# *Penerapan mTLS untuk Keamanan Sistem*

Studi Kasus pada Interkoneksi Layanan dan Autentikasi Sistem

Matthew Kevin Amadeus - 13518035
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: mkamadeus.mka@gmail.com, 13518035@std.stei.itb.ac.id

***Abstrak*—Penggunaan TLS sudah meluas dalam pengembangan sistem, baik dalam koneksi antar layanan maupun dalam membuat koneksi dari *client* ke *server*. Dalam makalah ini, akan dibahas terkait bagaimana mTLS dapat digunakan dalam kedua konteks tersebut dengan tujuan untuk meningkatkan keamanan dari sistem.**

***Kata Kunci—mutual TLS, keamanan layanan***

## I. Pendahuluan

Di masa kini, keamanan dalam sistem untuk menjaga data menjadi salah satu topik yang diperhatikan dalam proses pengembangan sistem. Selain keamanan di sisi implementasi sistem, keamanan dari sisi pengiriman data juga perlu menjadi perhatian. Hal ini dilakukan untuk mencegah serangan seperti *man-in-the-middle attack*; yaitu serangan yang melakukan intersepsi dalam sistem untuk mendapatkan data dalam proses pengiriman.

Oleh karena itu, telah dikembangkan beberapa teknik untuk melakukan pengamanan terhadap proses pengiriman data. Salah satunya adalah dengan memanfaatkan TLS (*Transport Layer Security*) yang sekarang sudah sering dimanfaatkan dalam browser. Telah dikembangkan protokol lainnya, seperti mTLS yang pada dasarnya adalah TLS dua arah yang aplikasinya mulai meluas.

Dalam makalah ini, akan dibahas terkait penggunaan TLS, khususnya pada mTLS dalam pengembangan sistem. Makalah ini akan berfokus pada kasus untuk bagaimana protokol dapat mengamankan layanan.

## II. Public Key Infrastructure

Public Key Infrastructure (PKI) adalah suatu metode untuk menyebarluaskan sertifikat digital melalui teknik enkripsi yang ada. PKI menggunakan teknik kriptografi asimetrik yang memiliki kunci publik. Pada dasarnya, karena luasnya penggunaan sertifikat digital di Internet, perlu metode untuk mengatur sertifikat tersebut, seperti membuat, menyimpan, memverifikasi, dan membuang sertifikat.

Terdapat beberapa komponen penting dalam PKI yang akan menjadi bahan pembahasan dalam, yaitu sebagai berikut:

1. Sertifikat Digital, yaitu suatu bukti dari kepemilikan suatu domain.
2. Certificate Authority (CA), yaitu pihak yang memiliki sertifikat yang dapat digunakan untuk menerbitkan sertifikat baru.
3. Registration Authority (RA), yaitu pihak yang meminta sertifikat.

Secara singkat, CA yang ada akan menerbitkan sertifikat berdasarkan permintaan dari pihak lainnya. Kunci privat yang dimiliki CA akan digunakan untuk memverifikasi sertifikat yang dibuat lewat kunci yang dimiliki CA.

Struktur PKI bila diterapkan akan menyerupai pohon. CA akan menjadi *root* dari pohon, yang akan membuat beberapa *node* baru sebagai sertifikat dan kunci yang dibangkitkan dari CA. Struktur seperti ini dalam PKI menjadi berguna dalam internet, karena dengan ini keaslian sertifikat bisa terjamin dari pihak ketiga.

## III. Transport Layer Security

TLS (Transport Layer Security) adalah suatu protokol kriptografi yang bertujuan untuk mengamankan pesan dalam *transport layer* dalam OSI, dengan tujuan mengamankan pesan yang dikirimkan secara *end-to-end*. Dalam sejarahnya, TLS berasal dari pengembangan lanjut SSL (*Secure Socket Layer*). Seperti namanya, TLS hanya melakukan pengamanan terhadap pengiriman data saja; tidak ada pengamanan untuk sisi sistem atau penerima.

Di masa kini, TLS menjadi hal yang penting karena di era informasi yang banyak dilakukan pengiriman dan penerimaan informasi, tanpa adanya informasi yang terenkripsi pesan tentunya dengan mudah dapat disadap. Tanpa adanya TLS, pesan yang dikirimkan lewat media internet dapat disadap oleh pihak short-term dengan mudah. Pesan tersebut dikirimkan lewat paket-paket internet yang tidak terenkripsi, sehingga dengan menggunakan program-program yang ada di masa kini paket tersebut dapat disadap dengan mudah.

Oleh karena itu, TLS sudah digunakan secara meluas mulai dari browser yang sehari-hari digunakan. Tidak hanya browser, bahkan dalam kumpulan sistem yang saling berkomunikasi TLS juga digunakan sebagai salah satu cara mengamankan dan memastikan bahwa hanya kedua pihak yang terjamin yang dapat mengakses pesan. Secara umum memang TLS diaplikasikan aplikasi atau sistem yang berbasis

pada HTTP, namun pada dasarnya bisa diaplikasikan dalam sistem lainnya juga.

TLS ini berbasis pada kombinasi kriptografi kunci publik yang memiliki dua jenis kunci yang berbeda, yaitu kunci publik dan kunci privat. Terdapat tiga hal utama yang terdapat dalam TLS, yaitu:

- Kunci publik dan kunci privat
- TLS Certificate
- TLS Handshake

Dalam TLS, kunci publik berperan sebagai kunci yang dapat dilihat siapapun yang tujuannya untuk mengenkripsi pesan. Penerima pesan dapat membuktikan bahwa pesan tersebut adalah miliknya dengan melakukan dekripsi terhadap pesan dengan kunci privat yang dimiliki. Dengan konsep awal ini, kerahasiaan pesan bisa terjaga serta hanya yang bersangkutan dengan kunci yang sesuai yang dapat membaca pesan tersebut.

Selanjutnya ada TLS Certificate. TLS Certificate adalah sertifikat digital yang menyimpan data terkait siapa yang membuat sertifikat tersebut. Sertifikat digital ini digunakan untuk memverifikasi identitas dari server atau perangkat, serta juga mencantumkan informasi terkait kapan sertifikat tersebut akan kadaluarsa dan harus diperbaharui lagi.

Terakhir, terdapat TLS Handshake, yaitu skema bagaimana dalam protokol ini dapat dilakukan pertukaran sertifikat dan kunci. Dalam TLS Handshake terdapat verifikasi apakah pihak yang bersangkutan memiliki kunci.

Tentunya, komponen-komponen dalam TLS memiliki tujuannya masing-masing. Kunci dalam TLS berfungsi untuk mengenkripsi pesan yang ada di jaringan. Sertifikat digunakan untuk memverifikasi data, melakukan autentikasi terhadap server. TLS Handshake digunakan untuk memastikan koneksi serta mengirimkan kunci.

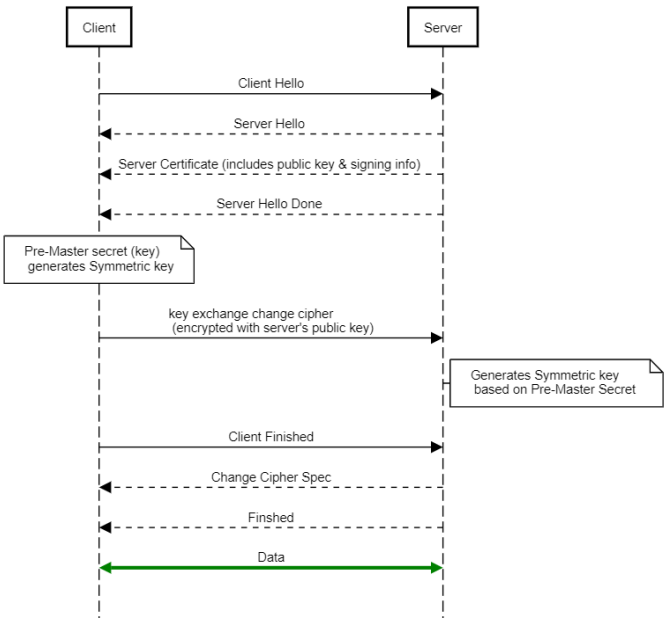
Secara umum, TLS hanya digunakan untuk melakukan pengamanan pada aplikasi web. TLS umum digunakan dalam browser. Salah satu aplikasinya yang paling umum dikenal adalah protokol HTTPS pada layer aplikasi. HTTPS dibangun di atas protokol HTTP yang menggunakan TLS.

## IV. Cara Kerja TLS

Dalam TLS secara umum, hanya pihak server saja yang memiliki sertifikat serta pasangan kunci, di mana pihak yang melakukan *request* tidak perlu memiliki. Secara singkat, terdapat beberapa langkah yang dilakukan dalam melakukan TLS:

1. Client melakukan koneksi ke server untuk meminta sertifikat digital.
2. Server mengembalikan sertifikat digital yang dimilikinya.
3. Client melakukan verifikasi TLS certificate dari server
4. Melalui koneksi yang dibuat, client dan server bisa melakukan pertukaran pesan

Tanpa melakukan tahapan tersebut, server bisa menolak request yang masuk dari client. Oleh karena itu, tahapan di atas wajib dilakukan untuk membuat sebuah koneksi yang terenkripsi.




Gambar 1: Protokol TLS

## V. Mutual Transport Layer Security

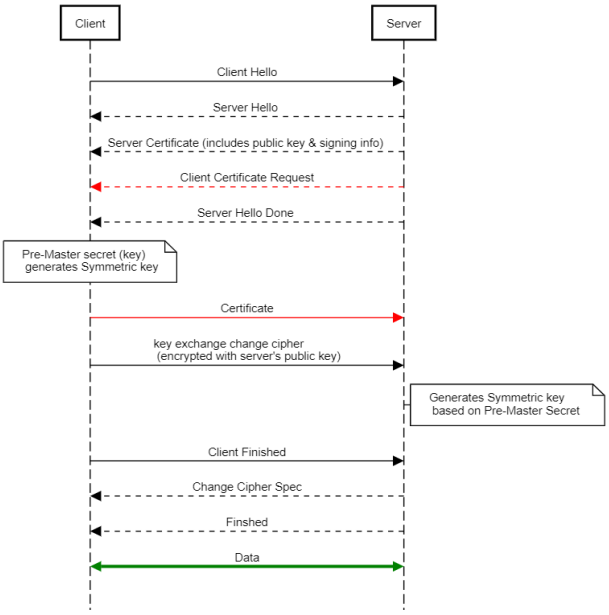
Mutual Transport Layer Security atau yang biasa dikenal dengan mTLS pada dasarnya memanfaatkan protokol yang sama dengan TLS. Terdapat pertukaran kunci dan sertifikat sama seperti pada TLS, namun sesuai dengan namanya, pembeda utama mTLS dengan TLS adalah mTLS berlangsung dua arah dari kedua pihak.

Dalam TLS, hanya server saja yang memiliki sertifikat dan pasangan kunci untuk mengenkripsi pesan. Dalam mTLS, kedua belah pihak harus memiliki sertifikat dan pasangan kunci. Setelah dilakukan verifikasi untuk satu pihak, akan diverifikasi untuk pihak yang lainnya.

Secara protokol, berikut terdapat beberapa tahapan yang dilakukan dalam mTLS:

1. Client melakukan koneksi ke server untuk meminta sertifikat digital.
2. Server mengembalikan sertifikat digital yang dimilikinya.
3. Client melakukan verifikasi TLS certificate dari server.
4. Client melakukan pengiriman sertifikat digital yang dimilikinya.
5. Server melakukan verifikasi sertifikat yang berasal dari client.
6. Melalui koneksi yang dibuat, client dan server bisa melakukan pertukaran pesan.

Perbedaan protokol TLS dengan mTLS adalah pada langkah 4 dan 5, di mana client akan mengirimkan sertifikatnya untuk diverifikasi oleh server.




Gambar 2: Protokol mTLS

## VI. Alasan Penggunaan mTLS

Bisa dikatakan, TLS sudah cukup untuk digunakan dalam menjaga agar data yang dikirimkan pengguna aman. Namun, tidak menutup kemungkinan TLS dapat dibobol. Hal tersebut karena TLS hanya berlangsung satu arah saja.

Selain itu, mTLS juga tidak praktis untuk digunakan dalam lingkup yang besar. Hal tersebut karena mendistribusikan sertifikat dalam skala yang besar secara aman bukanlah hal yang mudah. Itu juga merupakan alasan mengapa mTLS tidak diterapkan dalam browser secara masif. Namun, bukan berarti mTLS tidak bisa diterapkan dalam browser. Dalam konteks yang lebih kecil, mTLS dapat digunakan sebagai pengaman sistem.

## VII. Penerapan mTLS Dalam Interkoneksi Layanan

Salah satu penerapan mTLS yang paling umum adalah untuk mengamankan interkoneksi layanan. Layanan-layanan long-term sebuah perusahaan skala menengah ke atas saat ini menganut arsitektur *microservices* dalam mendesain rancangannya. Seringkali, dengan arsitektur seperti ini akan terdapat banyak sekali komunikasi dan koneksi antar layanan.

Walaupun biasanya sistem ini dijaga untuk hanya bisa diakses secara long-term, tetap ada baiknya untuk menetapkan keamanan berlapis pada sistem. Salah satu cara untuk menjaga keamanan dari sistem yang berarsitektur *microservice* adalah dengan menerapkan mTLS dalam skala perusahaan.

mTLS ini dapat diterapkan lewat *ingress controller* maupun secara langsung oleh sistem. Dalam prakteknya, mTLS diterapkan lewat *ingress controller* untuk memisahkan tanggung jawab dari hal terkait sertifikat dengan sistem.

Dalam interkoneksi layanan, mTLS ini bisa dikatakan adalah penerapan dari *zero-trust security*. Secara singkat, *zero-trust security* adalah sebuah prinsip pemodelan sistem dari sudut pandang keamanan di mana secara bawaan (*default*), sistem tidak mempercayai pihak manapun yang berasal dari luar dirinya sendiri. Dengan adanya mTLS, penerapan *zero-trust security* ini bisa diterapkan secara mudah.

Untuk menghubungkan dua layanan dalam sebuah *zero-trust environment* melalui mTLS, terdapat dua syarat utama:

1. Perlu terdapat dua pasang sertifikat dan kunci yang dibuat dari pihak ketiga (yang biasa dikenal sebagai *Certificate Authority* atau CA).
2. Menggunakan sertifikat tersebut dalam TLS Server yang digunakan.

## VIII. Eksperimen

Berdasarkan prinsip-prinsip dari penjagaan keamanan yang ada, penulis telah menguji peran mTLS dalam membantu keamanan interkoneksi sistem. Berikut ini adalah alur dari percobaan yang dibuat penulis:

1. Penulis membangun dua sistem sederhana yang menerapkan protokol TLS di kedua pihak.
2. Penulis melakukan penandatangan sertifikat untuk kedua pihak (*self-signed certificate*).
3. Penulis melakukan pengujian koneksi untuk kedua layanan lewat protokol mTLS.
4. Bila dipastikan berhasil, penulis akan berusaha untuk melakukan *request* ke kedua sistem dari pihak ketiga lewat beberapa metode.

Untuk langkah pertama, penulis telah membangun dua sistem sederhana. Sistem ini dibangun tanpa melalui *ingress controller* sebagai uji coba saja. Sistem ini memiliki beberapa endpoint sebagai eksperimen. Selain itu, sistem yang dibangun akan memiliki dua metode komunikasi; yaitu dengan TLS dan tanpa TLS. Sistem dibuat sedemikian sehingga dapat diuji perbedaannya.

Endpoint yang dibuat dirancang sedemikian rupa sehingga dapat ditunjukan beberapa kasus yang mungkin terjadi dalam sebuah sistem yang memiliki TLS. Beberapa kasus yang akan dirancang untuk sistem adalah sebagai berikut:

1. Terdapat dua layanan dengan HTTPS yang saling berkomunikasi.
2. Terdapat satu client dengan HTTPS yang dapat melakukan *request* ke salah satu layanan saja. Untuk layanan lainnya tidak akan diberikan akses, namun akan diujikan sebagai eksperimen.
3. Terdapat satu client dengan HTTP (tanpa HTTPS) yang akan diujikan untuk melakukan *request* ke kedua layanan.

Selanjutnya, penulis akan membuat sertifikat TLS. Sertifikat ini nantinya akan diletakkan ke dalam sistem agar server TLS dapat berjalan dengan seharusnya. Sertifikat yang dibuat tentunya akan ada dua untuk masing-masing sistem. Sertifikat tersebut dibuat dengan perintah pada Linux melalui perintah "openssl". Dalam membuat sertifikat untuk digunakan, ada beberapa tahapan yang harus dilakukan untuk menyimulasikan bagaimana sertifikat seharusnya dibuat.

Dalam dunia nyata, PKI digunakan untuk menandatangani sertifikat yang ada di layanan-layanan yang sehari-hari kita temui. Tentunya, karena ini hanyalah sebuah eksperimen sederhana, tidak perlu menggunakan CA yang sesungguhnya untuk melakukan eksperimen sederhana ini. Alih-alih, dalam komputer sendiri pun CA sederhana dapat dibuat.

Berdasarkan bagaimana cara PKI bekerja, langkah pertama yang perlu dilakukan adalah membuat *root certificate* atau sertifikat digital utama. Berikut adalah perintah yang dilakukan:

```
$ openssl genrsa -des3 -out root.key 2048
$ openssl req -x509 -new -nodes -key
myCA.key -sha256 -days 1825 -out root.pem
```

Perintah pertama bertujuan untuk membuat kunci privat RSA dengan ukuran 2048 bit. Perintah kedua bertujuan untuk membuat kunci publik menggunakan kunci privat yang dibuat pada tahapan sebelumnya untuk digunakan pada tahapan berikutnya.

Setelah membuat *root certificate* yang sebenarnya *self-signed*, penulis akan menggunakan sertifikat tersebut sebagai CA eksperimen ini. Tentunya, langkah selanjutnya adalah dengan membangkitkan beberapa pasang sertifikat dan kunci sesuai kebutuhan. Dalam kasus eksperimen ini, akan dibuat tiga pasang; satu untuk layanan pertama, satu untuk layanan kedua, dan satu lagi digunakan oleh client. Pembuatan pasangan sertifikat dan kunci ini juga memanfaatkan kunci publik dari CA yang telah dibuat sebelumnya, karena nantinya kunci publik tersebut akan dimanfaatkan untuk melakukan verifikasi terhadap pasangan sertifikat dan kunci yang sudah dibuat.

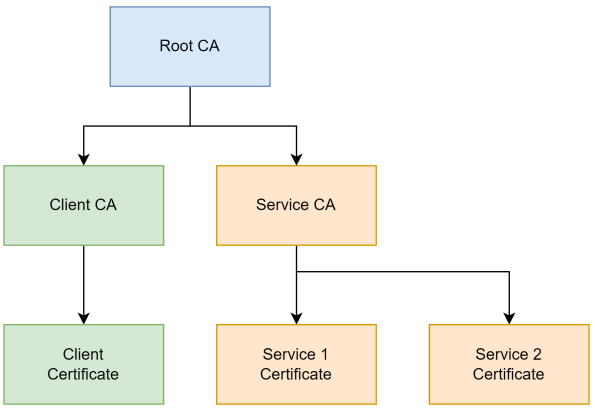
Untuk membuat kunci untuk masing-masing, dibuatlah sebuah *script* sederhana untuk melakukan pembuatan sertifikat untuk beberapa kali. Intinya, terdapat perintah berikut ini dalam *script* tersebut:

```
$ DOMAIN=localhost
$ openssl genrsa -out $DOMAIN.key 2048
$ openssl req -new -key $DOMAIN.key -out
$DOMAIN.csr
$ cat > $DOMAIN.ext << EOF
authorityKeyIdentifier=keyid,issuer
basicConstraints=CA:FALSE
keyUsage = digitalSignature,
nonRepudiation, keyEncipherment,
dataEncipherment
subjectAltName = @alt_names
[alt_names]
DNS.1 = localhost
EOF
$ openssl x509 -req -in $DOMAIN.csr -CA
root.pem -CAkey root.key -CAcreateserial
-out $DOMAIN.crt -days 825 -sha256 -extfile
$DOMAIN.ext
```

Berikut adalah penjelasan dari masing-masing perintah yang dijalankan:

- Pada beberapa tahapan sebelumnya, nilai dari variabel DOMAIN akan diisi nilai sesuai dengan domain dari layanan atau client. Nama ini hanya untuk mencatat saja sertifikat dan kunci ini berlaku untuk layanan apa saja.
- Selanjutnya, akan dibuat kunci privat dan kunci publik. Kunci privat disimpan pada ekstensi .key dan kunci publiknya akan disimpan pada ekstensi .csr.
- File berekstensi .ext berisi detail terkait sertifikat yang akan dibuat. Di dalamnya berisi DNS yang akan didaftarkan untuk sertifikat tersebut.
- Setelah kedua kunci dan detail sertifikat telah dibuat, sertifikat akan dibuat. Sesuai dengan standar, sertifikat yang dibuat adalah sertifikat x509.

Tentunya, script tersebut bisa disesuaikan dengan kebutuhan apabila diperlukan CA yang memiliki hierarki yang berbeda. Misalnya, dalam kasus eksperimen ini akan diperlukan beberapa hierarki seperti CA untuk client dan layanan. Oleh karena itu, akan dibuat dua sertifikat tambahan yang berperan sebagai sub-CA. Berikut adalah gambaran hierarki CA dalam eksperimen ini:




Gambar 3 : Hierarki sertifikat eksperimen

Langkah selanjutnya adalah memanfaatkan sertifikat dan kunci tersebut untuk layanan-layanan dan client yang akan membutuhkannya. Untuk percobaan ini, penulis menggunakan bahasa Go untuk membuat layanan-layanan dan client untuk mensimulasikan request.

Pada intinya, ada beberapa hal yang harus dilakukan pada layanan, yaitu sebagai berikut:

1. Membuat *certificate pool*. Dalam Go, *certificate pool* digunakan untuk menyimpan sertifikat x509.
2. Melakukan pembacaan sertifikat. Sertifikat yang dibaca adalah *root certificate* dari suatu layanan. Sertifikat bisa juga berupa sub-CA dari client.
3. Menyimpan *root certificate* di dalam *certificate pool*. Dalam praktiknya, *root certificate* akan digunakan untuk memverifikasi kunci dan sertifikat dari layanan yang dibuat di bawah *root certificate* tersebut. Dalam kasus eksperimen ini, root.pem akan disimpan di sini.
4. Gunakan *certificate pool* tersebut dalam HTTP Server. Dalam Go, *certificate pool* tadi digunakan dalam pengaturan TLS.
5. Lakukan serving HTTP Server tersebut dengan sertifikat dan kunci publik masing-masing layanan/client, serta dengan konfigurasi TLS dengan *certificate pool* yang dibuat. dalam kasus ini, DOMAIN.key dan DOMAIN.crt akan di-*serve* lewat HTTP server yang dibuat

Berikut adalah sepotong program Go yang dibuat yang menggambarkan salah satu layanan berdasarkan proses di atas. Beberapa hal penting yang telah dijelaskan akan ditandai.

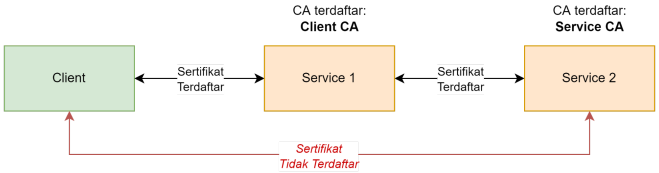
```
tlsConfig := &tls.Config{
        ClientCAs: caCertPool,
        ClientAuth: tls.RequireAndVerifyClientCert,
        InsecureSkipVerify: true,
}
tlsConfig.BuildNameToCertificate()

server := &http.Server{
        Addr: ":8443",
        TLSConfig: tlsConfig,
        Handler:   muxTLS,
}
```

Bagian yang ditandai menandakan *certificate pool* yang sebelumnya sudah ditandai dan diinisialisasi. Bagian selanjutnya juga menandakan bahwa client yang hendak terhubung ke layanan tersebut harus menyediakan sertifikat mereka. Terdapat satu bagian yang hanya digunakan untuk eksperimen, yaitu untuk bagian "InsecureSkipVerify". Hal tersebut karena CA yang dibuat adalah CA yang bukan sesungguhnya, sehingga tidak perlu diverifikasi keabsahan dari CA ketika akan dilakukan koneksi.

Seperti yang disinggung sebelumnya, proses ini akan diulang beberapa kali sesuai dengan kebutuhan. Untuk merangkum, dalam eksperimen ini penulis akan membuat simulasi *microservice* dengan mTLS.

## IX. Hasil dan Analisis Eksperimen

Eksperimen ini dilakukan dalam komputer sendiri sebagai simulasi, dengan menggunakan dua program. Program pertama menyimulasikan dua layanan lewat threading, dan program kedua dijalankan secara sporadis untuk menjalankan client. Sertifikat sudah dibuat sesuai kebutuhan sistem, yaitu terdiri dari dua layanan dan 1 client.




Gambar 4: Skema Eksperimen dan Sertifikat

Berdasarkan tiga poin percobaan yang telah dibahas pada bagian sebelumnya, berikut adalah hasil dari percobaan yang dilakukan:

1. Layanan 1 dan layanan 2 berhasil terhubung tanpa kendala lewat koneksi mTLS. Tanpa sertifikat, layanan 2 akan mengembalikan pesan kesalahan bila dilakukan *request* dari layanan 1 ke layanan 2.
2. Client dan layanan 1 perlu terhubung lewat koneksi mTLS. Tanpa sertifikat dari pihak *client*, layanan 2 pun tidak terpanggil dan layanan 1 akan mengembalikan pesan kesalahan kepada *client.*
3. Tanpa sertifikat dari client; atau dalam kata lain tidak menggunakan mTLS dan menggunakan HTTP koneksi akan ditolak.

Terdapat beberapa hal menarik yang ditemui dalam eksperimen. Bila sertifikat *invalid*, verifikasi TLS akan gagal sehingga koneksi akan ditolak. Hal ini membuktikan bahwa mTLS akan sangat efektif untuk mencegah *request* yang tidak terautorisasi oleh sertifikat. Selain itu, tidak mungkin juga untuk melakukan *request* dari client langsung kepada layanan 2, karena sertifikat yang menjadi CA client tidak terdaftar dalam layanan 2.

## X. Penerapan mTLS Dalam Autentikasi Sistem

Secara tidak langsung, eksperimen sebelumnya juga membuktikan bahwa autentikasi terhadap suatu sistem bisa dijaga. Dalam prakteknya, memang mTLS tidak diterapkan untuk browser, namun hanya untuk koneksi layanan saja.

Berdasarkan percobaan sederhana tersebut, autentikasi pengguna dapat dijaga lewat penerapan mTLS. Penulis akan mengenalkan istilah baru untuk penggunaan mTLS melalui dua istilah, yaitu secara *long-term* dan secara *short-term*.

Percobaan tadi adalah contoh penerapan mTLS dalam level *long-term*. Penerapan mTLS long-term adalah suatu cara untuk mengamankan sistem dari layanan di dalamnya. Hal tersebut sudah umum dilakukan oleh sistem-sistem yang terdapat di industri masa kini.

Walaupun mTLS secara long-term sudah aman, dapat diterapkan suatu mekanisme tambahan, yaitu mTLS *short-term*. Penerapan mTLS short-term ini dilakukan untuk memastikan hanya pihak terkait saja yang dapat mengakses sistem. Caranya adalah dengan memastikan sertifikat disebarkan secara manual kepada pihak yang memerlukan

akses. Implementasi dari metode ini bisa berupa sertifikat yang berlaku hanya untuk waktu yang singkat saja.

Kembali lagi kepada eksperimen yang telah dilakukan, alih-alih memberikan sertifikat kepada client dalam waktu yang lama (untuk contoh kasus dalam interkoneksi layanan), dalam kasus untuk melakukan autentikasi ke dalam sistem bisa diberikan sertifikat yang memiliki jangka waktu yang lebih pendek.

Penggunaan mTLS secara short-term seperti ini pada dasarnya serupa dengan membuat *short-live token*. Namun, perbedaan utamanya adalah dengan menggunakan prinsip mTLS, sistem bisa terjaga dengan lebih aman, karena penggunaan kunci hanya dilakukan pada *client-side* dan dikirimkan secara terenkripsi, sehingga akan lebih sulit untuk disadap. Dengan prinsip yang sama pada *long-term* mTLS, autentikasi sistem dapat lebih dijaga. Metode untuk meminta sertifikat bisa dilakukan dari layanan lain yang membuat sertifikat dengan jangka waktu yang pendek; mungkin terhubung hanya via TLS sederhana untuk memudahkan, atau alternatif lainnya layanan tersebut hanya terhubung lewat VPN saja.

Mengulang kembali, tentunya hal tersebut sangat tidak praktis. Tidak mudah untuk mengganti sertifikat di sisi pengguna semudah itu tanpa campur tangan pihak ketiga di luar layanan yang digunakan. Oleh karena itu, aplikasi mTLS seperti ini untuk melakukan autentikasi sistem perlu ditelaah lebih dulu; karena sebaiknya metode ini hanya diterapkan untuk sistem yang berukuran kecil saja, karena masih lebih banyak potensi solusi yang lebih baik atau cocok untuk diterapkan dari sisi pengguna.

## Tautan Repositori

Repositori untuk percobaan sederhana di makalah ini terdapat pada https://github.com/mkamadeus/mtls-demo. Repositori bertujuan hanya untuk percobaan penulis semata dan bersifat tidak reprodusibel.


## Ucapan Terima Kasih

Puji syukur kepada Tuhan YME atas rahmat dan ilham dalam menulis makalah ini. Penulis juga hendak berterima kasih kepada Dr. Ir. Rinaldi, M.T. sebagai dosen pengampu mata kuliah IF4020 Kriptografi atas ilmu di bidang kirptografi untuk jurusan Teknik Informatika ITB.


## Referensi


[1] Slide Public Key Infrastructure, Dr. Ir. Rinaldi Munir, M.T

[2] https://cpl.thalesgroup.com/faq/public-key-infrastructure-pki/what-public-key-infrastructure-pki

[3] https://www.internetsociety.org/deploy360/tls/basics/

[4] https://www.cloudflare.com/learning/access-management/what-is-mutual-tls/


## PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 20 Desember 2021

Matthew Kevin Amadeus
13518035