# Javascript Protection and Encryption menggunakan Advance Encryption Standard (AES) Symetric-Key Algorithm Untuk Aplikasi Mobile Berbasis Phonegap

**Cherry Nasa Putra**
**Fakultas Informatika, Universitas Telkom**
**Jl. Telekomunikasi No. 1, Terusan Buahbatu, Bandung**
**cherrynasa@live.com**

**Abstrak**

Dengan semakin berkembangnya teknologi informasi pada saat ini telah menjadikan pengembangan aplikasi berbasis mobile semakin banyak dan digemari karena tampilan antarmuka yang sederhana tetapi masih dapat berfungsi secara maksimal. *Webservice* menjadi pilihan untuk aplikasi mobile dalam memperoleh informasi yang dibutuhkan dari server penyedia layanan informasi. Aplikasi *mobile* yang di bangun menggunkan phonegap dapat dibuat menggunakan bahasa pemograman berbasis web seperti html, css dan javascript. Untuk aplikasi yang dibangun menggunakan bahasa pemograman web, untuk melakukan akses ke suatu *webservice* aplikasi mobile tersebut dapat menggunakan javascript untuk melakukan proses pertukaran datanya.

Informasi tentang *webservice* yang akan diakses beserta proses dari aplikasi mobile tersebut terdapat pada pada file javascript yang ada didalam aplikasi yang akan dilakukan proses instalasi oleh aplikasi mobile yang akan menggunakannya. *Sourcecode* javascript yang berisi informasi ini menjadi berbahaya apabila semua pihak bisa melihatnya karena kemungkinan bisa dilakukan reverse-engineering terhadap *sourcecode* javascript tersebut.

Penelitian ini akan membuat sistem proteksi javascript yang ada pada pada aplikasi mobile sehingga apabila user dapat mengakses source dari javascript nya maka tidak akan bisa dilakukan proses *reverse-engineering* karena sudah terproteksi menggunakan *advance encryption standard*.

Aplikasi yang akan dibuat adalah aplikasi *mobile*, server aplikasi dan aplikasi untuk menyisipkan javascript kedalam file html. Semua aplikasi dibangun menggunakan bahasa pemograman web.

**Kata Kunci** : *Javascript Protector, Advance Encryption Standard, Rijndael, Reverse-Engineering, Mobile Application, Phonegap*.

**Abstract**

*The rapid growing of information technology today makes mobile application development is more preferable because its has simple user interface and still gives the maximum functionality. Webservice becomes the choice for the mobile application to retrieve all kind of informations from the server. Mobile application that built with phonegap can be made by using the web based programming language such as html, css and javascript. For the application that built using the web programming language, it can use the javascript for doing the data exchange between mobile application and server.*

*The information of accessed webservice and the mobile application logic process is placed in javascript file inside the mobile appication. User can install the mobile application in mobile device that want to use it. The javascript file that contains webservice credential and application bussiness logic become vulnerable if anyone could read it. When anyone can read the unprotected javascript file, then the reverse-engineering process could possibility happen.*

*This research is making javascript protection system within the mobile application so when somebody grant access to the javascipt file in mobile application, he cant read it and cant performing the reverse-engineering process because the javascript is protected using the encryption algorithm advance encryption standard.*

*The application thats made in this research are mobile application that using the encrypted javascript, server decription application service, and encryption application that encrypt the javascipt file and insert it to the html file.*

***Keyword** : Javascript Protector, Advance Encryption Standard, Rijndael, Reverse-Engineering, Mobile Application, Phonegap.*

## 1. PENDAHULUAN

Perkembangan teknologi informasi saat ini telah menjadikan teknologi cloud computing yang dahulu masih jarang untuk digunakan menjadi suatu hal yang telah banyak di implementasikan oleh banyak perangkat lunak yang ada saat ini. Web service menjadi sarana untuk pertukaran informasi yang berpusat pada cloud server dari penyedia jasa dari web service kemudian di distribusikan kepada perangkat lunak yang telah melakukan request sesuai dengan aturan masing-masing webservice yang ada. Proteksi terhadap aplikasi yang menjadikan webservice sebagai *core* bisnis dari suatu perusahaan yang menggunakannya juga diperlukan agar source programnya aman dan tidak dapat ditiru oleh perusahaan kompetitor sejenis [7]. Disaat yang bersamaan, javascript sebagai bahasa pemograman web yang dipakai oleh kebanyakan website modern saat ini telah digunakan juga untuk pengembangan aplikasi console game, tablets, maupun smartphone. Javascript juga merupakan core dari teknologi webservice yang saat ini banyak digunakan oleh *cloud* server penyedia layanannya. Perkembangan dari aplikasi mobile saat ini telah berkembang dengan cepat dibandingkan dengan aplikasi berbasis web karena kemudahan untuk menggunakan aplikasi yang semakin mudah dan ringkas [8]. Tugas akhir yang akan dibuat difokuskan untuk melakukan proteksi enkripsi untuk melindungi javascript pada aplikasi *mobile* berbasis phonegap.

Dengan melakukan proteksi terhadap javascript pada aplikasi *mobile*, maka javascript menjadi tidak bisa dimengerti tetapi tidak merubah fungsinya, Serta memberikan perlindungan dari metode *reverse-engineering* yang bisa dilakukan pada aplikasi *mobile*. Perlindungan enkripsi ini dilakukan ketika request dan response dari *service* provider server, sehingga hasil dekripsi javascript dari *service* provider adalah javascript yang akan dipakai oleh aplikasi *mobile* yang menjalankannya.

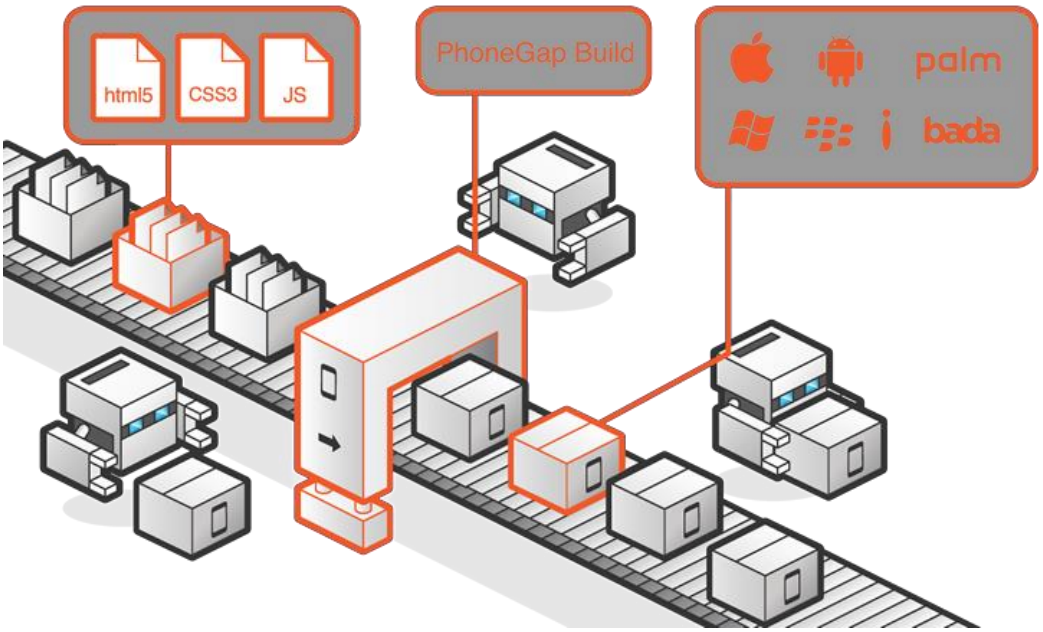
## 2. TINJAUAN PUSTAKA

### 2.1 Bahasa Pemograman Javascript

Javascript adalah bahasa pemograman tingkat tinggi, dinamis, yang mendukung gaya pemograman fungsional serta gaya pemograman berorientasi objek [1]. Javascript merupakan bahasa pemograman web client side yang tidak membutuhkan web server untuk dapat dijalankan. Dengan semakin berkembangnya teknologi informasi maka javascript tidak hanya digunakan untuk mengembangkan aplikasi berbasis web saja, tetapi javascript juga di implementasikan pada game development, aplikasi berbasis desktop serta aplikasi berbasis *mobile*. Javascript dalam pengembangan aplikasi *mobile* menggunakan web service agar aplikasi *mobile* tersebut bisa memberikan fitur tambahan seperti posting pada jejaring sosial seperti facebook atau twitter bahkan mengakses service-service penting lainnya.

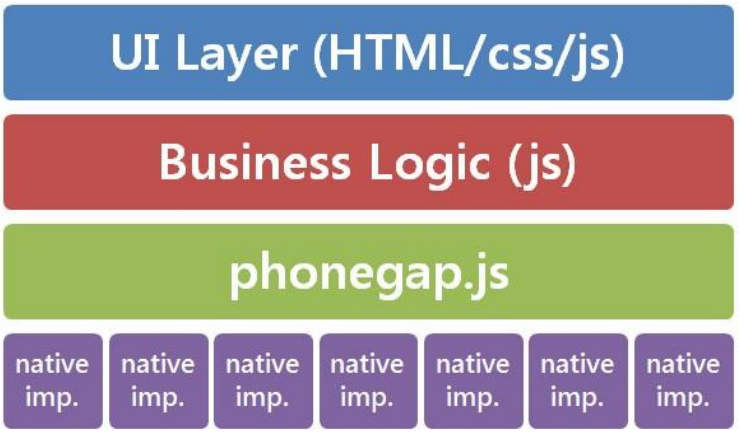
## 2.2 Phonegap

Phonegap adalah *mobile* development framework atau kerangaka kerja pengembangan aplikasi berbasis *mobile* yang memungkinkan programmer untuk membangun aplikasi *mobile* menggunakan bahasa pemograman web yaitu Javascript, HTML, dan css. Dengan menggunakan phonegap, programer hanya membuat satu jenis program yang dibuat menggunakan javascript, html, dan css kemudian dapan di *deploy* ke berbagai macam platform aplikasi *mobile* seperti ios, android, windows phone, blackberry serta platform lainnya. Untuk dapat men-deploy aplikasi ke dalam platform *mobile* yang bermacam macam, aplikasi yang dibuat menggunakan bahasa pemograman web harus dijadikan di compress menggunakan winzip hasil compres yang telah berformat .zip di upload ke server phonegap untuk dilakukan proses deploying ke platform *mobile* yang bermacam macam. Phonegap juga bersifat *open source* sehingga gratis untuk digunakan dan di dukung oleh banyak API untuk mengakses ke berbagai macam fungsi dari perangkat *mobile* yang didukungnya.




**Gambar 2.8**. Mekanisme phonegap build

Peran javascript dalam aplikasi *mobile* yang dibuat menggunakan phonegap ini ialah sebagai bussiness logic dari aplikasi. Agar aplikasi dapat melakukan aksi tertentu terhadap data atau response user maka javascipt harus ada pada setiap aplikasi *mobile* yang akan dibuat. Urutan arsitektur phonegap digambarkan pada gambar 2.9.




**Gambar 2.9**. Arsitektur phonegap

## 2.3 Advanced Encryption Standard (AES)

Advanced Encryption Standard (AES) merupakan standar enkripsi dengan kunci-simetris yang diadopsi oleh pemerintah Amerika Serikat. Standar ini terdiri atas 3 blok cipher, yaitu AES-128, AES-192 and AES-256, yang diadopsi dari koleksi yang lebih besar yang awalnya diterbitkan sebagai Rijndael. Masing-masing cipher memiliki ukuran 128-bit, dengan ukuran kunci masing-masing 128, 192, dan 256 bit. AES telah dianalisis secara luas dan sekarang digunakan di seluruh dunia, seperti halnya dengan pendahulunya, Data Encryption Standard (DES) [9].

AES diumumkan oleh Institut Nasional Standar dan Teknologi (NIST) sebagai Standar Pemrosesan Informasi Federal (FIPS) publikasi 197 (FIPS 197) pada tanggal 26 November 2001 setelah proses standardisasi selama 5 tahun, di mana ada 15 desain enkripsi yang disajikan dan dievaluasi, sebelum Rijndael terpilih sebagai yang paling cocok. AES efektif menjadi standar pemerintah Federal pada tanggal 26 Mei 2002 setelah persetujuan dari Menteri Perdagangan. AES tersedia dalam berbagai paket enkripsi yang berbeda. AES merupakan standar yang pertama yang dapat diakses publik dan sandi-terbuka yang disetujui oleh NSA untuk informasi rahasia.

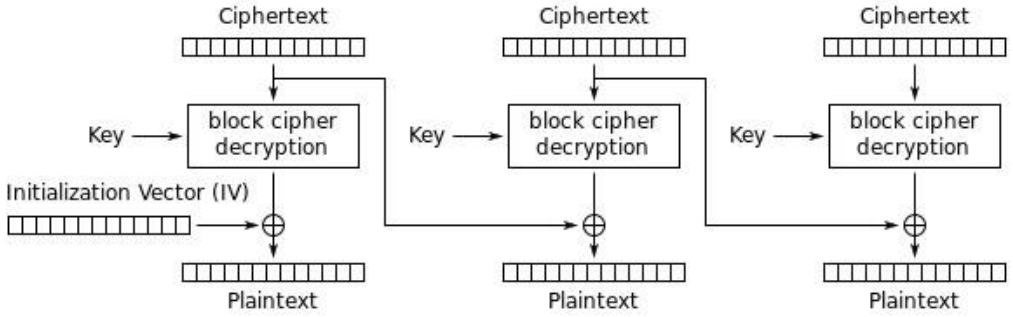
Rijndael dikembangkan oleh dua kriptografer Belgia, Joan Daemen dan Vincent Rijmen, dan diajukan oleh mereka untuk proses seleksi AES. Rijndael adalah permainan kata dari kedua nama penemu.

Jika dibandingkan dengan pendahulunya yaitu DES yang memiliki panjang key 56 bit, AES mendukung panjang key 128, 192, dan 256 bit. Diberikan bahwa blok data dienkripsi dan blok dekripsi diketahui, secara teori jika 56 bit DES dapat dipecahkan dalam 1 detik dengan mencoba setiap key. Cara yang sama 128 bit AES hanya dapat dipecahkan selama $1.5x10^{14}$ tahun, 192 bit AES hanya dapat dipecahkan selama $2.8x10^{33}$ tahun, 256 bit AES hanya dapat dipecahkan selama $5.1X10^{52}$ tahun. Dapat disimpulkan bahwa AES memiliki keamanan yang cukup tangguh hingga saat ini.

Pengujian keamanan dilakukan menggunakan metode pengujian keamanan Exhaustive Search atau yang lebih dikenal dengan istilah brute force. Percobaan metode ini akan dilakukan dengan cara mencoba kemungkinan kunci satu per satu. Exhaustive Search akan melakukan pencarian nilai private key. Sesuai dengan panjang kunci yang diterapkan pada sistem enkripsi pesan ini, jika yang terendah adalah 128 bit maka akan dilakukan pengujian keamanan Exchaustive Search untuk panjang kunci 128 bit .

AES menawarkan keamanan yang luar biasa dan juga performansi yang mumpuni, itulah mengapa hingga saat ini banyak produk perangkat lunak yang menggunakan AES dalam implementasi keamanannya. Pada cipherblok, rangkaian bit-bit plainteks dibagi menjadi blok-blok bit dengan panjang sama, yaitu 128 bit. Algoritma AES menghasilkan blok cipherteks yang berukuran sama dengan blok

plainteks. Mode pemrosesan cipherpaling sederhana adalah Electronic Code Book (ECB). Pada mode ini, setiap blok plainteks dienkripsi secara individual dan independen, kelemahan mode ECB adalah deterministik dikarenakan blok data yang sama selalu menghasilkan cipheryang sama juga. Penggunaan Cipher Block Chaining (CBC)pada sistem yang dikembangkan dapat menghasilkan sistem yang lebih baik. Dengan mode CBC, setiap blok cipherteks bergantung tidak hanya pada blok plainteksnya tetapi juga pada seluruh blok plainteks sebelumnya. Dekripsi dilakukan dengan memasukkan blok cipherteks yang current ke fungsi dekripsi, kemudian meng-XOR-kan hasilnya dengan blok cipherteks sebelumnya. Blok cipherteks sebelumnya berfungsi sebagai umpan-maju (feedforward) pada akhir proses dekripsi. blok plainteks pertama menggunakan Initialization Vector (IV)sebagai vektor awal. IV tidak perlu rahasia. Algoritma AES dengan CBC membutuhkan input yang memiliki ukuran tepat multiplikasi dari ukuran blok..

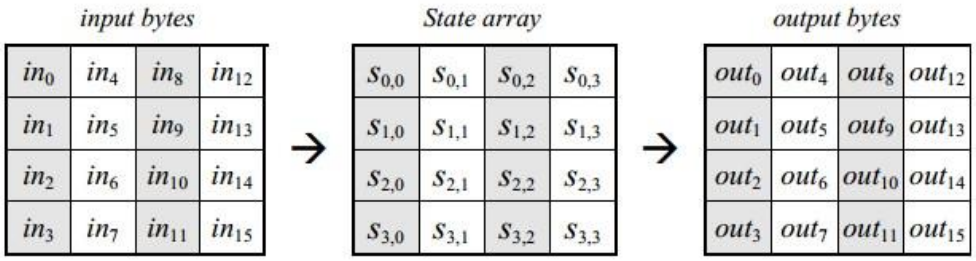



**Gambar 2.1**. Enkripsi Mode CBC

Pada gambar 2.1 digambarkan metode enkripsi algoritma AES dengan metode CBC yang menggunakan *initialization vector* (IV) sehingga menghasilkan ciphertext yang selalu berubah ubah walaupun kunci yang digunakan sama.

## 2.4 Sistem Enkripsi Algoritma AES

Cara kerja algoritma enkripsi AES ini di mulai dengan meletakan plaintext pada matriks segi empat atau yang biasa disebut dengan state matriks berukuran 4x4 yang setiap sel nya berisi 1 byte untuk AES-128.




**Gambar 2.2**. State matriks berukuran 4x4.

Karena algoritma AES merupakan algoritma iterated block chiper maka proses untuk mengenkripsi suatu plaintext menjadi chippertext memerlukan ronde atau round. Proses yang dijalankan ditiap roundnya adalah sebagai berikut.

```
AddRoundKey(State);
Round(State,keyround)
```

```
{
      ByteSub(State);
      ShiftRow(State);
      MixColumn(State);
      AddRoundKey(State);
}
FinalRound(State,keyround)
{
      ByteSub(State);
      ShiftRow(State);
      AddRoundKey(State);
}
```

Ukuran besarnya jumlah round pada tiap ukuran algoritma AES bervariasi. Untuk AES-128bit memiliki jumlah round yaitu 10, sedangkan untuk AES-192bit dan AES-256bit memiliki jumlah round sebesar 12 dan 14. Proses yang dilakukan ditiap round sama, hanya saja untuk round terakhir tidak dilakukan proses mixcolumn. Ukuran besarnya input blok, output blok dan state matriks adalah 128 bit. Hal ini bisa direpsredentasikan dengan Nb=4, yang merefleksikan jumlah 32-bit kata (jumlah kolom) yang terdapat pada state matriks. Sedangkan untuk ukuran chipher key atau K adalah 128, 192, dan 256 bit. Panjang kunci ini derepresentasikan oleh Nk= 4,6, atau 8 yang merefleksikan 32-bit kata (jumlah kolom) pada chiper key.

| | Panjang Kunci (Nk) | Panjang Blok (Nb) | Jumlah Ronde (Nr) |
|---|---|---|---|
| **AES-128** | 4 | 4 | 10 |
| **AES-192** | 6 | 4 | 12 |
| **AES-256** | 8 | 4 | 14 |

**Gambar 2.3.** Kombinasi kunci, blok dan ronde.

Algoritma AES menggunakan fungsi round ini pada saat proses enkripsi dan dekripsi yaitu ketika mengubah plaintext menjadi chipertext dan proses invers chiper yaitu mengubah chipertext menjadi plaintext.

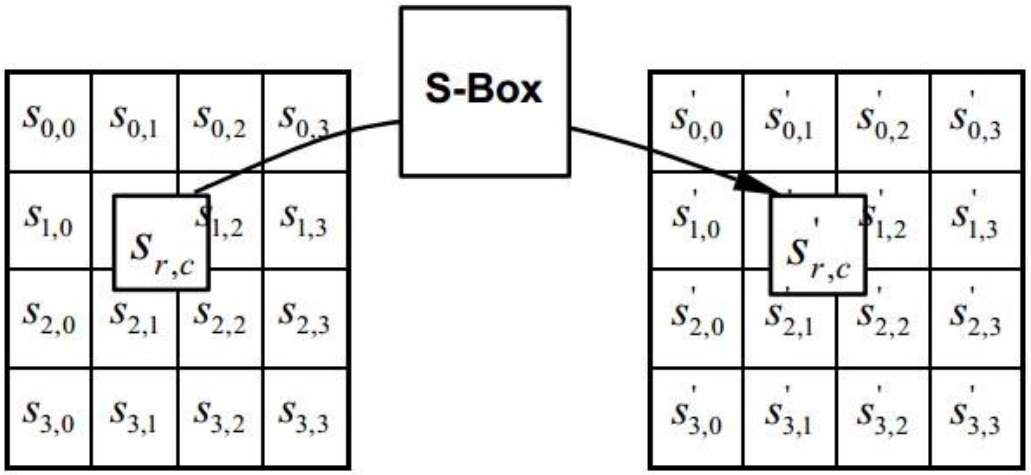
## 2.5 Transformasi SubBytes

Transformasi subbyte merupakan transformasi yang memiliki sifat non-linear pada algoritma AES. Sifat non linear ini terjadi ketika proses subtitusi byte yang beroperasi secara independen pada setiap byte pada state matriks menggunakan tabel subtitusi (s-box).

| | | y | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f |
| x | 0 | 63 | 7C | 77 | 7B | F2 | 6B | 6F | C5 | 30 | 01 | 67 | 2B | FE | D7 | AB | 76 |
| | 1 | CA | 82 | C9 | 7D | FA | 59 | 47 | F0 | AD | D4 | A2 | AF | 9C | A4 | 72 | C0 |
| | 2 | B7 | FD | 93 | 26 | 36 | 3F | F7 | CC | 34 | A5 | E5 | F1 | 71 | D8 | 31 | 15 |
| | 3 | 04 | C7 | 23 | C3 | 18 | 96 | 05 | 9A | 07 | 12 | 80 | E2 | EB | 27 | B2 | 75 |
| | 4 | 09 | 83 | 2C | 1A | 1B | 6E | 5A | A0 | 52 | 3B | D6 | B3 | 29 | E3 | 2F | 84 |
| | 5 | 53 | D1 | 00 | ED | 20 | FC | B1 | 5B | 6A | CB | BE | 39 | 4A | 4C | 58 | CF |
| | 6 | D0 | EF | AA | FB | 43 | 4D | 33 | 85 | 45 | F9 | 02 | 7F | 50 | 3C | 9F | A8 |
| | 7 | 51 | A3 | 40 | 8F | 92 | 9D | 38 | F5 | BC | B6 | DA | 21 | 10 | FF | F3 | D2 |
| | 8 | CD | 0C | 13 | EC | 5F | 97 | 44 | 17 | C4 | A7 | 7E | 3D | 64 | 5D | 19 | 73 |
| | 9 | 60 | 81 | 4F | DC | 22 | 2A | 90 | 88 | 46 | EE | B8 | 14 | DE | 5E | 0B | DB |
| | A | E0 | 32 | 3A | 0A | 49 | 06 | 24 | 5C | C2 | D3 | AC | 62 | 91 | 95 | E4 | 79 |
| | B | E7 | C8 | 37 | 6D | 8D | D5 | 4E | A9 | 6C | 56 | F4 | EA | 65 | 7A | AE | 08 |
| | C | BA | 78 | 25 | 2E | 1C | A6 | B4 | C6 | E8 | DD | 74 | 1F | 4B | BD | 8B | 8A |
| | D | 70 | 3E | B5 | 66 | 48 | 03 | F6 | 0E | 61 | 35 | 57 | B9 | 86 | C1 | 1D | 9E |
| | E | E1 | F8 | 98 | 11 | 69 | D9 | 8E | 94 | 9B | 1E | 87 | E9 | CE | 55 | 28 | DF |
| | F | 8C | A1 | 89 | 0D | BF | E6 | 42 | 68 | 41 | 99 | 2D | 0F | B0 | 54 | BB | 16 |

**Gambar 2.4.** S-box pada proses subbyte.

Walaupun s-box ini memiliki sifat non-linear tetapi tetap memiliki sifat invertible atau dapat dibalik prosesnya untuk proses dekripsi. Proses subbyte ini merupakan proses subtitusi antara state matriks dengan s-box kemudian menghasilkan output state matriks yang sudah dilakukan subtitusi.

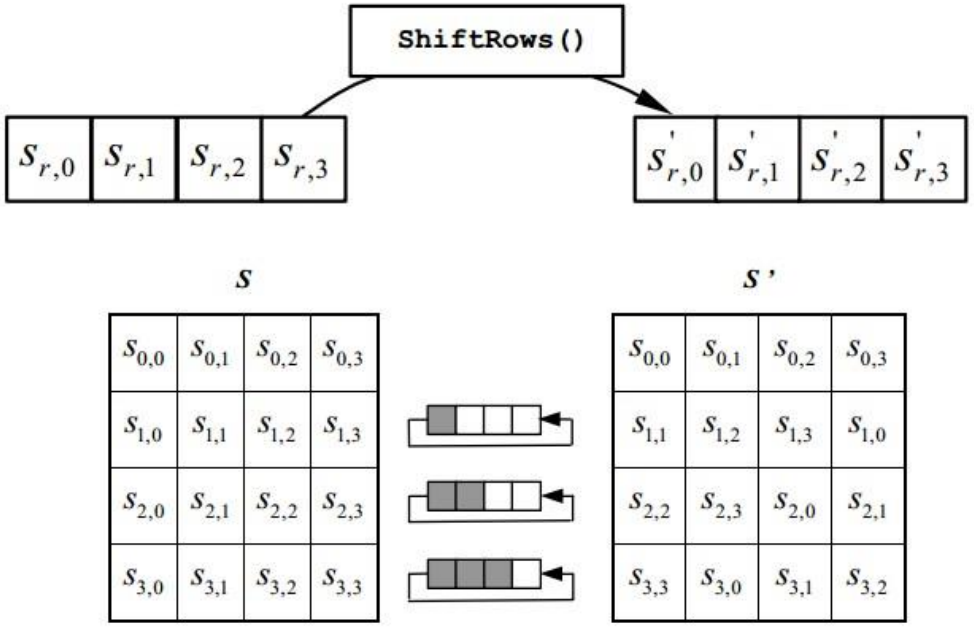



**Gambar 2.5.** Proses subtitusi state matriks dengan s-box.

S-box yang digunakan pada proses subbyte ini merupakan s-box pada gambar 3. Misalkan nilai kolom $s_{1,1}$ pada state matriks memiliki nilai 78 maka kita cari titik temu antara nilai 7 dan 8 ada s-box. Kita cari titik temu antara baris dengan nomor 7 dan colom dengan nilai 8 maka nilai subtitusinya adalah BC.

## 2.6 Transformasi ShiftRow

Proses tranformasi shiftrow merupakan suatu proses transposisi byte yang dapat mengeser baris dari suatu state matriks berdasarkan suatu pola (ke kiri atau ke kanan) dan nilai offset yang bervariasi. Pola ini bisa berupa pengeseran baris kekiri atau kekanan tergantung apakah proses tersebut merupakan proses enkripsi atau dekripsi. Proses penggeseran ini dimulai dari baris kedua hingga baris terakhir pada state matriks.




**Gambar 2.6.** Proses shiftrow

Sebagaimana contoh pada gambar 2.6. Kondisi state matriks pada baris kedua awalnya yaitu $S_{1,0,}$ $S_{1,1,}$ $S_{1,2,}$ $S_{1,3}$ akan dilakukan subbyte yaitu dengan arah pergeseran kekiri dan jumlah offset pergeserannya yaitu 1 maka akan menghasilkan urutan state matriks pada baris kedua yaitu $S_{1,1,}$ $S_{1,2,}$ $S_{1,3,}$ $S_{1,0,}$. Hal ini dilakukan pada baris kedua hingga baris terakhir.

## 2.7 Transformasi MixColumns

Tranformasi mixcolumn mengoperasikan kolom-kolom yang terdapat pada state matriks dengan merepresentasikan nilai-nilai pada kolom tersebut sebagai notasi polinomial dengan derajat 4.

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$$

proses mixcolumn merupakan operasi linier yang bertujuan untuk menciptakan diffusion (pengacakan) nilai nilai yang ada pada state matriks dengan cara melakukan perkalian matriks dengan suatu nilai matriks pengkalinya. Matriks pengkali ini merupakan matriks berukuran 4x4. Matriks ini yang kemudian dikalikan dengan kolom yang terdapat pada state matriks.

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix} \quad \text{for } 0 \le c < \boldsymbol{Nb}.$$

Persamaan diatas merupakan persamaan untuk nilai pengkali state matriks pada proses mixcolumn. $S'_{0,c}$ sampai $S'_{3,c}$ merupakan nilai state matriks output dari proses mixcolomn setelah proses pengkalian. $S_{0,c}$

sampai $S_{3,c}$ merupakan nilai state matriks awal sebelun dilakukan proses pengkalian dengan nilai-nilai pada matriks pengkali 4x4. Proses perkaliannya seperti perkalian matriks pada umumnya.

$$s'_{0,c} = (\{02\} \bullet s_{0,c}) \oplus (\{03\} \bullet s_{1,c}) \oplus s_{2,c} \oplus s_{3,c}$$

$$s'_{1,c} = s_{0,c} \oplus (\{02\} \bullet s_{1,c}) \oplus (\{03\} \bullet s_{2,c}) \oplus s_{3,c}$$

$$s'_{2,c} = s_{0,c} \oplus s_{1,c} \oplus (\{02\} \bullet s_{2,c}) \oplus (\{03\} \bullet s_{3,c})$$

$$s'_{3,c} = (\{03\} \bullet s_{0,c}) \oplus s_{1,c} \oplus s_{2,c} \oplus (\{02\} \bullet s_{3,c}).$$

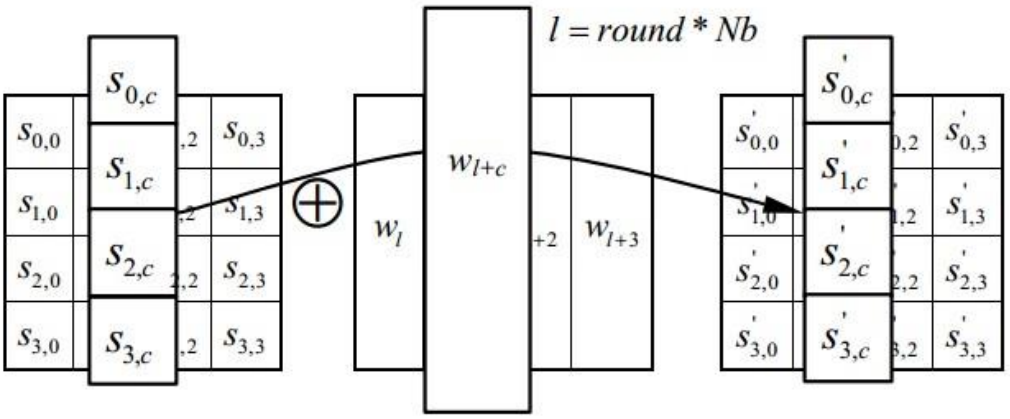
Proses perkalian state matriks dengan matriks pengali pada mixcolumn digambarkan pada persamaan diatas.

### 2.8 Transformasi AddRoundKey

Pada proses transformasi addroundkey, kunci ronde atau round key ditambahkan pada setiap kolom yang terdapat pada state matriks dengan menggunakan operasi bitwise XOR sederhana. Setiap round terdiri dari Nb (Jumlah round) kata-kata dari key schedule. Keyschedule terdiri dari dua komponen yaitu key expansion (ekspansi kunci) dan round key selection (pemilihan kunci pada suatu ronde). Key expansion menspesifikasikan bagaimana expandedkey (kunci yang diperbesar) didapatkan dari chiper key.

$$[s'_{0,c}, s'_{1,c}, s'_{2,c}, s'_{3,c}] = [s_{0,c}, s_{1,c}, s_{2,c}, s_{3,c}] \oplus [w_{round*Nb+c}] \quad \text{for } 0 \le c < \mathbf{Nb},$$

$S'_{0,c}$ sampai $S'_{3,c}$ merupakan kolom-kolom pada state matriks setelah dilakukan proses addroundkey. $S'_{0,c}$ sampai $S'_{3,c}$ merupakan kolom kolom pada state matriks setelah dilakukan proses addroundkey. $W_i$ merupakan kata-kata yang terdapat pada key schedule. Round merupakan nilai round atau posisi di round berapakah addroundkey ini dilakukan. Adapun besarnya nilai round ini minimal 0 dan maksimal adalah sebesar round maksimal yang digunakan. Round maksimal ini bisa 10, 12 atu 14 tergantuk jenis algoritma AES yang digunakan.




**Gambar 2.7.** Proses addroundkey pada setiap kolom state matriks.

Transformasi addroundkey melakukan proses XOR pada setiap kolom pada state matriks dengan menggunakan kata-kata yang didapatkan dari keyschedule.

### 2.9 Differential Cryptanalysis

Differential Crypanalis merupakan salah satu metode serangan yang dapat digunakan untuk menyerang algorima *advance encryption standard* (AES). Metode serangan ini biasanya menggunakan plaintext yang sudah dilakukan pemilihan dengan menggunakan parameter tertentu. Penyerang yang menggunkan metode ini juga harus bisa mendapatkan chipertext dari beberapa set plaintext yang sudah dipilih sebelumnya. Cara kerja dasar metode ini adalah dengan menggunakan pasangan plaintext yang memiliki suatu perbedaan yang konstan. Perbedan bisa didefinisikan dengan berbagai macam cara, perbedaan yang biasanya digunakan adalah operasi ekslusif OR (XOR).
Penyerang kemudian melakukan proses enkripsi untuk menghasilkan ciphertext dari plaintext yang telah dipilih kemudian menganalisa pola statistik dari data tersebut.
Misalnya akan dilakukan serangan differential attack terhadap algoritma enkripsi AES dengan dua round maka akan dilakukan analisa terhadap jumlah sbox yang aktif pada proses *subbyte*. Dari informasi *subbyte* yang aktif dapat dikalkulasikan untuk menentukan probabilitas dari kemungkinan kunci yang digunakan.

### 2.10 Brute Force Attack

Didalam kriptografi, *brute-force attack* atau *exhaustive key search* adalah serangan kriptoanalitik terhadap data yang telah terenkripsi. Serangan ini terdiri dari pengecekan secara sistematis dari semua kunci atau password hingga menemukan kunci yang digunkan pada suatu ciphertext. Dalam kasus yang terburuk, proses pengecekan kunci atau password ini bisa memakan waktu yang amat sangat lama. Ketika metode ini digunkan untuk menebak password, metode ini berjalan sangat cepat hanya ketika melakukan pengecekan pada password yang pendek. Pada kasus password atau kunci yang panjang digunakan metode dictionary attack. Misalkan kunci yang digunakan pada proses enkripsi terdiri dari gabungan huruf dan angka maka metode brute force ini melakukan pengujian terhadap semua kombinasi huruf dan angka sampai kunci yang dicari dapat ditemukan. Proses ini memakan waktu dan memori kumputasi yang sangat besar apabila kunci yang digunakan kompleks.

## 3. METODE PENELITIAN

Sistem enkripsi javascript ini merupakan sistem pengamanan javascript pada aplikasi *mobile* yang dibangun menggunakan bahasa pemograman yaitu html, javascript dan css dan di *deploy* ke aplikasi *native mobile* menggunakan phonegap.

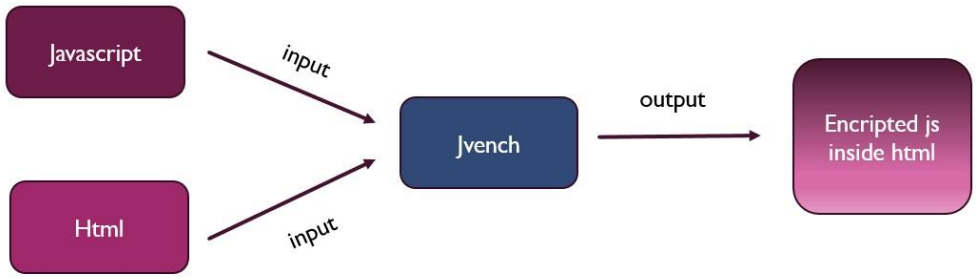
Karena aplikasi ini dibangun menggunakan bahasa pemograman web, maka ketika di deploy menjadi aplikasi native contoh nya aplikasi android yang berformat .apk maka setiap user yang akan menggunakan aplikasi tersebut haruslah mengunduh aplikasi tersebut kemudian melakukan proses instalasi aplikasi pada perangkat *mobile* mereka. Format aplikasi .apk tersebut masih bisa di ekstrak isinya menggunkan aplikasi seperti 7zip atau winrar. Apabila isi dari aplikasi tersebut berhasil diekstrak dan dilihat isi dari aplikasi tersebut maka sourcodenya pun bisa dibuka, dipelajari, serta bisa dilakukan eksploitasi [6]. Javascript sebagi proses bisnis dari aplikasi *mobile* ini menggunakan *webservice* untuk melakukan komunikasi ke server aplikasi untuk mendapatkan data-data yang dibutuhkan. Untuk melakukan akses ke *webservice*, javascript harus mendefinisikan alamat *server*, *port*, serta *credential* dari *webservice* apabila *webservice* tersebut dilindungi oleh password *credential*. Apabila javascript

dari aplikasi berhasil dibaca, maka untuk melakukan eksploitasi terhadap aplikasi dengan mengakses *webservice* server menggunakan *credential* aplikasi *mobile* dapat dilakukan.

Selama ini proses proteksi terhadap javascript telah banyak digunakan oleh aplikasi aplikasi berbasis web yang ada pada saat ini. Salah satu contohnya adalah google.com. search engine google telah mengimplementasikan proteksi javascriptnya dengan melakukan obfuskasi terhadap source code dari javascript yang mereka gunakan. Apabila kita melihat *sourcecode* halaman google.com dan membuka sintaks javascriptnya maka yang kita lihat hanyalah sekumpulan kombinasi karakter-karakter huruf, angka serta karakter lainnya yang seolah-olah tidak memiliki arti. Padahal proses dari mesin pencari google adalah terletak pada javascriptnya, baik untuk melakukan proses query ataupun untuk terkoneksi dengan *webservice search engine* google itu sendiri.

Proses obfuskasi memang bisa mengamankan *source code* javascript pada aplikasi yang dibangun menggunakan bahasa pemograman berbasis *web*. Tetapi apabila pola dari obfuskasi berhasil terbaca maka sintaks javascript yang telah terobfuskasi tersebut bisa dilakukan deobfuskasi dan sintaks asli dari javascript tersebut akan terbaca. Proses obfuskasi yang ada saat ini tidak menggunakan password sehingga apabila polanya sudah berhasil terbaca maka proteksi menjadi tidak aman.

Sistem enkripsi javascript akan memproteksi javacript dengan menggunakan enkripsi simetris AES untuk mengenkripsi javascript pada aplikasi yang dibangun menggunakan bahasa pemograman *web* dan phonegap. Ketika aplikasi yang telah di *deploy* di ekstraksi oleh pengguna untuk dilihat sintaks javascript maka pengguna tersebut akan melihat sintaks javascript yang sudah terenkripsi menggunakan algoritma enkripsi AES 128bit dan membutuhkan kunci tertentu untuk membukanya.



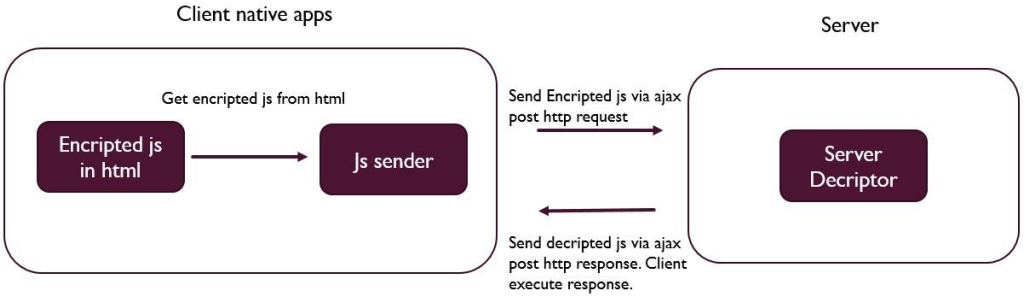

**Gambar 3.1**. Perancangan aplikasi enkripsi

Dari gambar 3.1 dapat dijelaskan bahwa input data yaitu file javascript dan file html. Javascript yang belum dienkripsi dimasukan kedalam aplikasi jvench kemudian aplikasi melakukan *read file* dari *file* yang di inputkan yaitu *file* html dan javascript. Setelah file berhasil di *read*, aplikasi kemudian mengirimkan *raw text* tersebut ke server untuk dilakukan proses enkripsi. Hasil *raw text* enkripsi ini ditampilkan oleh aplikasi jvench kemudian user bisa melakukan *download* file hasil enkripsi ini. Untuk disimpan di aplikasi *mobile* yang akan menggunakan sintaks javascript ini.

Ketika aplikasi mobile dijalankan, aplikasi akan terhubung dengan *webservice* pada dekriptor *server*. *Webservice decriptor* merupakan *service* untuk melakukan proses dekripsi javascript yang terenkripsi pada aplikasi *mobile*. *Server decriptor* akan melakukan validasi perangkat yang melakukan *request*. Apakah perangkat tersebut termasuk kedalam perangkat yang diijinkan atau tidak diijinkan. Apabila perangkat yang melakukan *request* dekripsi merupakan perangkat yang diijinkan maka *webservice server decriptor* akan melakukan proses dekripsi javascript yang terenkripsi.

Setelah javascript berhasil didekripsi, *webservice server decriptor* melakukan pengiriman ke aplikasi mobile yang melakukan request untuk dekripsi javascript yang terkripsi.

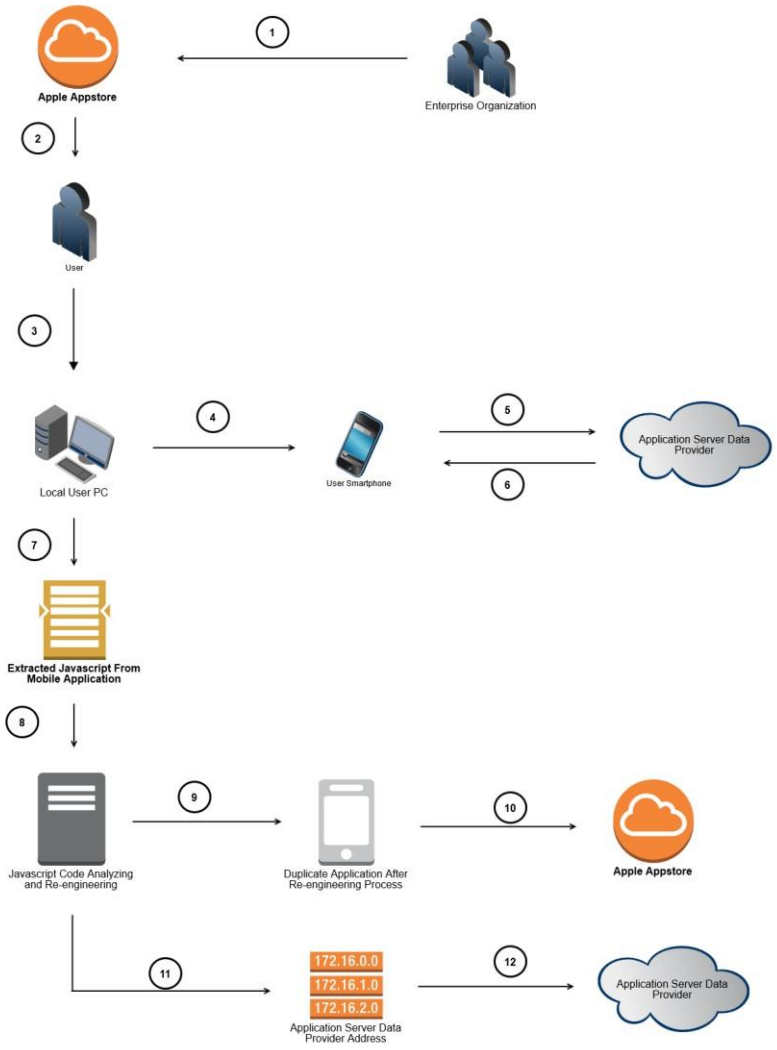
Untuk proses dekripsi dilakukan khusus oleh server dekripsi, hal ini dilakukan untuk mengurangi beban aplikasi mobile apabila dijalankan. Karena aplikasi *mobile* harus memiliki response yang cepat ketika user sedang menggunakannya.




**Gambar 3.2**. Perancangan aplikasi server dekriptor

Gambar 3.2 menjelaskan cara kerja server dekriptor ketika menerima *request* teks javascript yang terenkripsi dari aplikasi mobile dan ketika aplikasi mobile tersebut dijalankan. *Raw* teks javascript yang dikirimkan dikirim menggunakan javascript *sender* berbasis *ajax*. Setelah server menerima *raw* teks javascript yang terenkripsi, server melakukan dekripsi dan mengirimkan *raw* teks hasil enkripsi ke aplikasi *mobile* yang memberikan *request*. Aplikasi *mobile* yang mendapatkan *response* javascript yang telah didekripsi kemudian menjalankan sintaks javascipt untuk keperluan aplikasi yang menjalankannya.

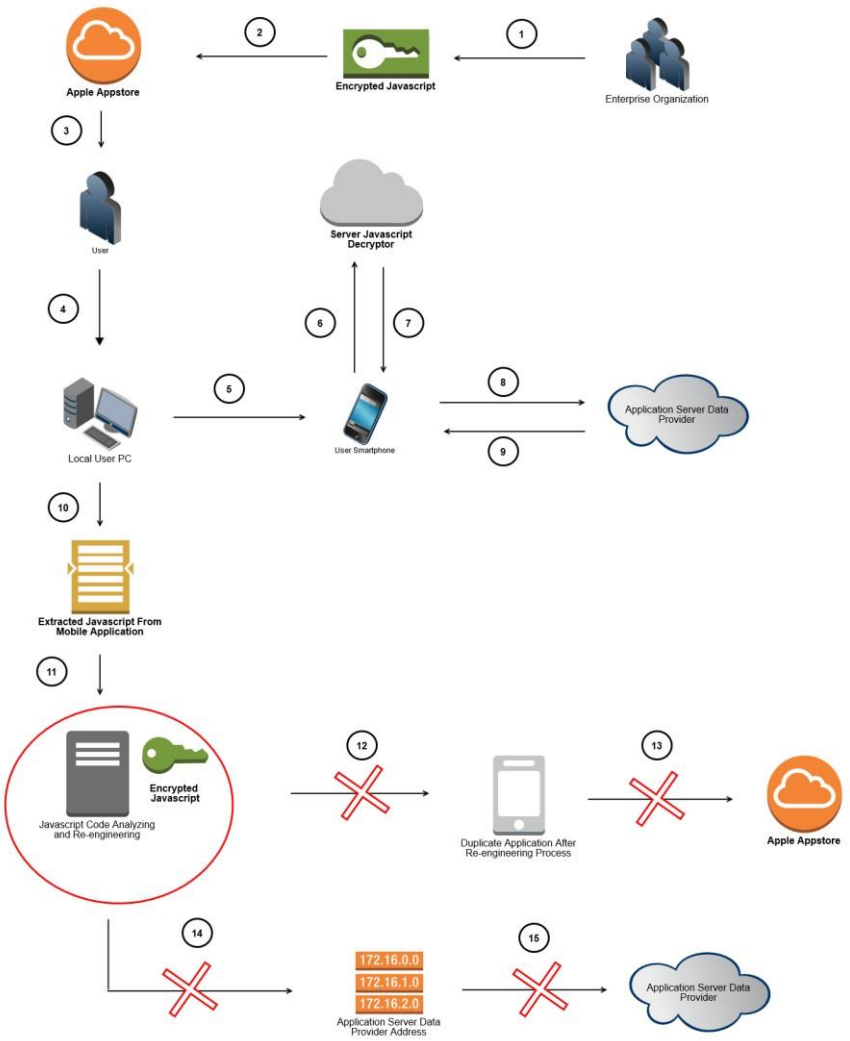
Berikut ini adalah gambaran sistem awal yang belum dilakukan enkripsi terhadap javascript yang terdapat pada aplikasi mobile.




**Gambar 3.3**. Gambaran sistem Sebelum Di Proteksi

Pada gambar 3.3 dijelaskan gambaran sistem sebelum dilakukan enkripsi terhadap sintaks javascript pada aplikasi *mobile.*
[1] Organisasi Enterprise / Perusahaan mem-*publish* aplikasinya ke *appstore*. *Appstore* merupakan portal aplikasi dimana user bisa mencari dan men-*download* aplikasi yang akan digunakan oleh *user* tersebut. [2]User men-*download* aplikasi tersebut dari *appstore*. [3]*User* menyimpan aplikasi yang telah di *download* di komputer *user*. [4]*User* melakukan instalasi ke perangkat *smartphone* yang dimiliki oleh *user* tersebut.[5]Aplikasi melakukan *request* data-data yang butuhkan oleh aplikasi tersebut dari server penyedia data yang disediakan oleh perusahaan penyedia aplikasi. [6]Aplikasi mendapatkan response data – data yang telah direquest sebelumnya. [7]User yang telah mendownload aplikasi dari appstore dan menyimpannya kemudian melakukan ekstraksi file yang telah di-*download*. Bisnis proses aplikasi yang terletak pada file javascript dapat dilihat.[8]*User* melakukan analisa terhadap javascript yang telah terbuka dan bisa dibaca. [9]*User* membuat aplikasi serupa dengan menggunakan proses *reverse-engineering*. [10]*User* mem-*publish* aplikasi ke *appstore*.[11]*User* mendapatkan informasi server serta *method-method* yang yang digunakan oleh aplikasi. [12] User melakukan manipulasi *method* dan parameternya agar server penyedia data memberikan data yang user tersebut tidak berhak untuk melihatnya.

Gambar 3.4 menunjukan gambaran sistem yang telah di proteksi dan menjelaskan tentang sistem yang telah dibuat dan dilakukan implementasi enkripsi pada javascript yang digunakan pada aplikasi mobile.



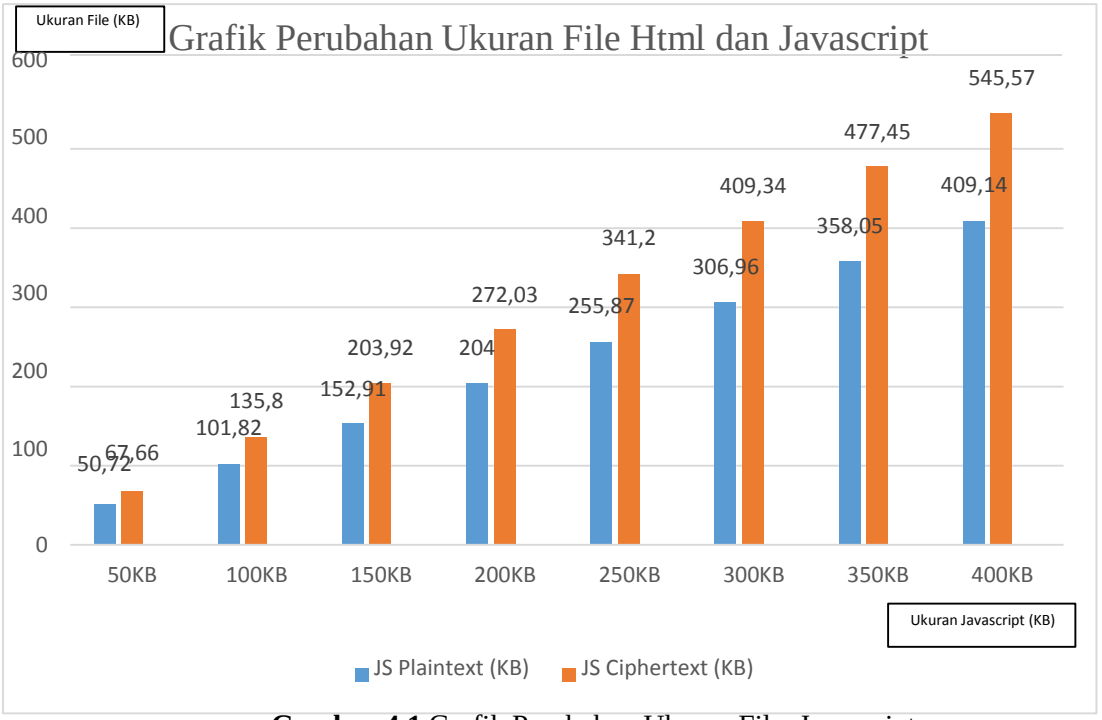

**Gambar 3.4**. Sistem Proteksi Javascript

Berikut ini adalah prosedur sistem proteksi javascript:
[1] Organisasi *Enterprise* / Perusahaan melakukan enkripsi terhadap javascript yang digunakan oleh aplikasi yang akan di *publish* di *appstore*. [2]Organisasi *Enterprise* / Perusahaan mem-*publish* aplikasinya ke *appstore*. [3]User men-*download* aplikasi tersebut dari *appstore*.[4]*User* menyimpan aplikasi yang telah di *download* di pc *user*. [5]*User* melakukan instalasi ke perangkat *smartphone*. [6]Ketika aplikasi dijalankan, aplikasi mengirimkan javascript yang terenkripsi ke server tertentu untuk melakukan proses dekripsi. [7]Hasil javascript yang telah di dekripsi diterima oleh aplikasi *mobile*. [8]Aplikasi melakukan *request* data-data yang butuhkan oleh aplikasi tersebut dari server penyedia data yang disediakan oleh perusahaan penyedia aplikasi menggunakan javascript yang telah dideksripsi. [9]Aplikasi mendapatkan response data-data yang telah di-*request* sebelumnya. [10]*User* yang telah men-*download* aplikasi dari *appstore* dan menyimpannya kemudian melakukan ekstraksi file yang telah di *download*. [11]*User* tidak dapat melakukan analisa terhadap javascript karena javascript tersebut dalam kondisi ter-enkripsi.[12]*User* juga tidak dapat membuat aplikasi baru yang serupa dengan proses *reverse-engineering*. [13]User tidak dapat mem-*publish* aplikasi *hasil reverse-engineering* karena tidak ada aplikasi didapatkan dari proses *reverse-engineering*. [14]User tidak mendapatkan informasi server

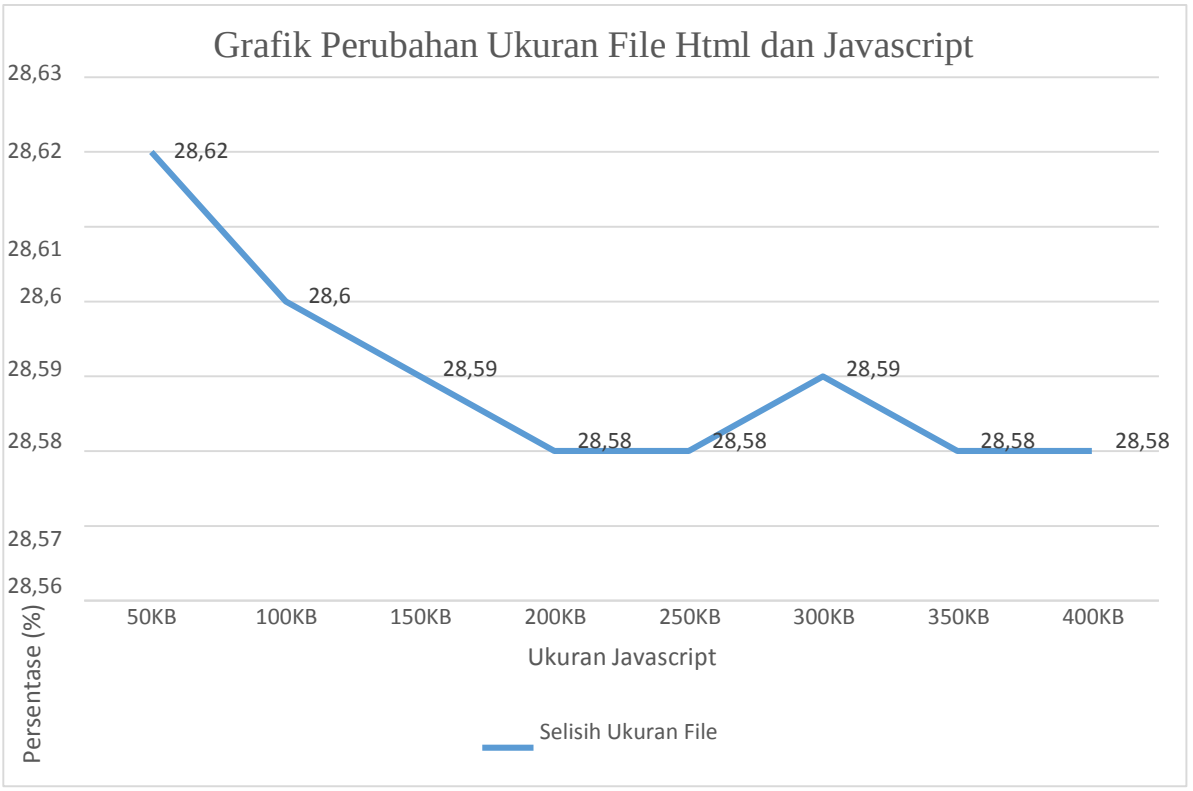
serta *method-method* yang yang digunakan oleh aplikasi.[15]*User* tidak dapat melakukan manipulasi *method* dan parameternya agar server penyedia data memberikan data yang user tersebut tidak berhak untuk melihatnya.

## 4. HASIL DAN PEMBAHASAN

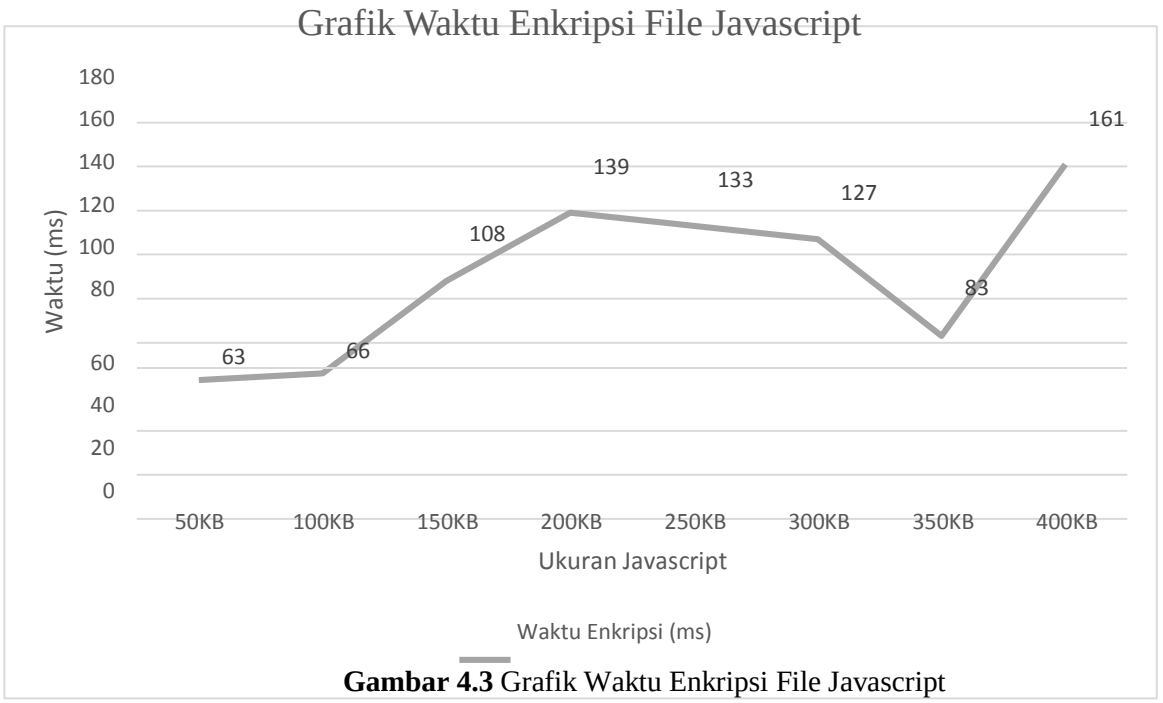
### 4.1 Hasil Pengujian Waktu Dan Perubahan Ukuran Javacript




**Gambar 4.1** Grafik Perubahan Ukuran File Javascript



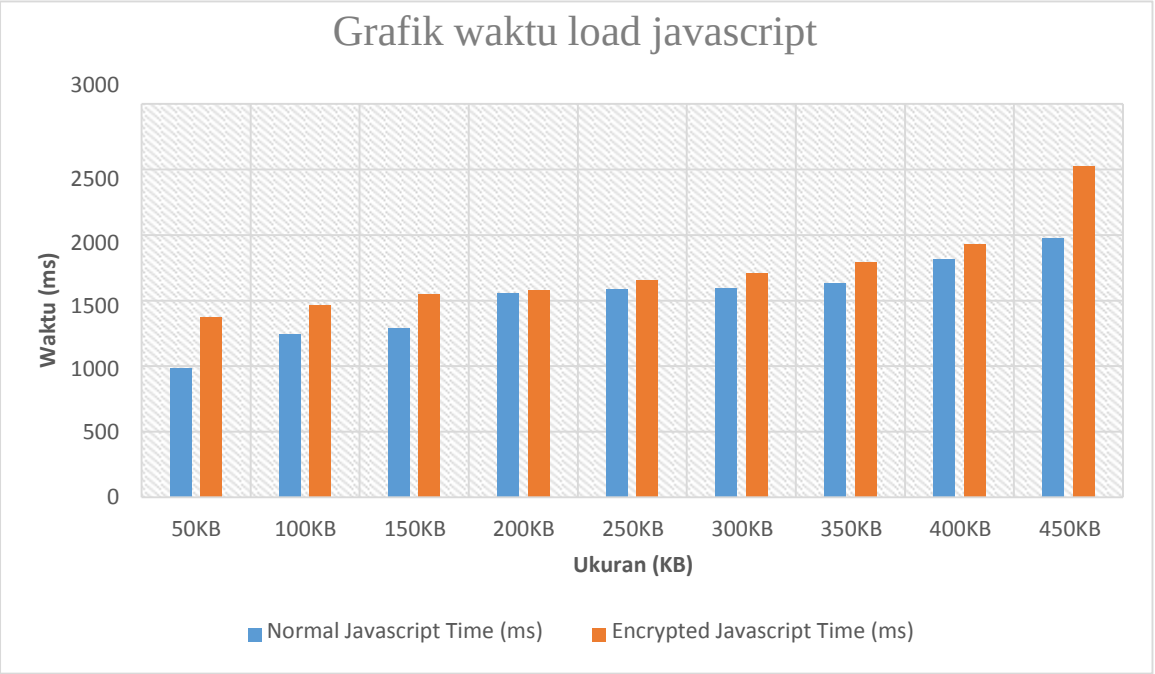

**Gambar 4.2** Grafik Perubahan Ukuran File Javascript



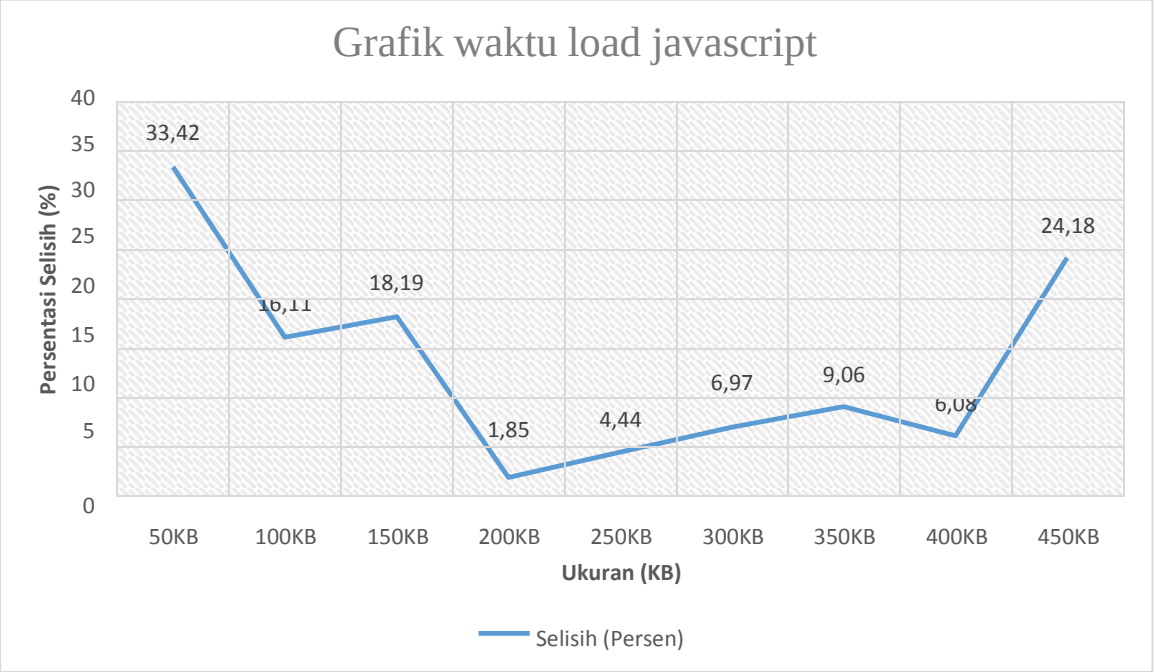

**Gambar 4.3** Grafik Waktu Enkripsi File Javascript

Dari grafik 4.1 dan 4.2 dapat diambil kesimpulan bahwa perubahan ukuran javascript yang belum dienkripsi dan ukuran javascript setelah di enkripsi memiliki perubahan yang tidak signifikan. Hal ini ini karena proses enkripsi menggunakan AES 128bit. Hasil mungkin dapat berbeda untuk AES 192bit ataupun AES 256bit. AES 128bit cocok karena selain memiliki keamaaman yang dibutuhkan oleh file javascript yang ada pada aplikasi mobile AES 128bit juga dapat mengenkripsi data text javascript dan menghasilkan output yang ukuran perbedaan dengan kondisi sebelum enkripsi yang tidak terlampau jauh. Ukuran diperhatikan karena aplikasi *mobile* yang akan melakukan instalasi aplikasi ini memiliki ukuran memori penyimpanan terbatas. Untuk gambar 4.3 waktu enkripsi yang berubah ubah dipengaruhi oleh jaringan komunikasi pertukaran data yang dilakukan aplikasi enkripsi dengan server enkripsi.

### 4.2 Hasil Pengujian Lama Waktu Aplikasi Mobile Dijalankan




**Gambar 4.4** Grafik perubahan waktu javascript 50KB sampai 450KB




**Gambar 4.5** Grafik perubahan waktu javascript 50KB sampai 450KB

Berdasarkan grafik 4.4 dan 4.5 perubahan terbesar terjadi pada javascript berukuran 50KB, perubahan terbesar dapat dipengaruhi oleh berbagai macam hal. Karena proses dekripsi dilakukan di server dan aplikasi mobile hanya mengirimkan *sourcecode* yang terenkripsi ke server dekripsi maka pengaruh jaringan yang digunakan untuk mengirimkan *sourcecode* javascript memiliki pengaruh. Bukan hanya ketika mengirimkan javascript ke server, tuntuk mengakses *webservice* ramalan cuaca juga memerlukan jaringan yang baik untuk melakukan proses pertukaran data antara aplikasi *mobile* dan server. Untuk ukuran file javascript 50KB sampai dengan 250KB tidak memiliki perubahan waktu yang signifikan ketika aplikasi dijalankan. Berarti dengan javascript yang terenkripsi didalamnya selain memberikan proteksi keamanan yang tinggi, sumber daya waktu juga tidak memiliki perbedaan yang terlampau jauh ketika aplikasi dijalankan.

Untuk file javascript yang berukuran 300KB sampai dengan 450KB perubahan yang paling besar terdapat pada ukuran javascript maksimal yang digunakan yaitu 450KB. Ada dua faktor penyebab perbedaan waktu ini. Yang pertama bisa diakibatkan oleh tidak stabilnya jaringan komunikasi pertukaran data antara aplikasi mobile dan server dekripsi. Yang kedua bisa dikarenakan algoritma AES 128bit kurang cocok untuk file text javascript dengan ukuram javascript 450KB atau lebih. Tetapi untuk ukuran file dibawah 450KB waktu yang diperlukan untuk melakukan proses dekripsi javascript di server masih terbilang kecil. Oleh karena itu dapat disimpulkan bahwa AES 128 cocok untuk melakukan dekripsi pada file yang berukuran kurang dari 450KB.

### 4.3 Pengujian Keamanan Algoritma Enkripsi AES 128bit (5 *Round*)

Pengujian keamanan algoritma AES 128bit dilakukan dengan menggunakan dua metode serangan (*attack*) yang biasa digunakan untuk menyerang algoritma enkripsi yang ada saat ini. Adapun tipe seragan yang akan digunakan adalah *Differential attack* dan *brute force attack*.

### 4.3.1 Differential Attack / Differential Crypanalysis

Differential crypanalysis atau yang biasa disebut dengan analisis sandi diferensial merupakan suatu metode analisa yang digunakan pada algoritma kriptografi untuk memahami pola perbedaan yang terbentuk oleh sepasang plaintext dan dibandingkan dengan sepasang ciphertext yang dihasilkan ketika proses enkripsi dilakukan. Perbandandingan yang dilakukan pada sepasang plaintext dan sepasang chippertext digunakan untuk menghasilkan nilai probabilitas dari *key* yang mungkin dapat ditemukan. Plaintext yang digunakan sebagai input akan di letakan pada state matriks 4x4. Tiap satu kolom state matriks memiliki kapasitas berukuran 8 bit.

| | | | |
|---|---|---|---|
| $a_{0,0}$ | $a_{0,1}$ | $a_{0,2}$ | $a_{0,3}$ |
| $a_{1,0}$ | $a_{1,1}$ | $a_{1,2}$ | $a_{1,3}$ |
| $a_{2,0}$ | $a_{2,1}$ | $a_{2,2}$ | $a_{2,3}$ |
| $a_{3,0}$ | $a_{3,1}$ | $a_{3,2}$ | $a_{3,3}$ |

**Gambar 4.6 State Matriks 4x4**

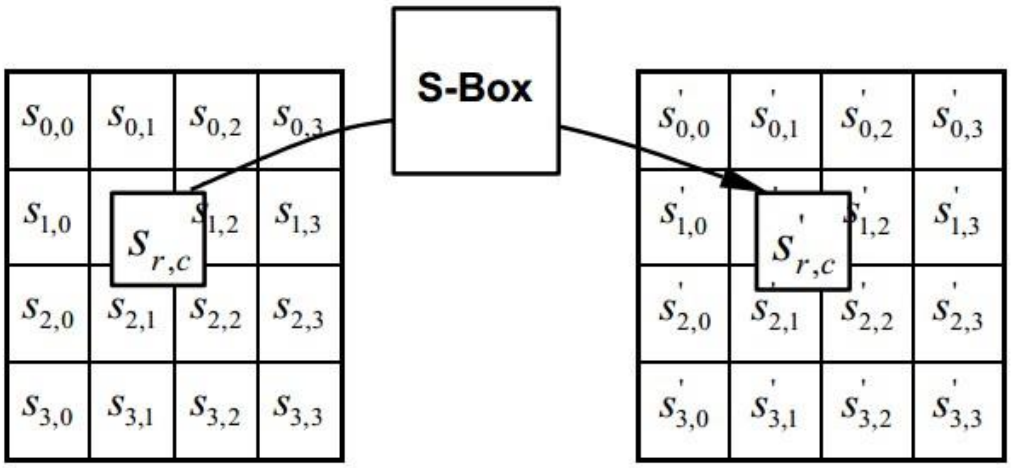
Pada proses sub bytes kolom kolom tertentu pada state matriks secara random akan mengalami perubahan nilai. Nilai yang berubah ini diakibatkan oleh operasi subtitusi antara state matriks dan rijndael s-box. Rijndael s-box merupakan matriks yang berisi nilai nilai yang digunakan sebagai lookup table (tabel acuan subtitusi) pada algoritma aes.

| | | $y$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f |
| $x$ | 0 | 63 | 7C | 77 | 7B | F2 | 6B | 6F | C5 | 30 | 01 | 67 | 2B | FE | D7 | AB | 76 |
| | 1 | CA | 82 | C9 | 7D | FA | 59 | 47 | F0 | AD | D4 | A2 | AF | 9C | A4 | 72 | C0 |
| | 2 | B7 | FD | 93 | 26 | 36 | 3F | F7 | CC | 34 | A5 | E5 | F1 | 71 | D8 | 31 | 15 |
| | 3 | 04 | C7 | 23 | C3 | 18 | 96 | 05 | 9A | 07 | 12 | 80 | E2 | EB | 27 | B2 | 75 |
| | 4 | 09 | 83 | 2C | 1A | 1B | 6E | 5A | A0 | 52 | 3B | D6 | B3 | 29 | E3 | 2F | 84 |
| | 5 | 53 | D1 | 00 | ED | 20 | FC | B1 | 5B | 6A | CB | BE | 39 | 4A | 4C | 58 | CF |
| | 6 | D0 | EF | AA | FB | 43 | 4D | 33 | 85 | 45 | F9 | 02 | 7F | 50 | 3C | 9F | A8 |
| | 7 | 51 | A3 | 40 | 8F | 92 | 9D | 38 | F5 | BC | B6 | DA | 21 | 10 | FF | F3 | D2 |
| | 8 | CD | 0C | 13 | EC | 5F | 97 | 44 | 17 | C4 | A7 | 7E | 3D | 64 | 5D | 19 | 73 |
| | 9 | 60 | 81 | 4F | DC | 22 | 2A | 90 | 88 | 46 | EE | B8 | 14 | DE | 5E | 0B | DB |
| | A | E0 | 32 | 3A | 0A | 49 | 06 | 24 | 5C | C2 | D3 | AC | 62 | 91 | 95 | E4 | 79 |
| | B | E7 | C8 | 37 | 6D | 8D | D5 | 4E | A9 | 6C | 56 | F4 | EA | 65 | 7A | AE | 08 |
| | C | BA | 78 | 25 | 2E | 1C | A6 | B4 | C6 | E8 | DD | 74 | 1F | 4B | BD | 8B | 8A |
| | D | 70 | 3E | B5 | 66 | 48 | 03 | F6 | 0E | 61 | 35 | 57 | B9 | 86 | C1 | 1D | 9E |
| | E | E1 | F8 | 98 | 11 | 69 | D9 | 8E | 94 | 9B | 1E | 87 | E9 | CE | 55 | 28 | DF |
| | F | 8C | A1 | 89 | 0D | BF | E6 | 42 | 68 | 41 | 99 | 2D | 0F | B0 | 54 | BB | 16 |

**Gambar 4.7 Matriks Rijndael S-box**

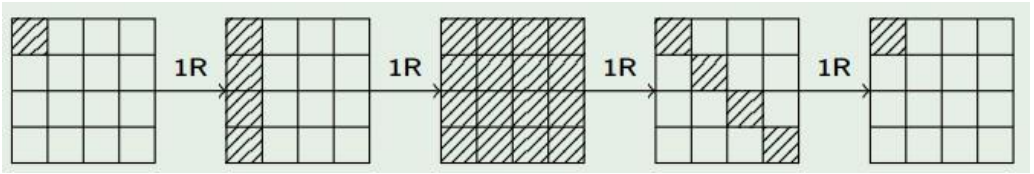
Nilai nilai yang terdapat didalam matriks rijndael s-box di representasikan dalam format bilangan heksadesimal. Bilangan heksadesimal ini yang nantinya akan disubtitusikan dengan nilai nilai yang

terdapat pada kolom di state matriks. Misalkan dalam kolom 1 dan baris 1 ($S_{1,1}$) state matriks memiliki nilai 89, contoh nilai 2 digit ini bukanlah nilai sembarang, karena didalam satu kolom state matriks berukuran 8 bit apabila nilai 89 dikonversi menjadi biner maka akan membentuk 4 bit pertama dari 8 yaitu 1000 dan membentuk 4 bit kedua dari 9 yaitu 1001 dengan total bit 8. Karena kolom state matriks ini memiliki nilai 8 dan 9 maka kita mencari nilai untuk subtitusinya pada s-box yaitu pada baris dengan angka 8 dan kolom dengan angka 9 maka bilangan subtitusi yang didapatkan untuk mensubtitusi state matriks $S_{1,1}$ adalah A7.




**Gambar 4.8 Proses Subtitusi State Matriks Mengunakan S-Box.**

Gambar kotak disebelah kiri merupakan state matriks sebelum dilakukan subtitusi dengan s-box dan gambar kotak disebelah kanan merupakan gambar hasil operasi subtitusi state matriks dengan s-box. Tidak semua kolom pada tiap round mengalami subtitusi dengan sbox. Tiap state matriks memiliki probabilitas yang berubah tiap state. Probabilitas ini dapat dilihat dari berapa banyak jumlah S-Box yang aktif pada setiap round dari awal enkripsi sampai dengan akhir enkripsi. Sebagai contoh untuk algoritma AES yang yang direduksi *round* nya dijelaskan pada gambar 3.




**Gambar 4.9 S-Box Aktif Pada Algoritma AES** .

S-box aktif yang digunakan oleh algoritma aes ketika melakukan proses enkripsi memiliki perubahan pada setiap round nya. Pada aes yang menggunakan 4 round memiliki jumlah s-box yang aktif yaitu 25 (Minimal) . S-box yang aktif ditandai dengan kolom yang diarsir pada state matriks 4x4 yang digunakan. Setiap S-box pada aes memiliki probabilitas maksimal $P_{max} = 2^{-6}$ dan karena s-box yang aktif yaitu 25 maka probabilitas maksimal dari seluruh s-box aktif yang digunakan adalah $(2^{-6})^{25} = 2^{-150}$ atau dengan kata lain diperlukan sebanyak $2^{150}$ pasangan plaintext untuk dapat menemukan kunci yang digunakan. Hal ini melebihi jumlah pasangan plaintext yang dapat dibentuk oleh aes 128 bit yaitu $2^{128}$. Untuk ujicoba pada aes dengan 5 round maka jumlah s-box yang aktif adalah 26 s-box aktif

(berdasarkan gambar 4.10). Maka untuk menghitung peluang differensial maksimal nya adalah $(2^{-6})^{26}$ = $2^{-156}$ jadi diperlukan sebanyak $2^{156}$ pasangan plaintext yang diperlukan untuk dapat menemukan kunci yang tepat, **sehingga AES dengan 5 ronde dapat kebal terhadap serangan analisis sandi diferensial (Differential Cryptanalysis)**. Ukuran minimal dari s-box yang aktif pada aes 128 bit dapat dilihat pada gambar 2.

| Rounds | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| min | 1 | 5 | 9 | 25 | 26 | 30 | 34 | 50 | 51 | 55 |

**Gambar 4.10 Perbandingan jumlah round AES terhadap s-box aktif.**

Nilai jumlah s-box yang aktif pada setiap round aes yang semakin meningkat menyebabkan meningkatnya pula probablitas dari pasangan plaintext yang digunakan untuk untuk dapat menentukan kunci yang akan ditebak.

### 4.3.2 Brute Force Attack

Brute force attack menggunakan algoritma exhaustive search untuk dapat menemukan kunci yang tepat dari algoritma AES. Bruto force attack melakukan pengecekan satu persatu dari kemungkinan kombinasi yang mungkin ditemukan. Kombinasi yang dapat dibentuk dari 128-bit AES adalah 3.4 x $10^{38}$ dan apabila kita mengasumsikan bahwa ada komputer super cepat yang melakukan pemrosesan 10.51 Pentaflops atau 10.51 x $10^{15}$ Flops. Flops (floating point operation per second) adalah operasi *floating point* yang bisa dioperasikan oleh komputer per detik. Jumlah *flops* dalam super komputer ini diasumsikan memiliki 1000 *flops*. Untuk dapat melakukan pengecekan kombinasi kunci yang tepat perdetik adalah (10.51 x $10^{15}$) / 1000 = 10.51 x $10^{12}$ dan apabila jumlah detik dalam setahun adalah 365 x 24 x 60 x 60 = 31536000 maka jumlah tahun yang dibutuhkan untuk melakukan proses *cracking* kombinasi semua kunci yang ada pada algoritma AES 128bit adalah sebagai berikut.

= (3.4 x $10^{38}$) / [(10.51 x $10^{12}$) x 31536000]
= (0.323 x $10^{26}$)/31536000
= 1.02 x $10^{18}$ Tahun

Jumlah tahun yang sangat besar ini mengindikasikan bahwa membutuhkan waktu yang sangat lama untuk dapat membobol algoritma AES 128bit. Peluang kunci mungkin saja bisa di temukan tetapi **jumlah waktu yang sangat besar ini menjadikan algoritma AES kebal terhadap serangan brute force.**

## 5. KESIMPULAN

Berdasarkan pembahasan implementasi dan analisa yang telah dikemukakan dalam tugas akhir ini dapat disimpulkan bahwa:

Dengan diimplementasikannya sistem enkripsi javascript menggunakan algoritma AES 128bit pada aplikasi *mobile* berbasis phonegap maka javascript yang ada didalam aplikasi *mobile* terlindungi dari proses *reverse-engineering* proses bisnis yang terdapat di dalam sintaks javascript. Aplikasi *mobile* juga menjadi lebih aman ketika mengakses *webservice* dari penyedia layanan *webservice* tertentu karena javascript untuk melakukan akses *webservice* terlindungi.

Dari segi keamanan, algoritma enkripsi AES 128bit kebal terhadap analisis sandi differensial (*differential attack*) dan *bruteforce attack*. Berdasarkan hasil pengujian keamanan terhadap *differential attack* terhadap 5 ronde AES 128bit dibutuhkan $2^{156}$ pasang plaintext untuk mendapatkan *key* yang tepat. Jumlah ini merupakan jumlah yang sangat besar baik dari segi waktu dan memori pemrosesan komputer saat ini sehingga AES 128bit kebal terhadap *differential attack*. Algoritma enkripsi AES 128bit dinyatakan cukup aman untuk mengamankan javascript pada aplikasi *mobile* dari *bruteforce attack* karena dibutuhkan milyaran tahun untuk bisa menemukan kunci yang tepat untuk dapat membuka javascript yang terenkripsi.

Algoritma AES 128bit cocok digunakan untuk mengenkripsi file javascript berukuran dibawah 450KB pada aplikasi *mobile* agar perbedaan waktu load ketika aplikasi *mobile* dijalankan tidak berbeda jauh dengan aplikasi *mobile* yang menggunakan javascript yang tdak terenkripsi. Lama waktu ketika aplikasi *mobile* dijalankan tergantung pada stabilitas jaringan untuk proses pertukan javascript dari client ke server dan bergantung juga pada besar kecil ukuran file javascript itu sendiri.

Lama waktu aplikasi *mobile* ketika dijalankan dalam keadaan jaringan yang normal membutuhkan waktu sedikit lebih lama karena aplikasi melakukan proses dekripsi javascript di server, tetapi perbedaan waktu antara javascript yang tanpa enkripsi dengan javascript yang telah dienkripsi tidak berbeda jauh.

Perubahan ukuran javascript yang belum di enkripsi dengan ukuran javascript yang telah dienkripsi tidak signifikan. Hal ini bagus karena javascript nantinya akan digunakan pada aplikasi *mobile* bersistem operasi android yang memiliki memori penyimpanan terbatas.

## DAFTAR PUSTAKA

[1] Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Springer (2002)

[2] Alan Kaminsky, Michael Kurdziel, Stanisław Radziszowski "An Overview of Cryptanalysis Research for the Advanced Encryption Standard".Rochester Institute of Technology; Rochester.

[3] Ross Anderson, Eli Biham, Lars Knudsen "Serpent: A Proposal for the Advanced Encryption Standard". Cambridge University; England.

[4] Biham, E. & Shamir, A., Differential Cryptanalysis of DES-like Cryptosystems, In Advances in Cryptology: Journal of Cryptology, Vol. 4, No. 1, pp. 3-72 (1991)

[5] Samir El Adib, Naoufal Raissouni ” AES Encryption Algorithm Hardware Implementation: Throughput and Area Comparison of 128, 192 and 256-bits Key” University Abdelmalek Essaad; Morocco.

[6] Phonegap.”Platform Security” . November 2014.

https://github.com/phonegap/phonegap/wiki/Platform-Security

[7] Marisa Peacock, “Trends in Enterprise Mobile App Development [Infographic]”

http://www.cmswire.com/cms/mobile-enterprise/trends-in-enterprise-mobile-app-development-infographic-021895.php, Juli 2013.

[8] Sarah Perez, ”Mobile App Usage Increases In 2014, As Mobile Web Surfing Declines”

http://techcrunch.com/2014/04/01/mobile-app-usage-increases-in-2014-as-mobile-web-surfing-declines/. April 2014.

[9] Adimas Fiqri Ramdhansya, "Implementasi Advance Encryption Standard (AES) Pada Sistem Kunci Elektronik Kendaraan Berbasis Sistem Operasi Android Dan Mikrokontroler Arduino" . Telkom University; Bandung.

[10] Yusuf Kurniawan, Adang Suwandi Ahmad, M. Sukrisno Mardiyanto, Iping Supriana, Sarwono Sutikno, "Analisis Sandi Diferensial terhadap AES, DES dan AE1". Institut Teknologi Bandung; Bandung.