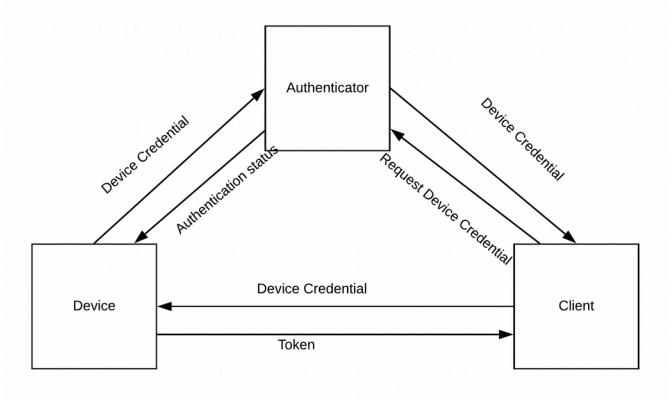
Bab III
Metode Penelitian

3.1 Design Sistem



Gambar 2: Arsitektur Sistem

Terdapat 3 pihak yang terdapat pada sistem. Setiap pihak dapat saling berkomunikasi secara langsung menggunakan protokol komunikasi yang sudah ada. Client adalah pihak yang akan melakukan *request* FOTA. Authenticator adalah server yang akan melakukan autentikasi *client*. Device adalah perangkat yang akan di perbaharui *firmware*-nya.

3.1.1 Platform

Dalam penelitian ini, sistem akan dibuat di atas sistem *network* yang sudah ada dan umum digunakan pada arsitektur IOT. Setiap komunikasi akan menggunakan TCP. Pembaharuan firmware dapat menjadi sebuah prosedur yang kritikal dan dibutuhkan jaringan yang reliable. TCP dapat memenuhi kebutuhan komunikasi tersebut. Komunikasi terhadap server akan menggunakan HTTP. HTTP adalah protokol komunikasi yang digunakan untuk *web technology*.

HTTP dipilih sebagai protokol komunikasi ke server karena HTTP sudah umum digunakan untuk *web service*. HTTP sudah memiliki banyak standard yang dapat mempermudah *developer* untuk menambahkan sistem ini ke sistem yang

sudah berjalan. HTTP juga memiliki protokol untuk *secure communication* yaitu HTTPS. Protokol ini dapat dimanfaatkan untuk menambah keamanan dari sistem. Oleh karena itu, sistem akan memiliki komunikasi yang *reliable*. Komunikasi *reliable* ini dibutuhkan untuk menjaga efisiensi pada pengiriman *firmware*.

3.1.2 FOTA Object

FOTA *object* adalah sebuah objek yang berisikan informasi tentang *state* komunikasi antar perangkat dan *client*. Objek ini akan digunakan untuk membuat sebuah token yang nantinya digunakan untuk memverifikasi *client*. FOTA *object* akan dibuat dalam bentuk JSON string yang memiliki field/attribut sebagai berikut:

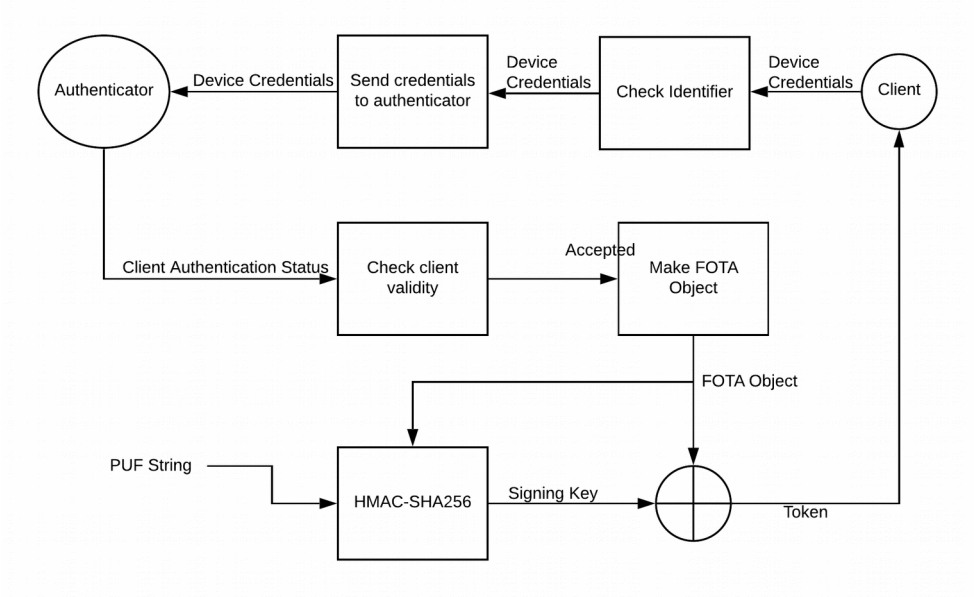
FOTA Object {
    sender_address: [IP Address Sender]
    destination_address: [IP Address receiver/device]
    expiration_date: [date time]
}

3.1.3 Perangkat

Device memiliki *pre-generated key* yang digunakan sebagai *secret key*. *Key* ini dapat dihasilkan melalui random function ataupun PUF. PUF akan memiliki performa yang lebih baik. Pembuatan *key* harus random dan unik. Selain itu, *key* akan selalu sama setiap perangkat *boot up*.

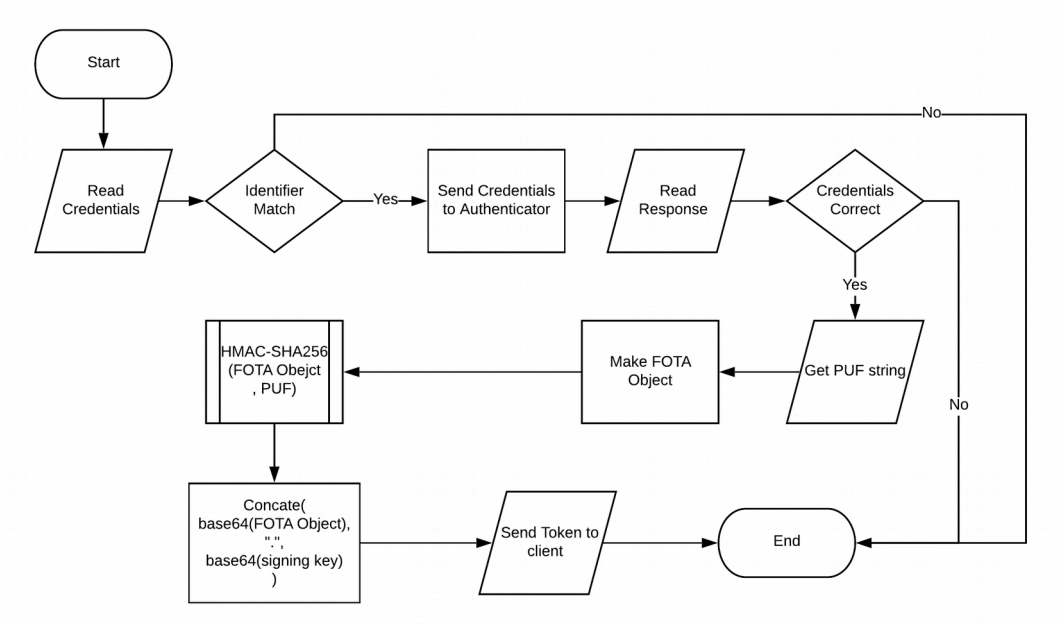
A Autentikasi *Client*

Ketika perangkat mendapatkan *credentials* dari *client*, perangkat akan menerima *packet* tersebut. Perangkat akan memverifikasi *identifier* dari packet tersebut. Jika *identifier* yang diterima sesuai, perangkat akan melakukan autentikasi pada *credentials* tersebut dengan mengirimkan credential tersebut ke authenticator.



Gambar 3: Block Diagram Autentikasi Perangkat

Authenticator akan memberikan *response credential* tersebut valid atau tidak. Jika *credenital* tersebut *valid* maka perangkat akan menghasilkan sebuah token. Token tersebut dibuat menggunakan FOTA *object* dan HMAC-SHA256 dengan *key* perangkta sebagai *secret key*. HMAC-SHA256 akan membuat objek tersebut menjadi sebuah *digest message*. *Digest message* ini akan menjadi *signing key* dari token. Setelah itu, FOTA *object* dan *signing key* tersebut masing-masing akan di

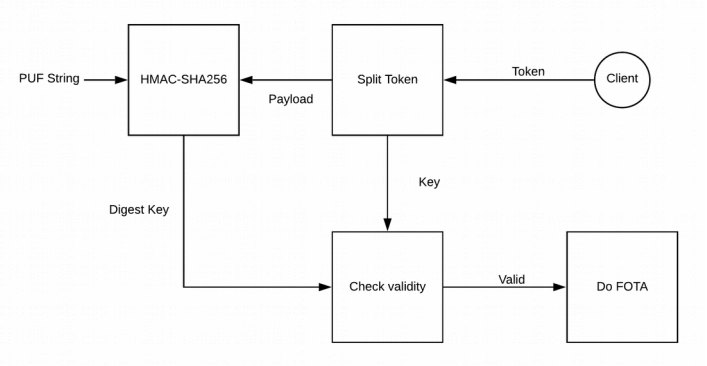


Gambar 4: Flowchart Autentikasi Perangkat

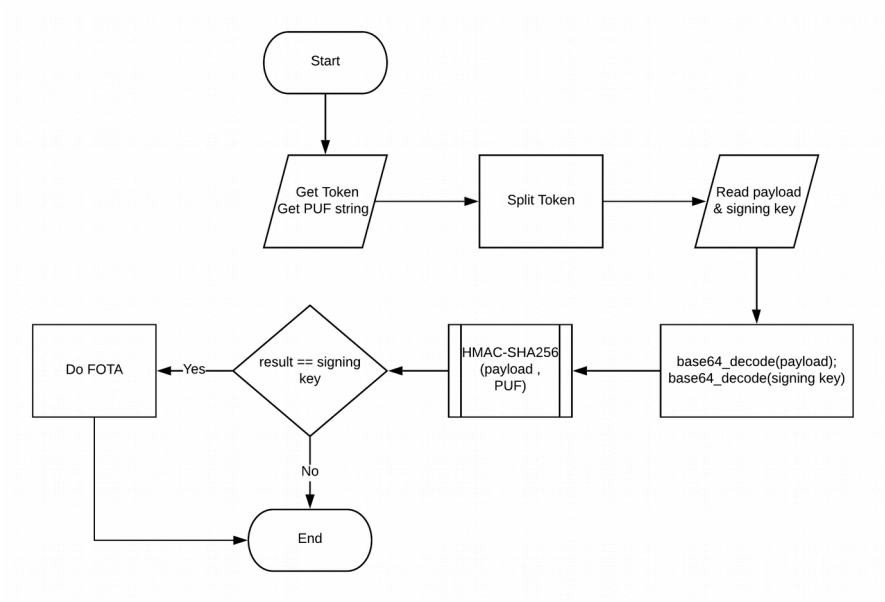
encode menggunakan base64 *encoder*. Selanjutnya, FOTA *object* dan *signing key* tersebut akan di satukan dengan "." sebagai pemisah. *String* keseluruhan tersebutlah yang menjadi token dan diberikan ke *client*. Selama token ini masih berlaku, token tersebut dapat digunakan untuk melakukan FOTA *update*. Pengecekan *identifier* diawal proses autentikasi ini berguna untuk mencegah perangkat melakukan proses yang tidak diperlukan. Cara tersebut dapat memaksimalkan penggunaan *resource* dari perangkat. Autentikasi dilakukan di server *authenticator* karena perangkat memiliki *resource* yang terbatas. Dengan mengirimkan autentikasi ke *authenticator* perangkat tidak perlu menyimpan credential yang digunakan untuk memverifikasi *client*. Hal ini dapat menghemat *resource* perangkat. Token di *sign* menggunakan HMAC untuk menjaga integritasnya saat dikirimkan ke *client* ataupun sebaliknya.

B Verifikasi Token

Pada proses ini, perangkat akan memverifikasi FOTA *packet*. Pertama, perangkat akan mengecek apakah token ada atau tidak. Token akan dipisah antar *payload* dan *key*. *Payload* dan *key* masing-masing akan di *decode*. *Payload* akan menghasilkan FOTA *object*. Perangkat akan memvalidasi payload yang diterima dengan *key*-nya. *Payload* akan di *digest* menggunakan HMAC-SHA256. Hasil *digest* tersebut akan dibandingkan dengan *key*. Jika *key* dan hasil *digest* sama maka *packet* akan diterima, jika berbeda makan *packet* akan di *drop*. Dengan melakukan validasi *sign key*, perangkat dapat terhindarkan jika ada *client* yang tidak terautorisasi yang mencoba untuk melakukan FOTA *update*. *Signing key* tidak dapat di rekonstruksi atau dibuat ulang selama *attacker* tidak memiliki *secret key* dari perangkat.



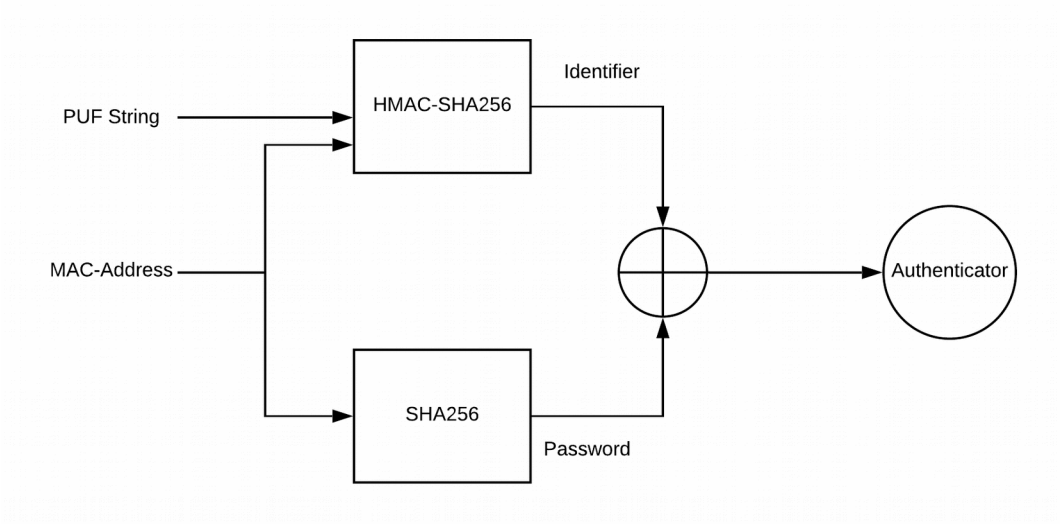
Gambar 5: Block Diagram Verifikasi Pengguna



Gambar 6: Flowchart Verifikasi Pengguna

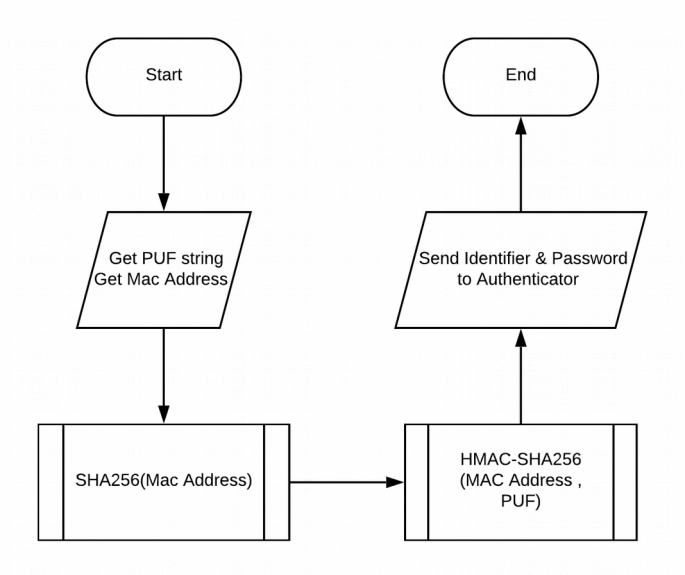
C Daftar ke *Authenticator*

Ketika perangkat *boot up*, perangkat akan mendaftarkan diri ke *authenticator*. Perangkat didaftarkan menggunakan *identifier* dan *password*. Identifier dibuat menggunakan MAC address dari perangkat. MAC address tersebut akan di *digest* menggunakan SHA256. *Password* dibuat menggunakan HMAC-SHA256. Hal tersebut dibuat dengan MAC address sebagai *message* dan *key* perangkat sebagai *secret key*. Kedua hal tersebut dikirimkan ke *authenticator* sebagai satu *credential*. *Identifier* dibuat menggunakan *MAC address* yang di *digest* menggunakan SHA256 untuk membuat sebuah identifier unik yang



Gambar 7: Block Diagram Pendaftaran Perangkat

merepresentasikan perangkat. Pada bagian *password*, HMAC digunakan agar *password* tidak mudah untuk dibuat ulang atau di rekonstruksi. HMAC pada *password* dapat menghindarkan perangkat dari *hash collision* yang dapat digunakan *attacker* untuk membuat ulang *password*.



Gambar 8: Flowchart Pendaftaran Perangkat

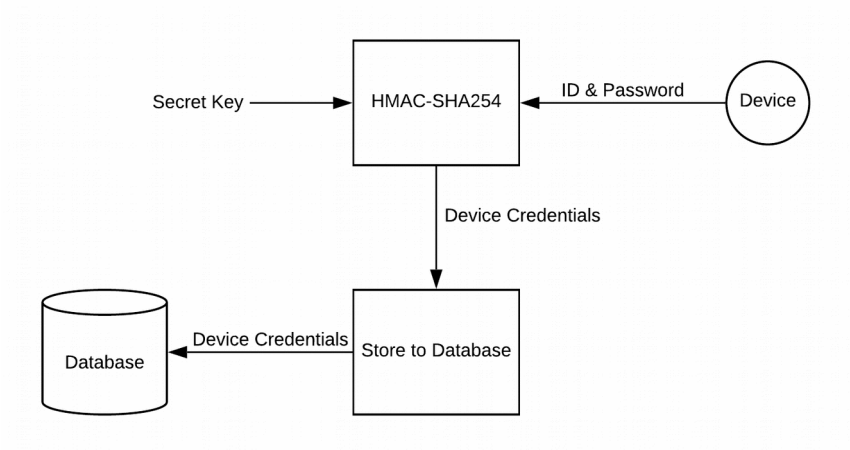
3.1.4 Authenticator

Authenticator adalah bagian pada sistem yang bertanggung jawab untuk melakukan autentikasi pada *client* yang akan melakukan FOTA *updates*. *Authenticator* ada dalam bentuk *web apps*. Terdapat 2 fungsi utama dari *authenticator* yaitu *register device* dan *get device credentials*. *Authenticator* memiliki sebuah *secret key*.

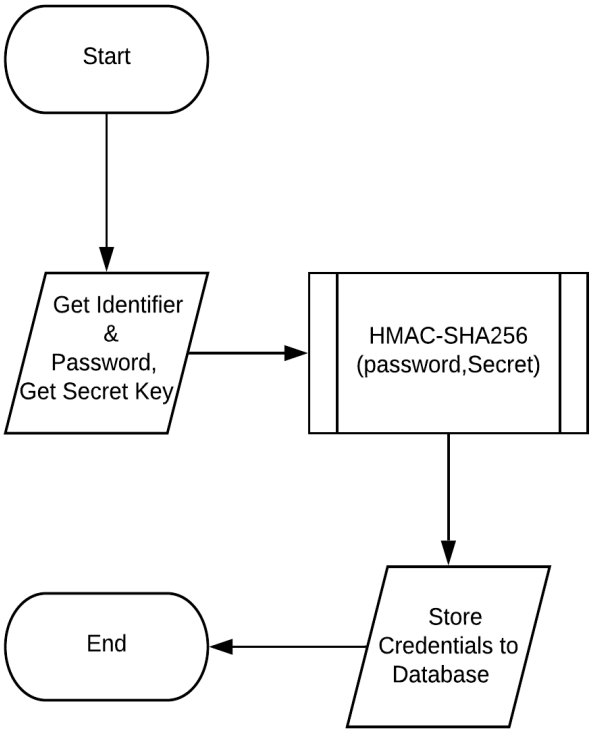
A Mendaftarkan Perangkat

Ketika *authenticator* mendapatkan *identifier* dan *password* dari perangkat, *authenticator* akan mendaftarkan *credentials* tersebut sebagai satu buah perangkat. *Authenticator* akan mengecek apakah perangkat tersebut sudah terdaftar. Jika perangkat tersebut belum terdaftar, *authenticator* akan men-*digest*

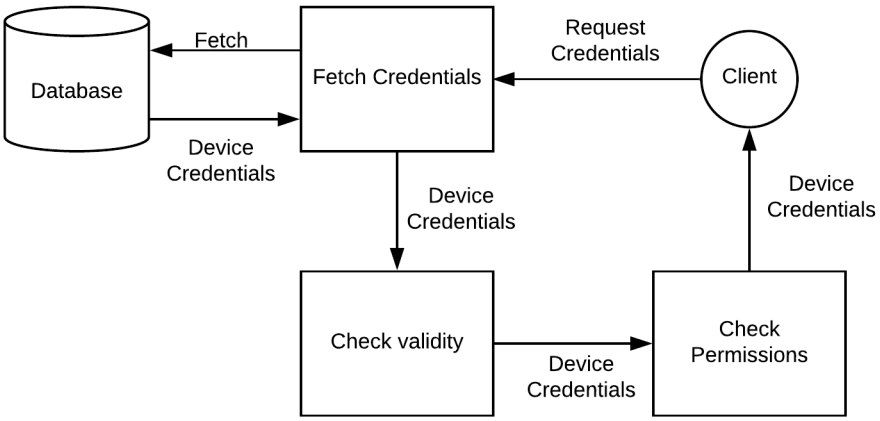
password menggunakan HMAC-SHA256 dengan *key server* sebagai *secret key*. Hasil *digest* dari *password* tersebut yang menjadi *password* baru untuk perangkat. *Identifier* dan *password* baru tersebut yang akan menjadi *credential* untuk autentikasi perangkat. *Password* di *digest* kembali dengan HMAC agar *password* tidak mudah untuk didapatkan oleh *attacker*.



Gambar 9: Block Diagram Authenticator Pendaftaran Perangkat



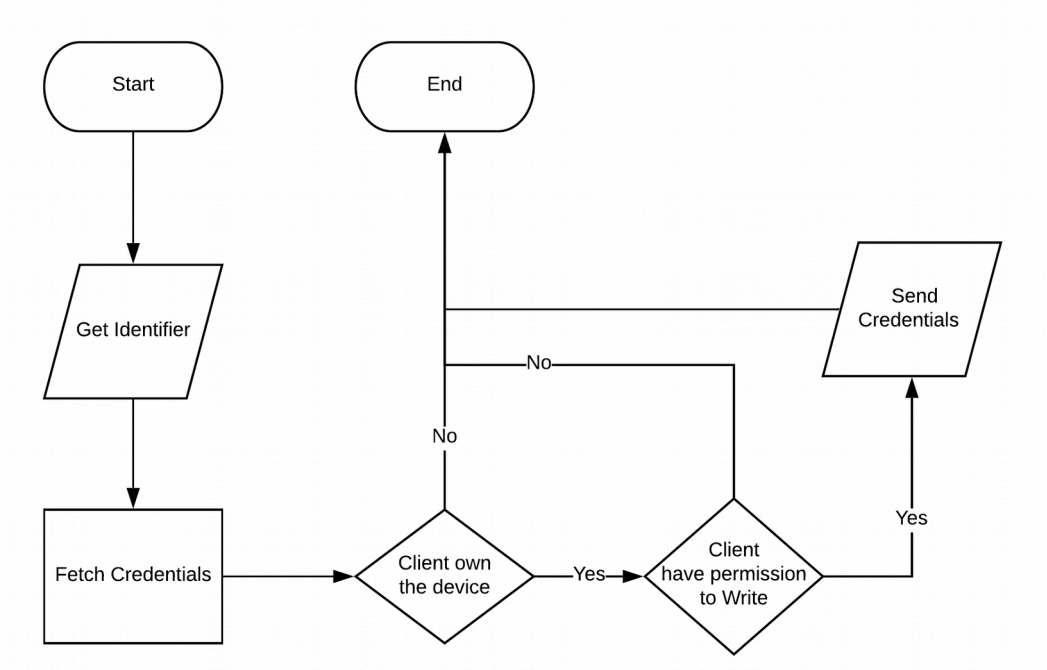
Gambar 10: Flowchart Authenticator Pendaftaran Perangkat



Gambar 11: Block Diagram Mendapatkan Credential

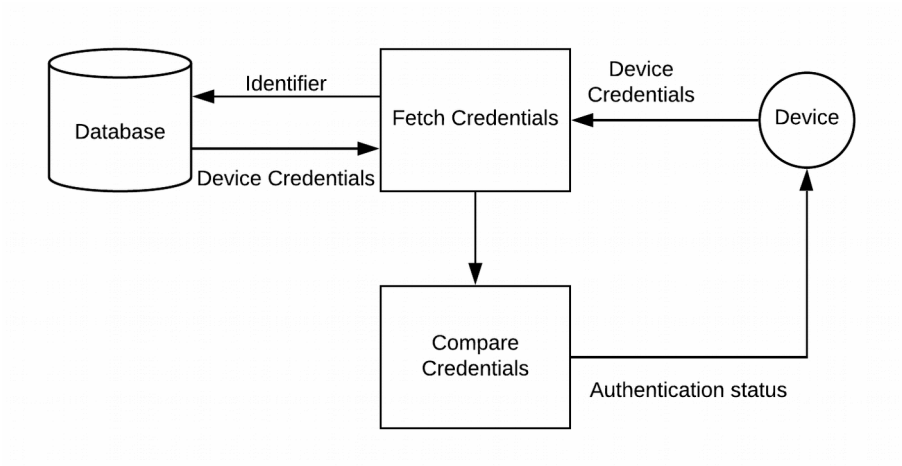
B Mendapatkan *Credential* Perangkat

Jika terdapat *request* untuk *credential* perangkat, *authenticator* akan mengecek *permission* dari *client*. Jika *client* memiliki *permission* untuk melakukan FOTA *update, authenticator* akan membuat sebuah *session* baru dengan *password* baru yang dibuat dari *password* perangkat yang ditambahkan *salt* yang di *digest* dengan HMAC-SHA256. Setelah itu , *credentials* akan



Gambar 12: Flowchart Mendapatkan Credential

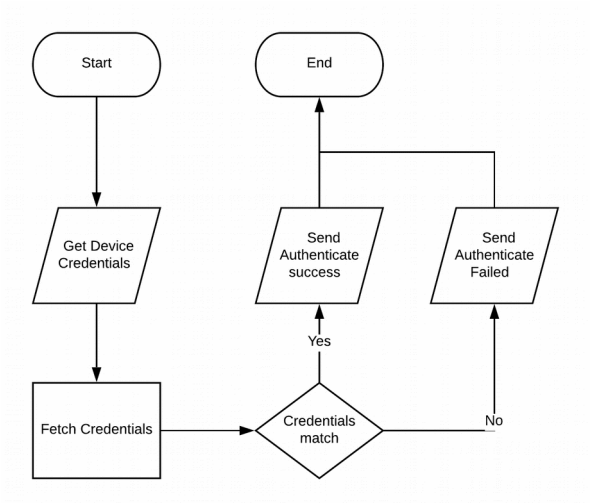
dikirimkan ke *client.* Jika perangkat tidak memiliki *permission*, *authenticator* tidak akan memberikan *credentials.* Dengan cara ini *client* dapat memilih perangkat yang akan diperbaharui tanpa harus menyimpan semua *credential* perangkat. Hanya satu *client* yang dapat melakukan FOTA pada satu waktu.



Gambar 13: Block Diagram Authenticator autentikasi pengguna

C Autentikasi

Ketika perangkat mengirimkan *authenticator credential* yang didapatkan dari *client*, *authenticator* akan mengambil data yang sesuai dengan *credential*

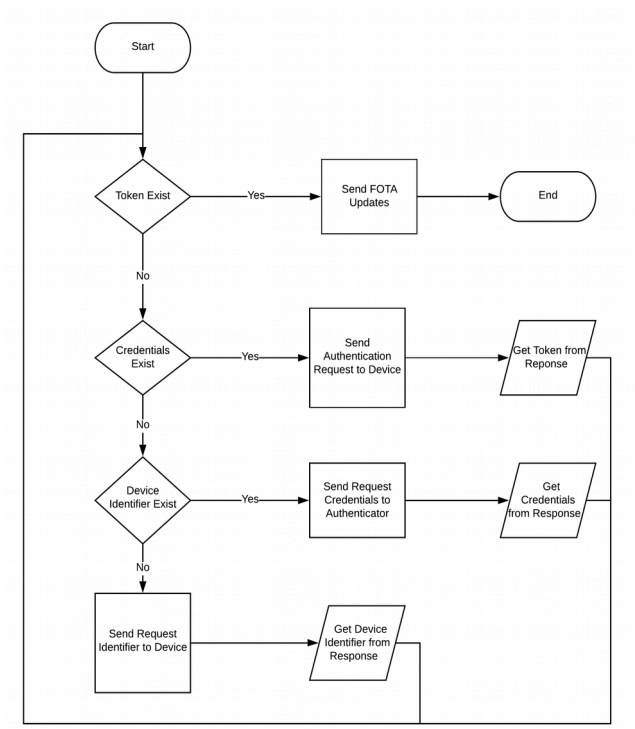


Gambar 14: Flowchart Authenticator autentikasi pengguna

teresbut dari *database*. *Authenticator* akan membandingkan apakah *password* yang diterima dari *credential* sama dengan *password* dari *database* yang ditambahkan *salt* dan di *digest* dengan HMAC-SHA256. Jika *credential* cocok maka *authenticator* akan memberikan *response* bahwa *credential* benar dan jika *credential tidak cocok*, *authenticator* akan memberikan *response* bahwa *credential* salah. Dengan cara ini *authenticator* dapat mengetahui perangkat mendapatkan *credential* dari *client* yang tepat atau tidak.

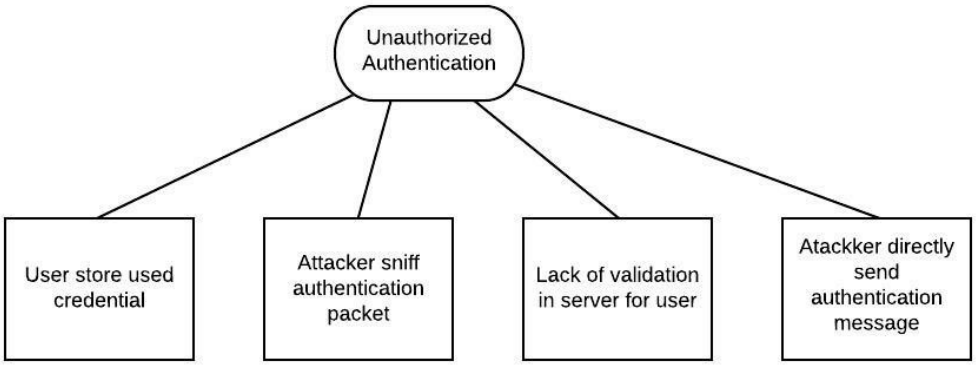
3.1.5 Client

Client adalah pihak yang akan melakukan FOTA *update* ke perangkat. *Client* akan mendapatkan *credential* dari perangkat yang akan diperbaharui dari server. *Credential* tersebut akan dikirimkan ke perangkat untuk melakukan autentikasi. Jika *credential* sesuai, perangkat akan memberikan *client* token yang akan digunakan untuk melakukan FOTA *update*. Selama melakukan FOTA *update*, *client* perlu memberikan token yang di terima dari perangkat.



Gambar 15: Flowchart pengguna FOTA Update

3.2 Threat Model



Gambar 16: Threat Model

Threat model yang dibuat adalah *attack trees*. *Model* ini memungkinkan untuk mengidentifikasi dan menjabarkan ancaman apa yang dapat terjadi dan kemungkin cara agak ancaman tersebut dapat terjadi. Seperti yang dijelaskan oleh Selin, *attack trees* mudah untuk di adopsi dan dapat digunakan secara terpisah untuk setiap komponen pada sistem [13]. Oleh karena ini pada pembuatan model ini, *attack trees* dibuat dengan menggunakan OWASP sebagai panduan untuk mengidentifikasi *threat* yang mungkin dimiliki oleh sistem.

OWASP dipilih sebagai panduan untuk menganasila *threat* karena OWASP menyediakan *source* yang berisikan *threat-threat* yang umum terjadi pada *web application*. OWASP juga mengeluarkan *handbook* yang berisikan tentang *cyber security* termasuk *threat modelling*. *Tools* dan materi yang di sediakan OWASP membantu dalam memandu menganalisa metode yang diajukan. OWASP juga memiliki OWASP *top ten list*. OWASP *top ten list* adalah *list* yang berisikan celah keamanan yang berat dan umum terjadi. List tersebut ada pada bidang *web application*, IOT, *mobile,* sampai *cloud-native technology*. Hal ini sangat membantu dalam melakukan segmentasi pada sistem untuk melakukan pengujian.

Pada metode yang diajukan pada penelitian ini, fokus utama *threat* yang akan diselesaikan adalah masalah adanya autentikasi yang tidak berwenang. Seperti yang ditunjukan pada gambar 16, Terdapat empat cara atau kemungkinan ancaman yang dapat menghasilkan masalah autentikasi tersebut. *Threat Model*

yang digunakan ini adalah *threat model* berdasarkan OWASP untuk *application*. Ancaman-ancaman tersebut yang selanjutnya akan di validasi pada tahap pengujian sistem.

Pada kemungkinan pertama yaitu *client*/*user* menyimpan *credential* disini artinya adanya kemungkinan credential tersebut akan digunakan lagi oleh *client* tersebut ataupun pihak lainnya. *Client* menyimpan *credential* artinya *credential* tersebut akan dapat selalu dapat didapatkan di dalam *client* tersebut. Hal ini juga berhubungan pada kemungkinan ancaman yang kedua yaitu *attacker* melakukan *sniffing* pada paket yang dikirimkan oleh *user*. Hal ini berhubungan karena ketika terdapat pihak yang melakukan *sniffing* maka dapat diasumsikan *credential* tersebut masih dimiliki oleh pihak tertentu.

Ancaman selanjutnya adalah jika *server* atau *service* yang digunakan pada metode ini memiliki validasi yang kurang. Validasi ini terutama berada di *server* yang menyediakan *credential* untuk *client*. Jika adanya celah atau kekurangan dalam validasi ini, *attacker* dapat melewati sistem validasi tersebut dan mendapatkan *credential* yang dibutuhkan untuk melakukan autentikasi tersebut. Selain itu validasi ini juga ada di bagian perangkat terutama pada validasi saat memverifikasi pengguna.

Pada ancaman terakhir, *attacker* secara langsung mengirimkan *message* untuk autentikasi ke perangkat. Pada ancaman ini diasumsikan *attacker* melakukan proses autentikasi dengan benar. Ancaman ini dapat terjadi pada kondisi *attacker* memiliki *credential* maupun *attacker* tidak memiliki *credential.* *Credential* tersebut dapat didapatkan dengan menebak *credential* maupun melalu ancaman-ancaman yang sudah dijelaskan sebelumnya.

3.3 Skema Pengujian Sistem

Pada pengujian ini bertujuan untuk menunjukan metode atau protokol yang dibuat pada penelitian ini dapat memberikan solusi untuk *threat model* pada subbab sebelumnya dan menjawab masalah dari penelitian ini. Hal yang akan di uji adalah kemampuan sistem memverifikasi pengguna, dan kemampuan sistem

agar tidak ada pengguna yang tidak berhak mendapatkan verifikasi dari perangkat. Skenario yang akan digunakan untuk pengujian sistem ada dua yaitu ketika sistem berjalan normal dan ketika terdapat *attacker* yang mencoba autentikasi. Skenario *attacker* yang akan digunakan adalah ketika *attacker* mencoba mengautentikasikan dirinya ke perangkat. Skenario ini dibuat berdasarkan OWASP *top ten list* 2017 untuk *broken authentication.* Oleh karena itu skenario yang digunakan adalah *attacker* akan menggunakan *credential* yang terakhir digunakan oleh *client* untuk melakukan autentikasi [14]. Skenario pada OWASP disesuaikan untuk metode yang diajukan agar dapat digunakan untuk menguji sistem ini. Skenario ini digunakan untuk pengujian karena pada tahap ini adalah tahap yang dapat menunjukan apakah sistem dapat hanya mengautentikasi pengguna yang tepat atau tidak. Pada tahap inilah *credential* ditukarkan antara perangkat dan pengguna.

3.3.1 Pengujian Autentikasi Pengguna

Skenario pengujian ini adalah skenario dimana tidak ada gangguan dari luar dan sistem berjalan normal. Sistem akan dijalankan secara keseluruhan untuk FOTA *update*. Proses ini akan sampai perangkat dapat memverifikasi *signature firmware* dan memperbaharui *code* di *flash* perangkat. Point utama yang dilihat pada pengujian ini adalah sistem dapat memverifikasi pengguna dan melakukan pembaharuan *firmware*.

3.3.2 Pengujian *Unauthorized User*

Pada pengujian ini akan ada gangguan dari luar yaitu *attacker* yang akan mencoba untuk melakukan autentikasi ke perangkat. Dalam pengujian ini *attacker* di asumsikan dapat memdapatkan *credential* terakhir yang digunakan oleh pengguna untuk mengautentikasi dirinya ke perangkat. *Attacker* juga dapat berkomunikasi dengan perangkat sehingga *attacker* dapat langsung mengirimkan paket autentikasi. Pengkondisian pada pengujian ini dibutuhkan untuk menguji metode autentikasi pada sistem secara spesifik ketika tidak ada mekanisme

pengamanan lainnya pada autentikasi tersebut. Pengujian bertujuan untuk menunjukkan apakah metode yang dibuat dapat tahan terhadap serangan dari luar.

Pengujian ini akan menjawab untuk masalah autentikasi terutama untuk ketika *client* menyimpan *credential* dan attacker mengirimkan *request* autentikasi langsung ke perangkat. Oleh karena itu, pengujian ini akan menunjukkan :

Table 1: Skema Pengujian

Ancaman	Metode Pengujian	Acuan Keberhasilan
Attacker dapat menggunakan ulang *credential* yang sudah digunakan *client*	*Attacker* di asumsikan memiliki *credential* terakhir yang digunakan *client*	*Credential* yang digunakan oleh *attacker* akan ditolak dan autentikasi gagal
Attacker dapat membuat ulang *token* yang digunakan untuk verifikasi *client*	*Analisa* algoritma yang digunakan untuk membuat token	*Attacker* tidak dapat membuat ulang token
Validasi *authenticator* minim sehingga *credential* dianggap valid meskipun salah	*Perangkat* mengirimkan *credential* dari *attacker* ke *authenticator*	*Authenticator* akan menolak *credential* tersebut.