BAB 1 LANDASAN KEPUSTAKAAN

Pada penelitian ini, pustaka sebagai referensi berasal dari beberapa jurnal yang terkait dengan Internet of Things secara umum, MITM, dan E2E IoT *security middleware* yang digunakan pada IoT.

1.1 Tinjauan Pustaka

Tinjauan pustaka digunakan sebagai acuan dalam pengerjaan penelitian sehingga bisa dibandingkan dengan penelitian sebelumnya. Beberapa jurnal yang dijadikan perbandingan dengan penelitian ini tersaji dalam tabel 2.1.

Tabel 1.1 Tinjauan pustaka

No	Judul Jurnal	Perbedaan	
		Penelitian Terdahulu	Rencana Penelitian
1	Luigi Atzori [2010] *The Internet of Things: A Survey*. University of Cagliari, Italy	Melakukan survey terkait dengan *Internet of Things.*	Melakukan analisis mekanisme keamanan TLS dan crypto pada pengiriman data antara node sensor dengan IoT middleware.
2	Stricot-Tarboton Shaun [2016] *Taxonomy of Man-in-the-Middle Attacks on HTTPS*. University of Waikato, New Zealand	Mengklasifikasikan dan pencegahan serangan MITM pada HTTPS	Melakukan analisis mekanisme keamanan TLS dan crypto pada pengiriman data antara node sensor dengan IoT middleware.
3	Mukherjee Bidyut [2017] *End-to-End IoT Security Middleware for Cloud-Fog Communication*. University of Missouri-Columbia, USA	Mendesain dan mengimplementasi middleware	Melakukan analisis mekanisme keamanan TLS dan crypto pada pengiriman data antara node sensor dengan IoT middleware.

Setelah mengetahui beberapa penelitian yang telah dilakukan terkait penelitian ini. Maka penelitian ini akan fokus pada analisis mekanisme *end-to-end security* komunikasi antara node sensor dengan IoT *middleware*.

1.2 Dasar Teori

Dasar teori berisi tentang beberapa penjelasan definisi tentang IoT *middleware*, *End-to-end encryption*, MQTT, CoAP, TLS, *Crypto*, pengujian pengiriman data, dan menghitung *delay*.

1.2.1 Internet of Things Middleware

Internet of Things (IoT) adalah sistem perangkat komputasi yang saling terkait, mesin mekanis dan digital, objek, hewan atau orang-orang yang dilengkapi dengan pengenal unik dan kemampuan untuk mentransfer data melalui jaringan tanpa memerlukan akses manusia ke manusia atau manusia. ke-komputer interaksi.

Sebuah *Things*, di *Internet of Things*, dapat dilakukan dengan monitor-monitor jantung, hewan ternak dengan *transponder biochip*, sebuah mobil yang memiliki sensor *built-in* untuk mengingatkan pengemudi saat tekanan ban rendah-atau sifat alami lainnya. atau objek buatan manusia yang bisa diberi alamat IP dan dilengkapi dengan kemampuan untuk mentransfer data melalui jaringan.

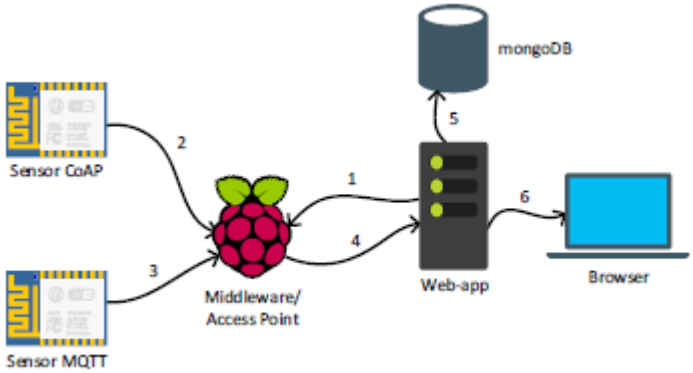
IoT telah berevolusi dari konvergensi teknologi nirkabel, sistem *micro-electromechanical* (MEMS), *microservices* dan internet. Konvergensi ini telah membantu merobohkan dinding silo antara teknologi operasional (PL) dan teknologi informasi (TI), sehingga data mesin yang tidak terstruktur untuk dianalisis untuk wawasan yang akan mendorong perbaikan.

Kevin Ashton, salah satu pendiri dan direktur eksekutif Auto-ID Center di MIT, pertama kali menyebutkan Internet of Things dalam sebuah presentasi yang dia sampaikan kepada Procter & Gamble pada tahun 1999.

Aplikasi praktis teknologi IoT dapat ditemukan di banyak industri saat ini, termasuk pertanian presisi, manajemen bangunan, perawatan kesehatan, energi dan transportasi. Pilihan konektivitas untuk insinyur elektronik dan pengembang aplikasi yang mengerjakan produk dan sistem untuk Internet Hal termasuk:

Meski konsepnya tidak disebutkan sampai tahun 1999, Internet of Things telah berkembang selama beberapa dekade. Alat internet pertama, misalnya, adalah mesin Coke di Carnegie Melon University di awal tahun 1980an. Para pemrogram dapat terhubung ke mesin melalui internet, memeriksa status mesin dan menentukan apakah akan ada minuman dingin yang menunggunya, jika mereka memutuskan untuk melakukan perjalanan ke mesin.

Middeware adalah sebagai antarmuka antara lapisan perangkat keras dan lapisan aplikasi, yang bertanggung jawab untuk berinteraksi dengan perangkat dan manajemen informasi. Peran middleware adalah untuk menyajikan sebuah model pemrograman terpadu untuk berinteraksi dengan perangkat. Sebuah middleware bertugas menutupi masalah heterogenitas dan distribusi yang kita hadapi saat berinteraksi dengan perangkat.



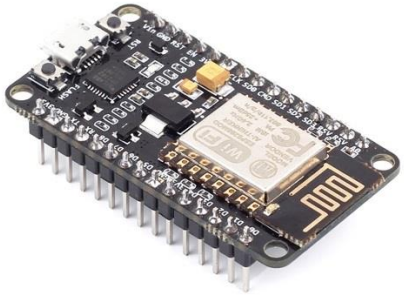
Gambar 1.1 IoT middleware

Sumber: (Anwari, 2017)

Kebutuhan middleware untuk mendukung perangkat IoT terdiri dari dua bagian berdasarkan karakteristik dari infrastruturnya yaitu middleware service requirements dan architectural Requirements (Abdur, 2016)

1.2.2 NodeMCU (ESP8266)

ESP8266 adalah sebuah *open-source firmware* dan alat pengembangan yang membantu untuk membuat prototipe produk IoT dalam beberapa baris skrip Lua. ESP8266 merupakan modul *wireless* berbasis IP untuk *embedded system*. Biasanya ESP8266 pada penerapannya sering kali berintergrasi dengan mikrokoumputer atau mikrokontroller lain seperti Raspberry dan Arduino.



Gambar 1.2 NodeMCU ESP8266

Sumber: (ktechnics.com, 2018)

ESP8266 memiliki berbagai macam fitur seperti wifi dengan biaya yang murah dan perangkat yang berukuran kecil. Perangkat ini mendukung komunikasi UART berbasis IP menggunakan PIN TX/RX. Bahasa pemrograman default dari ESP8266 adalah *AT-Command* atau Lua *Script* menggunakan *firmware* nodemcu, tetapi bisa juga dengan *firmware* Arduino.

Power yang dibutuhkan sangat kecil untuk ESP8266 bisa berjalan, sekitar 50 mA sampai 170 mA sehingga bisa menggunakan USB *power supply* atau baterai. ESP266 ini mendukung 802.11 b/g/n protokol, *Wi-Fi Direct* (P2P), soft AP, dan juga terintegrasi protokol TCP.

1.2.3 MQTT

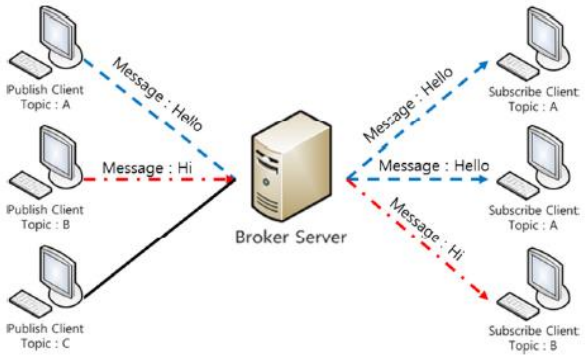
MQTT (*Message Query Telemetry Transport*) adalah protokol terbuka yang dirancang oleh IBM, pada awalnya ditujukan untuk jaringan yang tidak dapat diandalkan dengan sumber daya terbatas seperti *bandwidth* rendah dan *high-latency*. MQTT terdiri dari satu broker server dan dua macam klien yang disebut *publisher* (*Publish client*) dan *Subscriber* (*Subscribe client*). *Broker* server bertindak sebagai perantara pesan yang dikirim antara Publikasikan klien dan langganan klien untuk topik yang menarik. Saat klien *Publish* mengeluarkan topik dan mengirim pesan ke *Broker* server, klien Langganan memilih topik yang itu menarik (Lee, 2013).

Di MQTT, topik bersifat hirarkis, seperti sistem pengarsipan (misalnya dapur / oven / suhu). Wildcard diperbolehkan saat mendaftarkan langganan (tapi tidak saat menerbitkan) yang memungkinkan keseluruhan hierarki diperhatikan oleh klien. The wildcard + cocok dengan satu nama direktori tunggal, # cocok dengan sejumlah direktori dengan nama apapun. Misalnya, topik dapur/+/suhu cocok dengan dapur/foo/suhu tapi bukan dapur/foo/ bar/suhu. dapur/# cocok dengan dapur/lemari es/kompresor/katup1/suhu.

Klien MQTT dapat mendaftarkan "pesan terakhir" untuk dikirim oleh broker jika mereka memutuskan hubungan. Pesan ini dapat digunakan untuk memberi sinyal kepada pelanggan

saat perangkat terputus. MQTT memiliki dukungan untuk pesan persisten yang tersimpan di broker. Saat mempublikasikan pesan, klien mungkin meminta agar pialang tetap menerima pesan tersebut. Hanya pesan terus-menerus terbaru disimpan. Bila klien berlangganan sebuah topik, pesan yang terus berlanjut akan dikirim ke klien.Tidak seperti antrian pesan, broker MQTT tidak mengizinkan pesan yang terus berlanjut untuk dicadangkan di dalam server.

Broker MQTT mungkin memerlukan otentikasi username dan password dari klien untuk terhubung. Untuk memastikan privasi, koneksi TCP dapat dienkripsi dengan SSL/TLS.



Gambar 1.3 MQTT proses transmisi pesan

Sumber: (Lee, 2013)

Sebagai contoh, *Facebook Messenger* didasarkan pada MQTT, dan banyak proyek open source lainnya sedang dalam perjalanan MQTT. Protokol MQTT cocok untuk diimplementasikan terintegrasi *gateway* SNS server yang dapat menggabungkan berbeda Protokol SNS dan OS menjadi satu *platform* terpadu. Juga, Tidak ada pembatasan dalam pesan saat menggunakan *push* layanan pemberitahuan Untuk memastikan keandalan pesan, MQTT mendukung 3 Tingkat Kualitas Layanan (QoS). Gambar 1 menunjukkan paket ukuran tukar menurut 3 level QoS yang berbeda. QoS Level 0 hanya mengirim pesan setelah mengikuti pesan aliran distribusi, dan tidak mengecek apakah pesannya tiba ke tempat tujuan Oleh karena itu, dalam kasus pesan yang cukup besar, Mungkin saja pesan itu akan hilang bila ada Kehilangan datang di jalan. QoS Level 1 paling sedikit mengirim pesan sekali, dan cek status pengiriman pesan dengan enggunakan pesan cek status, PUBACK. Namun, kapan PUBACK hilang, ada emungkinan server akan mengirim pesan yang sama dua kali, karena tidak ada onfirmasi dari pesan sedang dikirim QoS Level 2 melewati pesan melalui tepat sekali memanfaatkan jabat tangan 4 arah. Bukan itu mungkin ada kehilangan pesan di level ini, namun karena adanya Proses rumit jabat tangan 4 arah, adalah mungkin untuk memilikinya keterlambatan end-to-end yang relatif lebih lama (Lee, 2013).

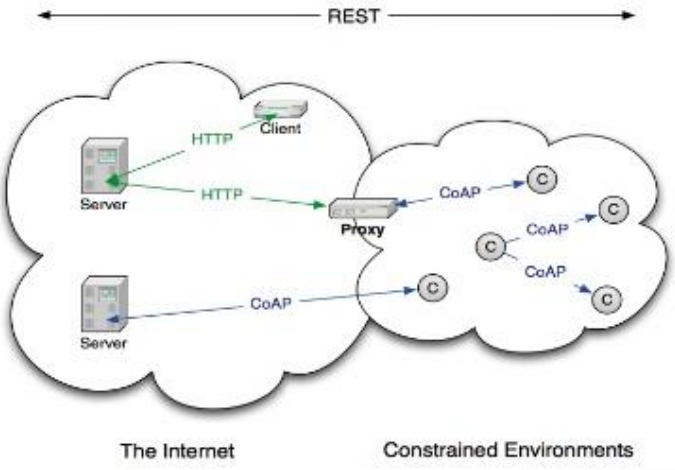
Gambar 1.4 Metode transmisi paket berdasar QoS

Sumber: (Lee, 2013)

Tingkat QoS yang lebih tinggi adalah semakin banyak paket yang akan ditukar. Memang benar bahwa QoS tingkat tinggi lebih efektif jika tidak Ingin kehilangan pesan, tapi proses rumit semacam itu akan terjadi meningkatkan keterlambatan end-to-end. Jika bisa kita deduksi sesuai Tingkat QoS memanfaatkan analisis korelasi hubungan antara kehilangan pesan karena ukuran payload dan Penundaan end-to-end, memungkinkan membangun layanan yang optimal jaringan untuk layanan notifikasi push (Lee, 2013).

1.2.4 CoAP

Lapisan aplikasi protokol CoAP awalnya dikembangkan untuk transfer web dengan node dan jaringan yang dibatasi di IoT. Protokolnya adalah versi HTTP yang dibuat ulang sesuai dengan persyaratan IoT untuk *overhead* rendah dan dukungan *multi-cast*. CoAP bergantung pada REST, sebuah prinsip diadopsi dari HTTP dan disematkan di UDP untuk transaksi. Alasan awal untuk mengembangkan protokol ini adalah mencocokkan Tingginya kebutuhan IoT dan kebutuhan akan tarif rendah dan protokol yang ringan.



Gambar 1.5 Arsitektur CoAP

Sumber: (ARM, 2014)

Secara keseluruhan, fitur utama CoAP adalah: Ini mendukung persyaratan M2M di lingkungan yang terbatas, UDP mengikat dengan opsional pendukung *uni-cast* dan *multicast* permintaan, pertukaran pesan asinkron, header rendah *overhead* dan parsing kompleksitas,

mendukung URI *(Universal Resource Identifier)* dan tipe konten, dan ia punya kemampuan *proxy* dan *caching* sederhana (Reem Abdul, 2016).

CoAP berjalan di atas UDP, bukan TCP. Klien dan server berkomunikasi melalui datagram tanpa koneksi. Retries dan penataan ulang diimplementasikan dalam aplikasi stack. Menghapus kebutuhan akan TCP memungkinkan jaringan IP penuh dalam mikrokontroler kecil. CoAP memungkinkan siaran UDP dan multicast digunakan untuk pengalamatan. CoAP mengikuti model *client / server*. Klien mengajukan permintaan ke server, server mengirim kembali tanggapan. Klien dapat memperoleh sumber daya *GET, PUT, POST dan DELETE.*

CoAP dirancang untuk beroperasi dengan HTTP dan web yang tenang secara luas melalui proxy sederhana. Karena CoAP berbasis datagram, ini bisa digunakan di atas SMS dan protokol komunikasi berbasis paket lainnya. Permintaan dan pesan tanggapan dapat ditandai sebagai "confirmable" atau "nonconfirmable". Pesan yang dapat dikonfirmasikan harus diakui oleh penerima dengan paket ack. Pesan yang tidak dapat dikonfirmasikan adalah "fire dan forget".

Seperti HTTP, CoAP mendukung negosiasi konten. Klien menggunakan opsi Terima untuk mengungkapkan representasi pilihan sumber daya dan server yang membalas dengan opsi Content-Type untuk memberi tahu klien tentang apa yang mereka dapatkan. Seperti HTTP, ini memungkinkan klien dan server berkembang secara independen, menambahkan representasi baru tanpa mempengaruhi satu sama lain. Permintaan CoAP dapat menggunakan string kueri dalam bentuk? A = b & c = d. Ini bisa digunakan untuk memberikan pencarian, paging dan fitur lainnya kepada klien.

Karena CoAP dibangun di atas UDP bukan TCP, SSL / TLS tidak tersedia untuk memberikan keamanan. DTLS, Datagram Transport Layer Security memberikan jaminan yang sama seperti TLS namun untuk transfer data melalui UDP. Biasanya, perangkat CoAP yang kompatibel dengan DTLS akan mendukung RSA dan AES atau ECC dan AES.

CoAP memperluas model permintaan HTTP dengan kemampuan untuk mengamati sumber daya. Bila flag pengamatan diatur pada permintaan CoAP GET, server dapat terus membalas setelah dokumen awal telah ditransfer. Hal ini memungkinkan server untuk melakukan perubahan status ke klien saat terjadi. Entah akhirnya bisa membatalkan pengamatan.

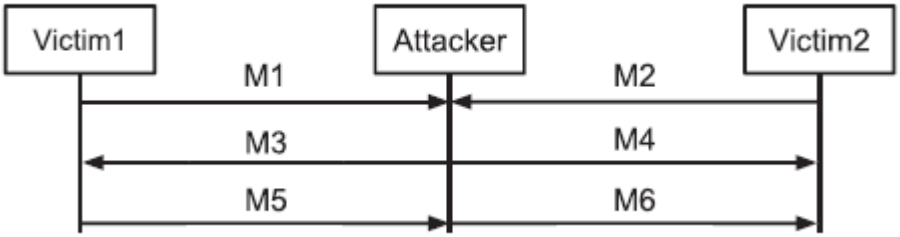
CoAP mendefinisikan mekanisme standar untuk penemuan sumber daya. Server menyediakan daftar sumber daya mereka (bersama dengan metadata tentang mereka) di /.well- known/core. Tautan ini ada dalam tipe media aplikasi / link-format dan memungkinkan klien menemukan sumber daya apa yang disediakan dan jenis media apa mereka.

Di CoAP, node sensor biasanya server, bukan klien (meski mungkin keduanya). Sensor (atau aktuator) menyediakan sumber daya yang dapat diakses oleh klien untuk membaca atau mengubah keadaan sensor.

Sebagai sensor CoAP adalah server, mereka harus dapat menerima paket inbound. Untuk berfungsi dengan baik di belakang NAT, perangkat pertama-tama dapat mengirimkan permintaan ke server, seperti yang dilakukan di LWM2M, yang memungkinkan penerus untuk menghubungkan keduanya. Meskipun CoAP tidak memerlukan IPv6, ini paling mudah digunakan di lingkungan IP di mana perangkat langsung routable.

1.2.5 MITM

Serangan Man-In-The-Middle (MITM) adalah jenis serangan cyber security umum yang memungkinkan penyerang untuk menguping komunikasi antara dua target. Serangan tersebut terjadi antara dua host yang berkomunikasi secara sah yang memungkinkan penyerang untuk mendengarkan percakapan yang seharusnya tidak dapat mereka dengarkan, maka namanya "man-in-the-middle" (www.rapid7.com, 2017)



Gambar 1.6 Analogi MITM

Dalam serangan MITM, skenario umum melibatkan: dua titik akhir (korban), dan pihak ketiga (penyerang). Penyerang memiliki akses pada saluran komunikasi antara dua titik akhir, dan bisa memanipulasi pesan mereka. Serangan MITM bisa terjadi divisualisasikan seperti yang ditunjukkan pada Gambar 2.3. Secara khusus, korban mencoba menginisialisasikan komunikasi yang aman dengan saling mengirim berita kunci (pesan M1 dan M2). Penyerang mencegat M1 dan M2, dan saat kembali mengirimkan kunci publiknya kepada para korban (pesan M3 dan M4). Setelah itu, korban mengenkripsi pesannya oleh penyerang kunci publik, dan mengirimkannya ke korban2 (pesan M5). Penyerang mencegat M5, dan mendekripsi menggunakan kunci privat yang diketahui. Kemudian, Penyerang mengenkripsi plaintext dengan kunci publik korban2, dan mengirimkannya untuk korban2 (pesan M6) (Conti, 2016).

1.2.5.1 Tipe MITM

- **Rogue Access Point**

Perangkat yang dilengkapi dengan kartu wireless ini seringkali akan mencoba menyambung otomatis ke access point yang memancarkan sinyal terkuat. Penyerang dapat membuat titik akses nirkabel dan trik di sekitar perangkat mereka untuk bergabung dengan domainnya. Semua lalu lintas jaringan korban sekarang dapat dimanipulasi oleh penyerang. Ini berbahaya karena penyerang bahkan tidak harus berada di jaringan terpercaya untuk melakukan ini-penyerang hanya memerlukan kedekatan fisik yang cukup dekat.

- **ARP Spoofing**

ARP adalah Address Resolution Protocol. Ini digunakan untuk menyelesaikan alamat IP ke alamat MAC (media access control) fisik di jaringan area lokal. Ketika sebuah host perlu berbicara dengan host dengan alamat IP tertentu, ia merujuk pada cache ARP untuk menyelesaikan alamat IP ke alamat MAC. Jika alamat tidak diketahui, permintaan diajukan untuk meminta alamat MAC perangkat dengan alamat IP.

Penyerang yang ingin berpose sebagai host lain dapat menanggapi permintaan yang seharusnya tidak merespons dengan alamat MAC-nya sendiri. Dengan beberapa paket yang tepat ditempatkan, penyerang bisa mengendus lalu lintas pribadi antara dua host. Informasi

berharga dapat diekstraksi dari lalu lintas, seperti pertukaran token sesi, yang menghasilkan akses penuh ke akun aplikasi yang seharusnya tidak dapat diakses oleh penyerang.

- **mDNS Spoofing**

Multicast DNS mirip dengan DNS, tapi dilakukan pada jaringan area lokal (LAN) dengan menggunakan siaran seperti ARP. Hal ini menjadikannya target yang sempurna untuk serangan spoofing. Sistem resolusi nama lokal seharusnya membuat konfigurasi perangkat jaringan sangat sederhana. Pengguna tidak perlu tahu secara pasti alamat mana perangkat mereka harus berkomunikasi dengan; mereka membiarkan sistem mengatasinya untuk mereka. Perangkat seperti TV, printer, dan sistem hiburan menggunakan protokol ini karena mereka biasanya berada di jaringan tepercaya. Ketika sebuah aplikasi perlu mengetahui alamat perangkat tertentu, seperti tv.local, penyerang dapat dengan mudah menanggapi permintaan tersebut dengan data palsu, menginstruksikannya untuk menyelesaikan ke alamat yang dikuasainya. Karena perangkat menyimpan cache alamat lokal, korban sekarang akan melihat perangkat penyerang sebagai terpercaya selama kurun waktu tertentu.

- **DNS Spoofing**

Serupa dengan cara ARP menyelesaikan alamat IP ke alamat MAC di LAN, DNS menyelesaikan nama domain ke alamat IP. Saat menggunakan serangan spoofing DNS, penyerang mencoba mengenalkan informasi cache DNS korup ke host dalam upaya untuk mengakses host lain menggunakan nama domain mereka, seperti www.onlinebanking.com. Hal ini menyebabkan korban mengirimkan informasi sensitif ke host jahat, dengan keyakinan mereka mengirimkan informasi ke sumber yang terpercaya. Penyerang yang telah memalsukan alamat IP bisa memiliki waktu yang jauh lebih mudah untuk menipu DNS hanya dengan menyelesaikan alamat server DNS ke alamat penyerang.

1.2.5.2 Teknik MITM

- **Sniffing**

Penyerang menggunakan alat pengambilan paket untuk memeriksa paket pada tingkat rendah. Menggunakan perangkat nirkabel tertentu yang diizinkan untuk dimasukkan ke dalam mode pemantauan atau promiscuous dapat memungkinkan penyerang untuk melihat paket yang tidak dimaksudkan untuk dilihat, seperti paket yang ditujukan ke host lain.

- **Packet Injection**

Penyerang juga dapat memanfaatkan mode pemantauan perangkat mereka untuk menyuntikkan paket berbahaya ke arus komunikasi data. Paket dapat berbaur dengan arus komunikasi data yang valid, yang tampaknya merupakan bagian dari komunikasi, namun berbahaya. Injeksi paket biasanya melibatkan sniffing pertama untuk menentukan bagaimana dan kapan harus membuat dan mengirim paket.

- **Session Hijacking**

Sebagian besar aplikasi web menggunakan mekanisme login yang menghasilkan token sesi sementara yang akan digunakan untuk permintaan di masa depan agar pengguna tidak mengetikkan kata sandi di setiap halaman. Seorang penyerang bisa mengendus lalu lintas sensitif untuk mengidentifikasi token sesi bagi pengguna dan menggunakannya untuk

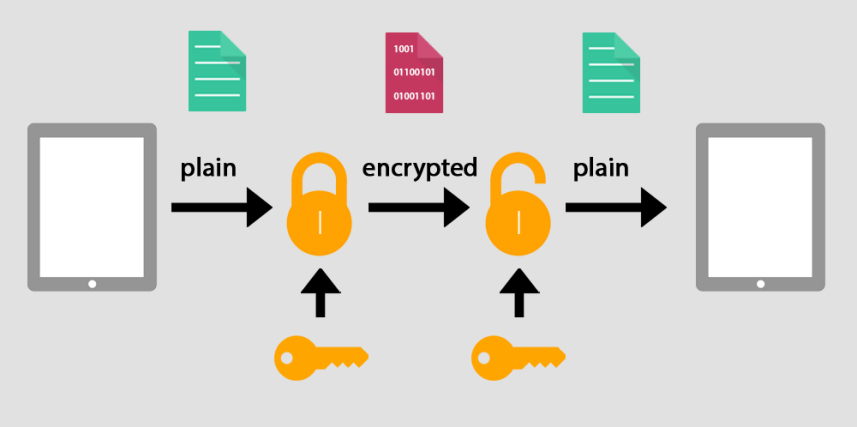
membuat permintaan sebagai pengguna. Penyerang tidak perlu menipu begitu dia memiliki token sesi.

- **SSL Stripping**

Karena menggunakan HTTPS adalah perlindungan umum terhadap spoofing ARP atau DNS, penyerang menggunakan pengupasan SSL untuk mencegat paket dan mengubah permintaan alamat berbasis HTTPS mereka untuk mencapai titik akhir HTTP yang setara, memaksa host untuk mengajukan permintaan ke server yang tidak dienkripsi. Informasi sensitif bisa bocor dalam teks biasa.

1.2.6 E2EE

End to End Encryption (E2EE) adalah salah satu cara yang paling umum digunakan untuk mengirimkan informasi dengan aman ke seluruh internet. Pada prinsipnya, E2EE adalah cara untuk mengirim informasi melalui jaringan sedemikian rupa sehingga hanya penerima dan pengirim yang bisa mengaksesnya. Menurut Lead Engineer di Cisco Systems, Michael Behringer, E2EE berisi komponen berikut: identitas dan protokol, algoritma, implementasi yang aman, dan operasi yang aman. Komponen identitas dan protokol bertindak sebagai satpam untuk memeriksa verifikasi bahwa pihak-pihak yang terlibat adalah mereka yang mereka katakan. Protokol digunakan untuk menyiapkan semua yang dibutuhkan untuk E2EE, seperti pertukaran kunci dan algoritma. Algoritma menggunakan proses matematika untuk mengacak data sedemikian rupa sehingga hampir tidak mungkin untuk menguraikan tanpa kunci yang telah ditentukan.



Gambar 1.7 Mekanisme E2EE

Implementasi dan pengoperasian yang aman diperlukan untuk memastikan bahwa proses E2EE tidak rentan terhadap serangan cyber di sisi perangkat keras, seperti virus dan malware. Komponen ini semua bekerja sama untuk menyediakan sistem cairan yang berjalan efisien untuk memberikan keamanan terbaik bagi pengguna akhir. Ini membantu E2EE menyesuaikan diri dengan berbagai ancaman yang ditimbulkan oleh hacker, karena komponen induvial dapat berkembang untuk mengatasi kelemahan induvial, daripada harus menemukan kembali keseluruhan sistem (Erik Wehner, 2016).

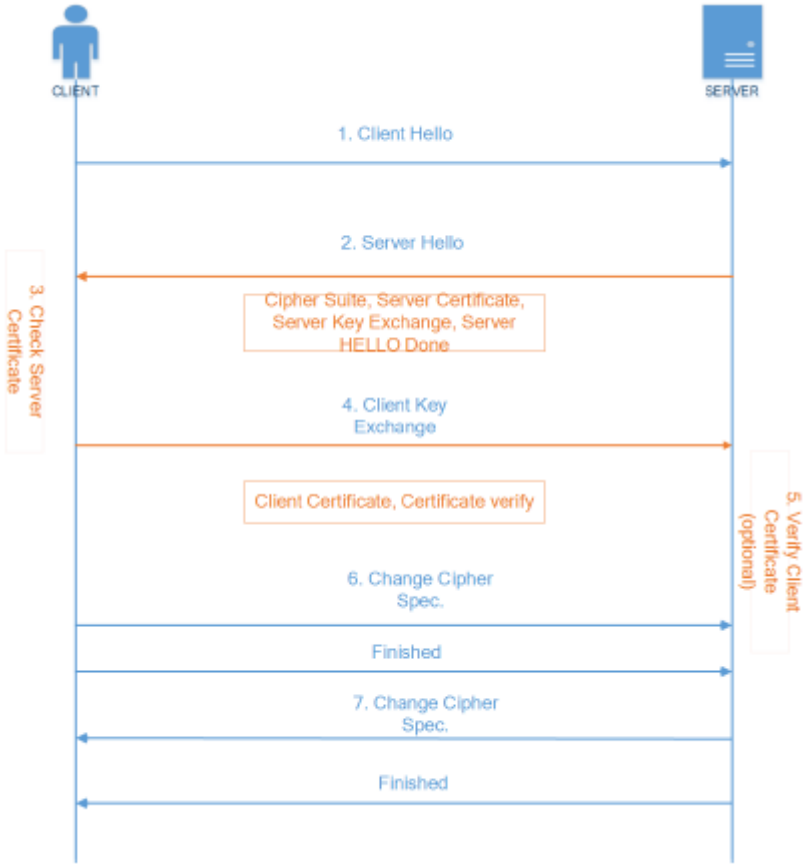
1.2.7 TLS

Transport Layer Security (TLS) dimaksudkan untuk memberikan keamanan saluran antara dua entitas berkomunikasi melalui internet. Entitas menggunakan sertifikat X.509 untuk

pertukaran kunci publik. Kunci publik lebih baik digunakan untuk otentikasi. Kemudian digunakan untuk menukar kunci sesi karena seperti yang kita ketahui publik kriptografi kunci tidak dapat digunakan secara langsung untuk enkripsi dan dekripsi (karena perhitungan overhead) jadi ada kebutuhan kunci simetris atau kunci sesi di TLS. Jadi kunci sesi adalah digunakan untuk enkripsi data yang mengalir antara berkomunikasi entitas. Ini menyediakan kerahasiaan data dan juga menggunakan MAC (kode otentikasi pesan) untuk otentikasi pesan. Di ringkasan, untuk menghindari penyadapan dan pemalsuan pesan yang kita gunakan protokol ini (Ranjan, 2014).

Protokol TLS terdiri dari empat subprotocols Handshake, Subprotokol ChangeCipherSpec, Record, dan Alert. Klien dan server saling mengotentikasi satu sama lain sertifikat. Ini disediakan oleh pihak ketiga yang terpercaya dan telah terinstal di browser klien Selama proses jabat tangan, server mengirim sertifikat dan kemudian membandingkan browser klien dengan yang terinstal satu. Namun dengan prinsip otentikasi klien Juga harus diperiksa, tapi itu adalah fitur opsional. Ini adalah bagaimana otentikasi terjadi di TLS (Ranjan, 2014).

TLS Handshake beroperasi pada lapisan sesi untuk menetapkan parameter kriptografi dan kunci sesi maka enkripsi adalah dilakukan pada lapisan presentasi menggunakan cipher simetris dan kunci sesi TLS / SSL bekerja atas nama yang mendasari lapisan transport Di Internet Protocol Suite, enkripsi data sedang dilakukan oleh TLS / SSL di layer aplikasi. Di Model OSI, TLS / SSL diinisialisasi pada lapisan sesi dan karya pada lapisan presentasi (Ranjan, 2014).



Gambar 1.8 TLS handshake

Sumber: Ranjan (2014)

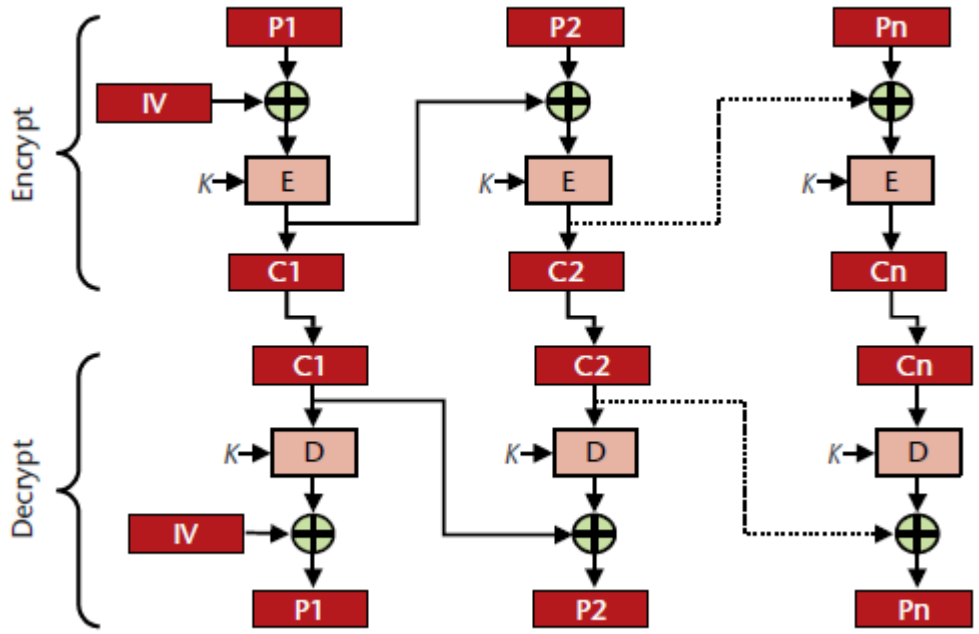
Protokol ini memungkinkan server dan client melakukan otentikasi satu sama lain dan untuk menegosiasikan enkripsi dan algoritma MAC dan kunci sesi untuk mengenkripsi data dalam catatan TLS. Jabat tangan adalah dilakukan sebelum data aplikasi dikirim (Ranjan, 2014).

1.2.8 Crypto

Kriptografi adalah sebuah metode menyimpan dan mengirim data di dalam sebuah bentuk sehingga hanya untuk siapa yang dimaksudkan yang bisa membaca dan memprosesnya. Kriptografi modern fokus pada empat tujuan (searchsoftwarequality.techtarget.com, 2014):

- Confidentiality, informasi tidak bisa dimengerti oleh siapa pun tanpa disengaja.
- Integrity, informasi tidak dapat diubah dalam penyimpanan atau transit antara pengirim dan penerima yang dituju tanpa adanya perubahan yang terdeteksi.
- Non-repudiation, pencipta/pengirim informasi tidak dapat menyangkal pada tahap selanjutnya, maksudnya dalam tahap penciptaan atau pengiriman informasi.
- Authentication, pengirim dan penerima bisa saling mengkonfirmasi identitas dan asal/tujuan dari informasi.

Ersam, Meyer, Smith dan Tuchman menemukan mode operasi Cipher Block Chaining (CBC) pada tahun 1976. CBC mode, diilustrasikan pada gambar, adalah mode di mana ciphertext dari blok i di XOR dengan plaintext dari blok i + 1 sebelum dienkripsi. Blok pertama adalah Di XOR dengan vektor inisialisasi (IV) yang tidak perlu dirahasiakan Modus CBC menyembunyikan pola data di ciphertext dan lebih aman dari mode ECB. Satu Masalah dengan mode CBC, meskipun, adalah sangat ketat blok serial, jadi blok saya harus dienkripsi sebelum blok i + 1, membatasi kemampuannya untuk melakukan banyak operasi sejajar. Kesalahan bit tunggal dalam saluran komunikasi biasanya menghasilkan dua blok penuh plaintext korup (Burr, 2003).

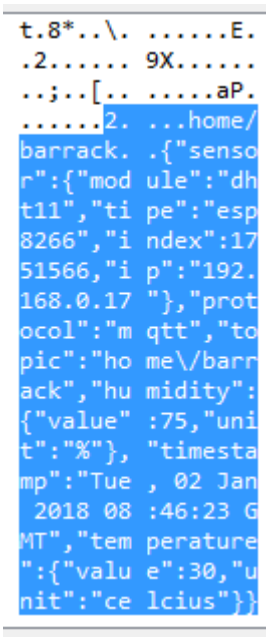


Gambar 1.9 Block cipher enkripsi pada mode CBC

Sumber: Burr (2003)

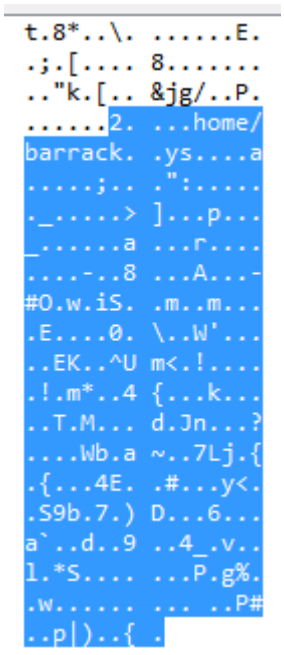
1.2.9 Pengujian Pengiriman Data

Pengujian dilakukan dengan tujuan untuk mengetahui metode mekanisme keamanan yang ditambahkan berhasil atau tidak dalam pengiriman data. Pengujian dilakukan untuk melihat perbedaan dari sebelum dan sesudah metode mekanisme keamanan ditambahkan. Sniffing adalah salah satu metode yang digunakan untuk melihat isi dari paket pengiriman data pada sistem (Ramdhansya, 2014). Program yang digunakan untuk meng-capture paket data adalah tcpdump.



Gambar 1.10 Data paket tanpa keamanan

Pada gambar 2.6 bisa terlihat bahwa data yang dikirim masih bisa dibaca dengan mudah. Karena pengiriman data tanpa mekanisme keamanan



Gambar 1.11 Data paket dengan keamanan

Pada gambar 2.7 bisa terlihat bahwa paket data tidak bisa terbaca karena telah dienkripsi dengan AES-CBC.

1.2.10 Pengolahan Data

Pengolahan data dilakukan setelah pengujian dan pengambilan data. Hal ini dilakukan untuk mendapatkan hasil analisis dari penelitian ini.

1.2.10.1 Packet Loss

Packet loss adalah salah satu parameter pengukuran baik buruknya jaringan. Packet loss adalah paket yang hilang atau gagal terkirim ke tujuan. Semakin besar nilai sebuah packet loss maka semakin buruk kualitas jaringan tersebut dalam pengiriman data. Nilai packet loss didapat dari rumus berikut:

Nilai expected didapat dari rumus: $\frac{lama\ pengujian(detik)}{30\ detik}$

Nilai actual didapat dari jumlah publish message pada MQTT atau CON pada CoAP

Nilai *success rate* didapat dengan rumus: $\frac{actual\ packet}{expected\ packet} \times 100\%$

Nilai *packet loss rate* didapat dengan rumus: $\frac{expected - actual}{expected} \times 100\%$

Nilai *expected* adalah nilai yang diharapkan atau dugaan banyak nilai paket terkirim sampai pada tujuan. Nilai *actual* adalah nilai dari paket yang berhasil terkirim ke tujuan.

1.2.10.2 Delay

Delay adalah satu parameter yang sering digunakan pada jaringan sebagai quality of service (QoS). Delay bisa dijadikan sebuah acuan dari baik buruknya sebuah jaringan. Banyak hal yang bisa mempengaruhi *delay*. Nilai *delay* bisa didapat dari aplikasi wireshark. Nilai *delay* adalah nilai *Delta time* pada wireshark. Nilai *delta time* adalah waktu jeda paket dikirim dari sumber sampai dengan waktu pengiriman ACK dari tujuan. Kemudian dicari nilai rata-rata dari *Delta time* tersebut atau rata-rata dari *delay*.

No.	Time	Source	Destination	Info	Delta
8	1.054041	192.168.0.17	192.168.0.1	Connect Command	0.018199
10	1.061723	192.168.0.1	192.168.0.17	Connect Ack	0.007399
50	31.141087	192.168.0.17	192.168.0.1	Publish Message	0.025886
51	31.146947	192.168.0.1	192.168.0.17	Publish Ack	0.005860
95	61.144327	192.168.0.17	192.168.0.1	Publish Message	0.030688
97	61.168379	192.168.0.1	192.168.0.17	Publish Ack	0.019587
128	91.148381	192.168.0.17	192.168.0.1	Publish Message	0.026223
129	91.152779	192.168.0.1	192.168.0.17	Publish Ack	0.004398
240	121.154261	192.168.0.17	192.168.0.1	Publish Message	0.038869
241	121.158293	192.168.0.1	192.168.0.17	Publish Ack	0.004032
319	181.210515	192.168.0.17	192.168.0.1	Publish Message	0.080873

Gambar 1.12 Delta time

1.2.10.3 Jitter

Jitter adalah nilai dari variasi *delay* pengiriman paket. Penyebab dari *jitter* adalah antrian dari sebuah pengiriman data dan *reassemble* pada akhir pengiriman. Nilai *jitter* didapat dengan rumus berikut:

$sum(variasi\ delay)/(jumlah\ data - 1)$

Total dari variasi delay didapat dengan rumus:

$$sum(delay2 - delay1) + (delay3 - delay\ 2) \ldots . (delayn - delay(n-1))$$