefinisikan di dalam database, memiliki batasan akses pada tiap peran dan tanpa memerlukan persetujuan dari pemilik tiap proses otorisasi
4	M. Saikrishna & G. Vijaykiran, 2017, *IOT Based Home Electrical Appliances Control Using Node MCU* (Saikrishna dkk. 2017)	Mengontrol Aplikasi *Home Automation* menggunakan nodeMCU.	NodeMCU digunakan sebagai sensor dan aktuator pada aplikasi *Home Automation* menggunakan GSM modem, dengan mengirimkan pesan melalui SMS	NodeMCU digunakan sebagai sensor dan aktuator yang akan bertindak sebagai publisher atau subscriber pada protokol MQTT pada sistem untuk menerapkan mekanisme otorisasi.

2.2 *Internet of Things* (IoT)

Internet of Things (IoT) merupakan sebuah paradigma dimana "semua" objek dapat memiliki konektivitas dengan sebuah tujuan untuk membuat kemampuan objek tersebut untuk saling terhubung dan saling berkolaborasi kapanpun, dimanapun dan dalam bentuk apapun. Banyak domain pengaplikasian contohnya personal dan sosial, transportasi, bisnis, servis, utilitas dan monitoring (Aziz, B., 2015). IoT melibatkan implementasi *smart-object*, yaitu perangkat embedded yang dilengkapi dengan sensor dan aktuator yang saling terhubung satu sama lain dan ke internet. Perangkat ini terhubung secara cerdas sehingga menghasilkan sebuah interaksi yang baru antara benda dan menusia, serta antara benda dan benda lainnya (Evans, D., 2011).



Gambar 2.1 Diagram Konsep IoT

Sumber : Evans, D. (2011)

IoT memungkinkan sebuah objek fisik untuk dapat "berpikir", "melihat" atau "mendengarkan" yang memungkinkan objek terebut untuk melakukan tugas tertentu serta saling "berbicara" dengan objek yang lain, untuk kemudian berbagi informasi untuk menentukan suatu keputusan. IoT memiliki komponen utama yang terdiri dari *identification*, *sensing*, *computation*, *communication*, *semantics* dan *services* yang memiliki peran masing-masing untuk memenuhi fungsionalitas tertentu. *Identification* merupakan kemampuan untuk menyediakan sebuah identitas bagi tiap objek yang ada dalam IoT yang digunakan untuk melakukan penyesuaian dengan permintaan yang ada. *Sensing* merupakan kemampuan untuk mengumpulkan data dari berbagai informasi yang dapat disimpan dan diperlukan untuk pengolahan lebih lanjut. *Computation* merupakan kemampuan untuk mengolah data yang berperan sebagai *processing unit* untuk mendapatkan sebuah kesimpulan yang berguna bagi fungsionalitas. *Communication* merupakan kemampuan untuk saling terhubung dan berinteraksi antara objek satu dan lainnya. *Semantics* merupakan kemampuan untuk mengambil pengetahuan secara akurat dari berbagai objek untuk menyediakan fungsionalitas. *Services* merupakan kemampuan utama yang menyatukan kemampuan lainnya untuk melakukan virtualisasi terhadap data, pengumpulan data, pengambilan keputusan, dan penyediaan akses terhadap data tersebut (Al-Fuqaha, dkk., 2015).

2.3 Mekanisme Autentikasi & Otorisasi

Menurut Andress, J. (2011), terdapat tiga langkah untuk mengamankan informasi yang dimulai dari identifikasi, autentikasi dan otorisasi. Identifikasi merupakan proses klaim identitas dimana kita menuntut identitas pihak tertentu (bisa berupa seseorang atau sesuatu), Autentikasi merupakan sarana atau metode validasi terhadap klaim identitas pihak tersebut apakah benar atau tidak dan Otorisasi memungkinkan kita untuk menentukan kegiatan pihak tersebut di dalam sistem apakah dibolehkan atau dilarang. Perlu diketahui bahwa ketiga mekanisme ini tidak sama dan memiliki batasan dari proses yang dilakukan, identifikasi tidak melebihi dari proses klaim dan tidak melibatkan proses verifikasi atau validasi klaim tersebut karena proses merupakan bagian proses autentikasi, sedangkan

autentikasi tidak akan menentukan kegiatan apa yang akan dilakukan oleh pihak yang terautentikasi dalam sistem karena ini merupakan bagian dari proses otorisasi. Contoh perbedaan ketiganya adalah saat pengguna ingin mengakses sistem kita menerapkan form login sebelum sistem dapat diakses merupakan proses identifikasi, ketika pengguna memasukkan identitas maka akan diperiksa apakah identitas tersebut sesuai dengan persyaratan merupakan proses autentikasi dan ketika pengguna tersebut ingin mengakses suatu fungsionalitas sistem maka akan diperiksa apakah pengguna tersebut diizinkan atau tidak merupakan proses otorisasi.

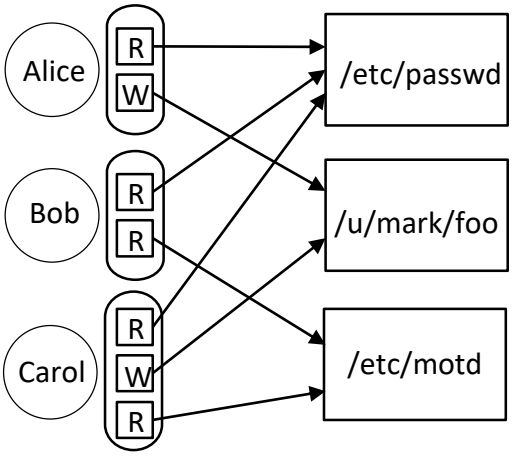
Gambar 2.2 Mekanisme Keamanan Informasi

Sumber : Andress, J. (2011)

Implementasi otorisasi dapat dilakukan dengan menggunakan *Access Control List* (ACL). Otorisasi memungkinkan kita untuk menentukan apakah pihak tertentu diizinkan atau ditolak untuk mengakses serta ACL memungkinkan kita untuk mengatur hak akses secara rinci. ACL dapat dibangun dengan berbagai cara, berdasarkan pada atribut fisik, kumpulan aturan, daftar per-individu atau per-sistem atau dalam bentuk yang lebih kompleks lagi (Andress, J., 2011).

2.3.1 *Access Control List* (ACL)

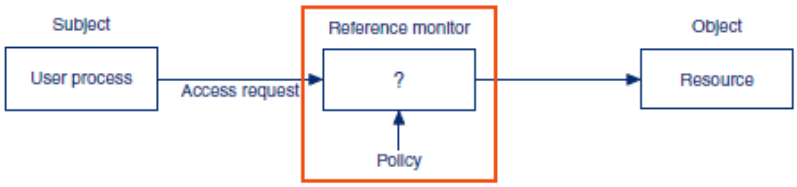
Access Control List (ACL) adalah sebuah aturan yang mengatur subjek mana yang dapat memiliki hak akses terhadap objek yang mana dan jenis akses seperti apa yang dapat dilakukan (Laskov, P., 2014). ACL memiliki tujuan untuk memastikan pengguna yang memiliki hak akses mendapatkan data sesuai keinginan dan pengguna yang tak memiliki hak akses untuk tidak mengakses data tersebut (Gusmeroli, dkk., 2012).



Gambar 2.3 Contoh Access Control List (ACL)

Sumber : Gusmeroli, dkk. (2012)

Menurut Laskov, P. (2014), Struktur dari ACL seperti yang digambarkan pada gambar 2.4 terdiri dari Subjek, yaitu entitas yang melakukan permintaan untuk melakukan akses. Bersifat aktif dalam sistem dan memiliki kegiatan yang dilakukan terhadap sistem. Objek, yaitu entitas yang aksesnya dilakukan permintaan. Bersifat pasif pada sistem seperti memori, file, direktori, fungsi, dll. *Reference Monitor*, yaitu entitas yang mengatur pemilihan autorisasi. Bersifat abstrak yang biasanya merupakan mekanisme yang mengontrol permintaan akses. Hak Akses, yaitu entitas yang merepresentasikan berbagai macam tipe akses. Biasanya direpresentasikan berbagai operasi yang di-*support* oleh sistem seperti *read, write, delete, execute, search, append*, dll. *Policies,* yaitu entitas yang berisi kumpulan hak akses yang biasanya terkait pada sebuah objek yang dapat diakses oleh subjek tertentu.

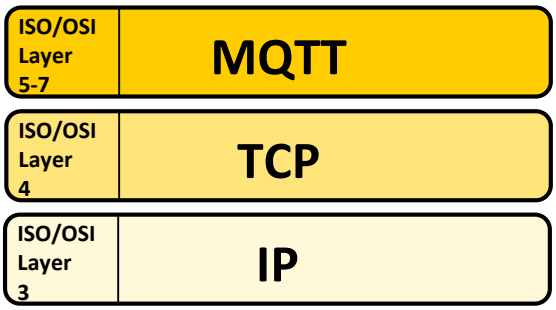


Gambar 2.4 Struktur Access Control List (ACL)

Sumber : Laskov, P. (2014)

Ketika ACL dibangun, pada umumnya mereka dibangun pada sumberdaya tertentu dan memiliki identifikasi mengenai pihak mana saja yang memiliki akses terhadap sumberdaya tersebut dan aksi apa yang dapat dilakukan oleh pihak tersebut. Terdapat beberapa model ACL yang sering dijumpai yaitu *discretionary-based*, *mandatory*, *rule-based*, *role-based* dan *attribute-based*. Terdapat 4 tujuan utama yang harus dicapai oleh ACL yaitu mengizinkan hak akses, menolak hak akses, membatasi hak akses dan mencabut hak akses. Kebanyakan ACL akan menolak hak akses secara bawaan, dan hanya pihak yang telah terautorisasi yang diizinkan untuk melakukan akses (Andress, J., 2011).

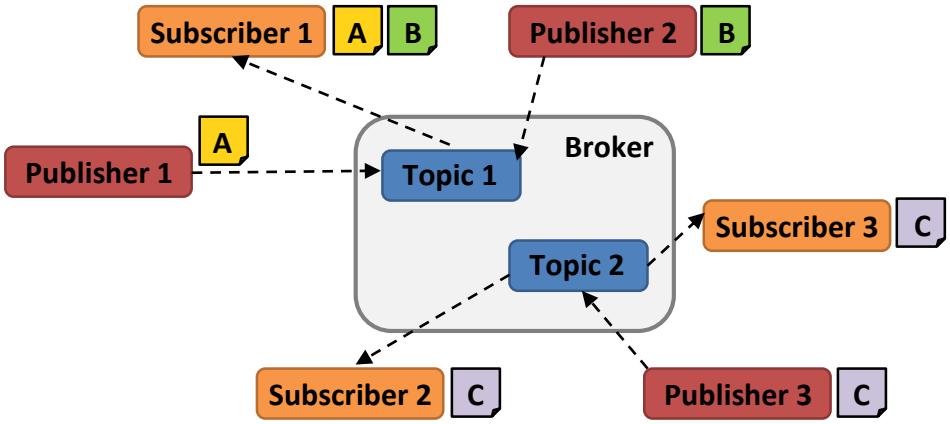
2.4 *Message Queuing Telemetry Transport* (MQTT)



Gambar 2.5 MQTT OSI Layer

Sumber : Götz, C. (2015)

Message Queuing Telemetry Transport (MQTT) merupakan protokol yang digunakan untuk komunikasi pada *machine-to-machine* (M2M) / *Internet of Things* (IoT). MQTT adalah protokol komunikasi *topic-based publish/subscribe* yang di desain secara ringan, sederhana, terbuka dan mudah untuk diimplementasi (OneM2M, 2016). Protokol ini berjalan diatas *stack* TCP/IP serta memiliki ukuran paket data yang memiliki *low-overhead* yang kecil (yang paling kecil hanya 2 bytes) sehingga daya yang dikonsumsi juga semakin kecil. Bentuk data yang dapat dikirimkan melalui protokol ini juga beragam, dimulai dari data biner, teks, xml dan JSON. TCP/IP stack sendiri seudah didukung oleh berbagai mikrokontroler seperti seri STM32Fx7 serta perangkat seperti Arduino + *Ethernet Shield*, Arduino Yun, Raspberry Pi, ESP8266 wifi soc dan masih banyak lagi (Warda, A., 2017).

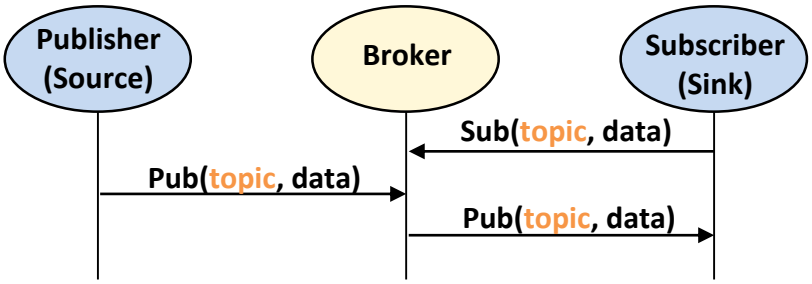


Gambar 2.6 Arsitektur Protokol Komunikasi MQTT

Sumber : OneM2M. (2016)

Model komunikasi *topic-based publish-subscribe* merupakan komunikasi berbasis *event-driven* yang mana memungkinkan sebuah pesan untuk di-*pushed* menuju klien. Pihak pengatur komunikasi sentralnya terdapat pada MQTT broker, yang akan bertugas untuk mengatur transaksi/distribusi pesan antara pengirim dan penerima. Setiap kali klien mengirim (*publish*) pesan menuju broker, mereka akan menyertakan sebuah topik terkait pesan tersebut. Topik merupakan informasi untuk *routing* yang akan dilakukan oleh broker. Klien yang ingin menerima pesan akan *subscribe* topik tertentu dan broker akan mengirimkan semua pesan yang memiliki topik tersebut ke klien. Oleh karena itu, tiap klien tidak saling mengetahui keberadaan satu sama lain, mereka hanya akan bertukar pesan berdasarkan kecocokan topik. Sebuah pesan pub dapat dikirimkan ke banyak *subscriber* sekaligus jika topik terkait pesan tersebut cocok dengan topik yang diminati oleh beberapa *subscriber*. Arsitektur semacam ini memungkinkan solusi dengan skalabilitas tanpa adanya dependensi antara pengirim dan penerima yang biasa disebut *decoupling* (Sengupta, S. & Ghosh, D., 2011). Konsep dimana klien tidak saling mengetahui keberadaan dan identitas satu sama lain disebut sebagai *space decoupling*, dan konsep dimana klien satu dan lainnya tidak harus berada dalam waktu yang sama disebut sebagai *time decoupling* (Warda, A., 2017).

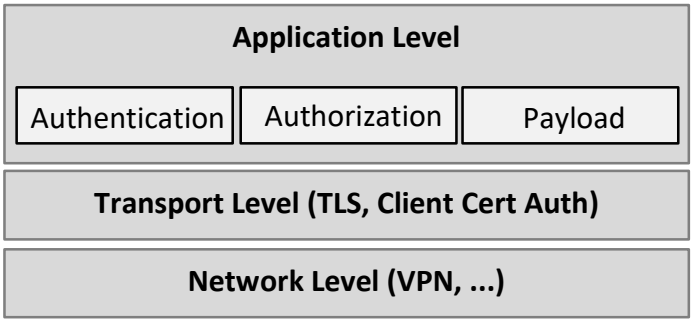
Distribusi data pada MQTT terikat akan sebuah topik yang digunakan oleh *publisher* dan *subscriber* untuk melakukan transaksi data. Topik tersusun kedalam *topic tree*, yang mana digambarkan dalam sebuah hirarki menggunakan *forward-slash* "/" sebagai pembatas. Klien mendaftarkan ketertarikan mereka pada sebuah topik dengan menyediakan satu atau banyak topik. Topik dapat berupa nama dari topik, atau dapat berupa *wildcard*, yang memungkinkan klien untuk *subscribe* pada beberapa topik sekaligus, diantara satu level atau beberapa level pada topic tree (OneM2M, 2016). *Wildcard* tanda plus "+" merupakan *wildcard* satu level yang memungkinkan untuk digantikan dengan nilai apapun di level tersebut, sedangkan *wildcard* tanda pagar "#" merupakan *wildcard* yang digunakan untuk *subscribe* ke beberapa topik di dalam dan di bawah level tersebut (Sengupta, S. & Ghosh, D., 2011).



Gambar 2.7 Alur Komunikasi Publish/Subscribe MQTT

Sumber : Hunkeler, dkk. (2008)

MQTT mendukung *Quality of Services* (QoS) tergantung dari seberapa *reliable* pesan akan dikirimkan, mqtt membedakan 3 level reliabilitas, yaitu QoS level 0 yang menyediakan servis dimana pesan akan dikirimkan sekali atau tidak sama sekali ke destinasi, tanpa pengiriman ulang atau pemberitahuan bahwa pesan telah sampai. QoS level 1 yang menyediakan servis diamana pesan akan terus dikirimkan ulang hingga penerima pesan memberitahu bahwa pesan telah diterima. Oleh karena itu, pesan dapat dipastikan akan sampai namun dapat terjadi duplikasi pesan dikarenakan pengiriman ulang secara terus menerus. QoS level 2 yang menyediakan servis yang memastikan tidak hanya penerima menerima pesan tetapi juga pesan akan dikirimkan hanya sekali oleh pengirim tersebut (Hunkeler, dkk., 2008).



Gambar 2.8 Keamanan pada MQTT

Sumber : Götz, C. (2015)

MQTT memiliki model keamanan yang sederhana, autentikasi di MQTT pada saat ini hanya terdapat *username* dan *password* yang bersifat opsional, untuk *integrity* dan *confidentiality* pada MQTT dapat diatur menggunakan standar protokol keamanan seperti *secure socket layer* (SSL), *tranport layer security* (TLS) dan *internet protocol security* (IPsec) yang juga bersifat opsional. Spesifikasi bawaan MQTT tidak menjelaskan mengenai model otorisasi namun pengimplementasian ACL pada topik tertentu telah didukung oleh MQTT (Niruntasukrat, dkk., 2016).

MQTT merekomendasikan untuk menggunakan port yang telah disediakan yaitu port 1883 (mqtt over tcp/ip) dan 8883 (mqtt over tls). Meskipun implementasi MQTT terbilang simpel, namun ada beberapa persyaratan implementasi yaitu aplikasi MQTT terikat pada sebuah *library* yang menyediakan fungsionalitas klien MQTT yang tersedia dalam berbagai bahasa pemrograman. Eclipse foundation, menyediakan klien MQTT yang *open-source* melalui *project* Paho dan server *open-source* via *project* mosquitto broker. Berbagai implementasi *open*-source dan komersial untuk mqtt juga tersedia dalam berbagai bentuk (OneM2M, 2016).

2.5 Mosquitto Broker

Mosquitto adalah sebuah broker pesan yang mengimplementasikan MQTT v3.1/v3.1.1 dan bersifat *open-source*. Sebagai MQTT broker maka semua pesan yang ada dalam sistem akan melewati mosquitto, dan memiliki karakteristik *lightweight*. Berdasarkan implementasi terbaru dapat berjalan sukses dengan jumlah koneksi klien mencapai 100.000 pada pengiriman *data-rate* dengan ukuran standar. Mosquitto juga dapat terhubung dengan berbagai aplikasi klien MQTT dan juga server, yang memungkinkan untuk meneruskan pesan dari lokasi manapun pada jaringan yang dapat diatur pada konfigurasi *bridges*. Mosquitto juga memiliki komunitas yang luas, dimana banyak penelitian dan *library* yang dapat diterapkan pada broker untuk menunjang kinerja dari mosquitto (Eclipse, 2013).

2.5.1 Mosquitto Auth-plug

Auth-Plug adalah *plugin* yang diimplementasikan pada broker mosquitto yang digunakan untuk melakukan autentikasi dan otorisasi, *plugin* ini menggunakan berbagai macam *backend* seperti MySQL, SQLite, HTTP, MongoDB, PostGres, JWT dll. Namun tidak semua *backend* memiliki fitur yang sama dikarenakan beberapa diantaranya masih dalam perkembangan. Manajemen *password* pada plugin ini menggunakan enkripsi PBKDF2 *hash*, pembuatan *password hash* telah disediakan oleh *plugin* ini. Penerapan mekanisme autentikasi pada *plugin* ini dilakukan dengan pemeriksaan *username* dan *password* pengguna melalui *backend* setiap pengguna melakukan koneksi ke broker, sedangkan mekanisme otorisasi pada *plugin* ini dilakukan dengan menggunakan *access control list* (ACL) yang akan dilakukan pemeriksaan hak akses melalui salah satu *backend* setiap ada pesan yang di-*publish* ke broker (Mens, J. P., 2016).

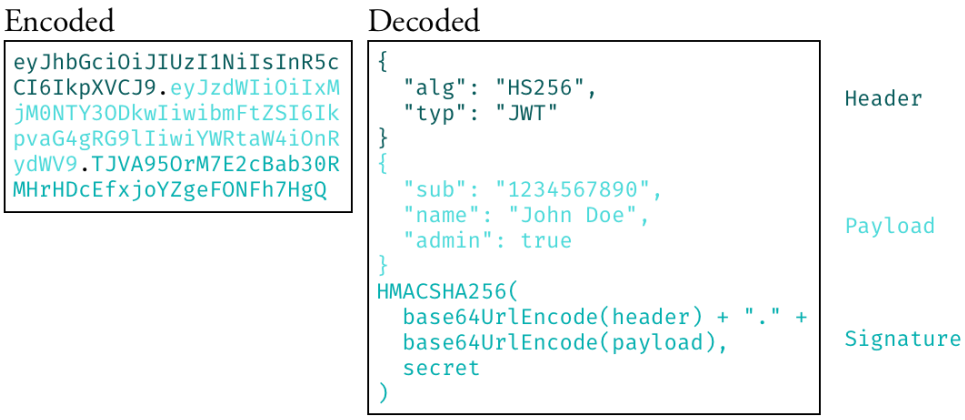
Tabel 2.2 Auth-Plug Backend Support

Sumber: Diadaptasi dari Mens, J.P. (2016)

	Autentikasi	Superuser	ACL	Static Superuser
MySQL	✓	✓	✓	✓
Redis	✓	-	✓	✓
CDB	✓	-	?	✓
SQLite	✓	-	?	✓
LDAP	✓	-	-	-
PSK	✓	?	?	?
Postgres	✓	✓	✓	✓
HTTP	✓	✓	✓	✓
JWT	✓	✓	✓	✓
MongoDB	✓	✓	✓	✓

2.6 JSON Web Token (JWT)

JSON Web Token (JWT) adalah *open standard* (RFC 7519) yang digunakan untuk mengamankan pengiriman informasi yang memiliki ukuran yang ringan dan *self-contained* sebagai sebuah JSON. Informasi ini dapat terjamin keamanannya dikarenakan telah ditandai secara digital. JWT dapat ditandai menggunakan sebuah *string* yang bersifat rahasia (menggunakan algoritma HMAC) atau menggunakan *public/private key* menggunakan RSA. JWT dapat dimanfaatkan untuk melakukan autentikasi dan pertukaran informasi sebagai pesan. (Peyrott, A., 2016) JWT memiliki 3 bagian terpisah yang dipidahkan oleh titik ".", ketiga bagian itu adalah *header*, *payload* dan *signature*.



Gambar 2.9 Struktur JWT pada kondisi Encoded & Decoded

Sumber : Nasseri, A. (2016)

Struktur dari tiap bagian di JWT pada umumnya dapat digambarkan dalam bentuk sebagai berikut <*header*>.<*payload*>.<*signature*>. Bagian *header* berisi tipe token yang berisi JWT dan algoritma *hashing* yang digunakan. Bagian *payload*

berisi klaim serta data yang disediakan oleh *issuer* mengenai pengguna dan aplikasi (contohnya seperti id pengguna, username, nama, dll.), terdapat beberapa jenis klaim pada JWT yaitu *registered claims*, *public claim*s dan *private claims*. Bagian *signature* berisi hasil *encode* dari bagian *header* dan *payload* serta algoritma enkripsi yang digunakan untuk *encoding* (contohnya algoritma HMAC SHA256) dengan menentukan sebuah teks *secret* yang berupa karakter string rahasia, bagian ini digunakan untuk memastikan identitas pengirim JWT dan memastikan bahwa token tidak dimodifikasi saat pengiriman (Nasseri, A., 2016).

2.7 NodeMCU (ESP8266)

NodeMCU merupakan sebuah *firmware* IoT bersifat *open-source* yang berjalan pada *platform* esp8266 wifi *system on chip* (SOC) dari Espressif System dan perangkat keras yang berbasis pada esp12e. Istilah "NodeMCU" sebenarnya lebih merujuk pada *firmware*-nya dibandingkan pada *development kit*-nya, karena nodeMCU dapat dijadikan *firmware* pada berbagai macam *development kit*. NodeMCU merupakan mikrokontroler yang telah terintegrasi dengan wifi, sehinga tidak diperlukan lagi chipset wifi tambahan. NodeMCU banyak digunakan pada lingkungan IoT dan dapat berkomunikasi dengan protokol MQTT. NodeMCU adalah mikrokontroler 32bit, dan sangat mirip dengan arduino dimana memiliki input output ports built-in, nodemcu juga kompatibel dengan Arduino *Integrated Development Environment* (IDE). NodeMCU *firmware* IDE untuk pemrograman lua yang dapat berjalan pada berbagai modul berbasis esp8266 yang memiliki *flash-memory* minimal sebesar 512Kb (Saikhrisna, dkk. 2017).

Gambar 2.10 NodeMCU v2 (ESP8266)

Sumber : Warda, A. (2017)

NodeMCU membutuhkan daya sekitar 3.3v dan dilengkapi dengan prosesor, memori dan *General-Purpose Input/Output* (GPIO) serta memiliki 3 mode wifi yaitu *station, access point* dan keduanya (Warda, A. 2017). Desain SOC memungkinkan modul ini untuk berkomunikasi via GPIO serta di saat yang sama terhubung dan berkomunikasi melalui internet. Perangkat ini merupakan perpaduan sempurna antara internet dan things sesuai dengan konsep IoT (Jaffe, S., 2016).