****

**Onderzoek handmatig AES toepassen**

Naam: Bart Janssen

Studentennummer: 389419

Datum: 14 december 2020

# **Inleiding**

Het AES-algoritme is een algoritme wat gebruikt wordt om bestanden en ander dataverkeer te versleutelen zodat de integriteit van de data bewaard blijft. Het AES-algoritme is een complex algoritme en daarom is het een uitdaging om verder onderzoek te doen naar hoe dit algoritme inhoudelijk werkt en waarom dit zo krachtig is. In de minor van de cyber security opleiding van Fontys hogeschool te Eindhoven zijn er verschillende uitdagingen opgesteld om de kennis van studenten op het gebied van cyber security te verbreden. Betreft het onderwerp AES zijn er twee uitdagingen geformuleerd. Eén om het algoritme in een programmeertaal naar keuze te implementeren, de andere om dit algoritme beter te leren kennen en de stappen handmatig toe te passen. Dit onderzoek betreft alleen het onderzoeken van het handmatig toepassen.

**Inhoudsopgave**

[**Inleiding** 2](#_Toc60907737)

[**1 Probleemstelling** 4](#_Toc60907738)

[**2 Doelstelling** 5](#_Toc60907739)

[**3 Onderzoeksmethoden** 6](#_Toc60907740)

[**4 Hoofd- en deelvragen** 7](#_Toc60907741)

[**4.1 Hoofdvraag** 7](#_Toc60907742)

[**4.2 Deelvragen** 7](#_Toc60907743)

[**5 Het onderzoek** 8](#_Toc60907744)

[**5.1 Hoe werkt het AES-algoritme?** 8](#_Toc60907745)

[**5.1.1 Het algoritme** 8](#_Toc60907746)

[**5.1.2 KeyExpansion** 10](#_Toc60907747)

[**5.1.3 AddRoundKey** 14](#_Toc60907748)

[**5.1.4 SubBytes** 15](#_Toc60907749)

[**5.1.5 ShiftRows** 16](#_Toc60907750)

[**5.1.6 MixColumns** 17](#_Toc60907751)

[**5.2 Welke voorbeelden voor het handmatig toepassen bestaan er al?** 21](#_Toc60907752)

[**5.3 Wat voor methodes kunnen gebruikt worden om het toe te passen?** 22](#_Toc60907753)

[**6 Conclusie** 23](#_Toc60907754)

[**7 Aanbeveling** 24](#_Toc60907755)

[**8 Planning** 25](#_Toc60907756)

[**Literatuurlijst** 26](#_Toc60907757)

[**Afkortingen en woordenlijst** 27](#_Toc60907758)

[**Bijlagen** 28](#_Toc60907759)

# **1 Probleemstelling**

Het AES-algoritme bestaat uit verschillende stappen om tot het versleutelde resultaat te komen. Al deze stappen hebben een eigen werking waar ook weer een verschillende moeilijkheidsgraad aan vast zit. Het is dan de vraag hoe deze stappen handmatig toegepast kunnen worden voor educatieve doeleinden.

Het onderzoek is relevant voor de minor van de cyber security opleiding van Fontys hogeschool te Eindhoven omdat het toepassen van AES een leeronderdeel is. Om dit leeronderdeel te kunnen toepassen, is het wel belangrijk om te weten wat voor manieren geschikt zijn om het AES-algoritme handmatig toe te kunnen passen. Er zijn meerdere manieren om het AES-algoritme toe te passen en te bestuderen, alleen mogelijk is de ene manier makkelijker dan een andere manier.

# **2 Doelstelling**

Het doel van dit onderzoek is een manier vinden hoe het AES-algoritme handmatig toegepast kan worden zodat hiervan geleerd kan worden en zodat de uitdaging op een goede manier uitgevoerd kan worden. Waarmee het belangrijkste doel, het leren en beter begrijpen van de inhoudelijke werking van het algoritme is. Het op te leveren product is een aanbeveling zijn zodat mensen die dit willen leren, beter een idee hebben van waar ze zouden moeten beginnen mocht iemand hier meer kennis van willen opdoen. Ook wordt er onderzocht of er verschillende manieren zijn om tot hetzelfde resultaat te komen en om in kaart te brengen welke gemakkelijker is om mee te beginnen. Dit onderzoek is puur gericht op educatieve ontwikkeling.

# **3 Onderzoeksmethoden**

Binnen dit onderzoek wordt gebruik gemaakt van een hoofdvraag met deelvragen. De deelvragen worden onderzocht en beantwoord om de hoofdvraag te kunnen beantwoorden. Per deelvraag wordt er onderzoek gedaan om dat specifieke gedeelte van die deelvraag te kunnen onderzoeken en beantwoorden. De deelvragen worden onderzocht met de onderzoeksmethoden uit het DOT-framework, deze bestaan uit de Field, Library, Workshop, Lab en Showroom strategieën.

* De **Field** strategie kan gebruikt worden om te onderzoeken wie of wat met dit onderzoek te maken heeft en zo een inzicht te krijgen wat voor deze groep/persoon belangrijk is of wat er nodig is en waarom dit product van belang is;
* De **Library** strategie kan gebruikt worden om te onderzoeken wat al gedaan is qua dit onderwerp en of hier al iets van hergebruikt kan worden, of juist nieuwe technieken van kunnen worden geleerd. Dit wordt meestal gedaan met behulp van bijvoorbeeld een internet methode;
* De **Lab** strategie is bedoeld om bijvoorbeeld een concept of prototype van het onderwerp te testen, hieruit kan dan blijken dat er bijvoorbeeld nog veranderingen nodig kunnen zijn;
* De **Workshop** strategie wordt gebruikt om de ideeën uit te werken in een werkend prototype. Hieruit kan dan weer blijken dat er misschien veranderingen nodig zijn;
* De **Showroom** strategie wordt gebruikt om je ideeën te laten zien. Dit kan ook een prototype zijn zodat iemand zoals de klant een beeld krijgt van hoe het eruit gaat zien en van hoe het gaat werken.

# **4 Hoofd- en deelvragen**

De onderzoeksvragen zijn opgedeeld in een hoofdvraag met deelvragen. De hoofdvraag bevat het project in één zin. Met het onderzoeken en beantwoorden van de deelvragen, wordt de hoofdvraag beantwoordt.

## **4.1 Hoofdvraag**

* Welke manieren zijn geschikt om het AES-algoritme handmatig te kunnen toepassen?

## **4.2 Deelvragen**

Hieronder staan de opgestelde deelvragen vermeld met per deelvraag beschreven welke strategieën er gebruikt gaan worden.

**4.2.1 Hoe werkt het AES-algoritme?**

Deze onderzoeksvraag onderzoekt hoe het AES-algoritme inhoudelijk werkt.

**Onderzoekmethodes:**

* Met de Library strategie kan er onderzocht worden hoe het werkt. De methodes kunnen video’s hierover bestuderen of informatie op het internet zoeken zijn.

**Tijdschatting:**

* Mijn tijdschatting voor deze deelvraag zou op ongeveer een dag komen.

**4.2.2 Welke voorbeelden voor het handmatig toepassen bestaan er al?**

Deze deelvraag worden al bestaande voorbeelden onderzocht.

**Onderzoekmethodes:**

* Met de Library strategie kan er onderzocht worden welke voorbeelden er al bestaan. De methodes hiervoor zullen voornamelijk het opzoeken op internet zijn om bestaande voorbeelden te kunnen onderzoeken en deze te begrijpen.

**Tijdschatting:**

* Mijn tijdschatting voor deze deelvraag zou een tot twee dagen zijn.

**4.3.3 Wat voor methodes kunnen gebruikt worden om het toe te passen?**

Deze deelvraag gaat in op de methodes die gebruikt kunnen worden.

**Onderzoekmethodes:**

* Met de Workshop strategie kunnen prototypes gemaakt worden om te onderzoeken welke manier het meest geschikt is. De methodes zullen afgangen van de voorbeelden.

**Tijdschatting:**

* Mijn tijdschatting voor deze deelvraag gaat naar een week.

# **5 Het onderzoek**

De deelvragen beschreven in hoofdstuk 4.2, worden in dit hoofdstuk onderzocht. De hoofdvraag die eerder geformuleerd is, is gebaseerd op de probleemstelling om als resultaat een oplossing te kunnen vinden voor het probleem.

## **5.1 Hoe werkt het AES-algoritme?**

Deze deelvraag betreft hoe het AES-algoritme zelf inhoudelijk werkt. Binnen deze deelvraag wordt er dieper ingegaan op de werking ervan en welke stappen nodig zijn om de data correct te kunnen versleutelen en decoderen. Dit hoofdstuk is erg technisch en er wordt zeer gedetailleerd op de stappen van het algoritme ingegaan.

### **5.1.1 Het algoritme**

Om te beginnen heeft het AES-algoritme een sleutel nodig, deze sleutel is te vergelijken met een wachtwoord waarmee het mogelijk is om de data te versleutelen en decoderen. De sleutel kan variëren tussen een 128, 192 of 256 bit sleutel. Een 256 bit sleutel wordt tegenwoordig aangeraden. Eén byte is 8 bit, dus 256 / 8 zijn 32 bytes. Eén karakter is gelijk aan één byte, een 256 bit sleutel is dus 32 karakters lang.

Het algoritme bestaat uit vier stappen waarvan één op basis van de sleutel lengte herhaaldelijk uitgevoerd wordt. Bij een lengte van 128 bit word deze 10 keer herhaald, een lengte van 192 bit 12 keer en bij 256 bit 14 keer.

**De te versleutelen tekst**

De te versleutelen tekst wordt opgedeeld in segmenten van 128 bit (16 karakters).

Stel de te versleutelen tekst is “**Dit is een geheime tekst dat versleuteld moet worden!**”, dan wordt deze verdeeld in de volgende segmenten:

* Dit is een gehei
* me tekst dat ver
* sleuteld moet w
* orden!

Het laatste segment is geen 16 karakters (128 bit) lang, dit wordt vaak met een padding opgevuld.

Hierna worden de volgende stappen per te versleutelen tekst segment uitgevoerd.

* **KeyExpansion** – hiermee wordt de sleutel omgezet naar sub sleutels die per ronde gebruikt worden;
* **AddRoundKey** – hiermee wordt de sleutel toegevoegd aan de te versleutelen tekst;

De volgende vier stappen worden **9**, **11** of **13** keer uitgevoerd afhankelijk van de sleutellengte:

* **SubBytes** – Bij deze stap worden bytes vervangen met een byte met behulp van een opzoektabel;
* **ShiftRows** – Bij deze stap worden de rijen van een tekstsegment een plekje opgeschoven, dat per segmentrij opgeteld wordt;
* **MixColumns** – Bij deze stap worden de bytes samengevoegd met een voor gedefinieerde tabel;
* **AddRoundKey** – Bij deze stap wordt de sub sleutel weer toegevoegd.
* De laatste ronde, wat een totaal maakt van **10**, **12** of **14** rondes, bestaat uit de volgende onderdelen:
  + **SubBytes**;
  + **ShiftRows**;
  + **AddRoundKey**.

Voor het decoderen worden dezelfde stappen precies in tegenovergestelde richting herhaald, in de MixColumns stap, wordt dan ook de tabel gebruikt voor het decoderen.

### **5.1.2 KeyExpansion**

Bij de KeyExpansion stap wordt de sleutel omgezet naar sub sleutels zodat iedere ronde een eigen sleutel heeft. Als voorbeeld wordt er een sleutel gebruikt van 128 bit, de sleutel is als volgt:

**GR96dzKYZP73hbUG**

Deze sleutel kan als hexadecimaal geschreven worden als:

**47 52 39 36 64 7A 4B 59 5A 50 37 33 68 62 55 47**

Deze hexadecimale waardes kunnen als bytes vernoemd worden met een **bx** notatie als:

**b1 b2 b3 b4 b5 b6 b7 b8 b9 b10 b11 b12 b13 b14 b15 b16**

AES gebruikt voor bijna alle stappen een grit. In een grit zou de sleutel er als volgt uitzien:

|  |  |  |  |
| --- | --- | --- | --- |
| b1 | b5 | b9 | b13 |
| b2 | b6 | b10 | b14 |
| b3 | b7 | b11 | b15 |
| b4 | b8 | b12 | b16 |

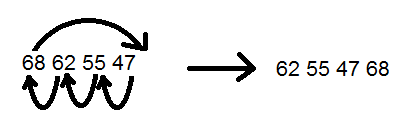
Iedere verticale rij wordt een ‘Word’ genoemd. Deze vier rijen worden benoemd als Word0, Word1, Word2 en Word3. Deze Words worden uitbereid naar 44 Words. Ieder blokje van 4 Words wordt gebruikt om het volgende blokje te bepalen. Ieder blokje van 4 Words heet een RoundKey, Die sleutel wordt gebruikt voor die ronde in het algoritme, de onderstaande tabel is de volledige sleutel in een grit, deze wordt gebruikt om de overige 40 sub sleutels te bepalen.

|  |  |  |  |
| --- | --- | --- | --- |
| W0 | W1 | W2 | W3 |
| 47 | 64 | 5A | 68 |
| 52 | 7A | 50 | 62 |
| 39 | 4B | 39 | 55 |
| 36 | 59 | 33 | 47 |

Om W4, W5, W6 en W7 te bepalen wordt er een functie gebruikt G(W3). Deze functie pakt de laatste Word per blokje (W3 in het eerste blokje) en voert het volgende uit:

|  |
| --- |
| W3 |
| 68 |
| 62 |
| 55 |
| 47 |

De Word hierboven is W3, dit is de laatste kolom van het eerste blokje. Deze gebruikt een circulaire een-byte left shift. Dit betekend dat alle bytes van dit blokje een worden opgeschoven. De onderstaande afbeelding laat dit zien van blok W3:

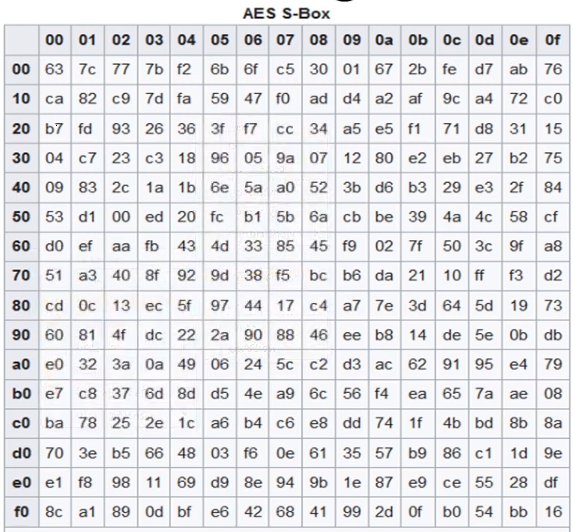


Afbeelding 1 Circulaire een-byte left shift

De uitkomst hiervan heet een RotWord en wordt benoemd als X1.

|  |
| --- |
| X1 |
| 62 |
| 55 |
| 47 |
| 68 |

Deze wordt nog met een voor gedefinieerde tabel vervangen. Deze tabel heet de S-Box en ziet eruit als volgt:



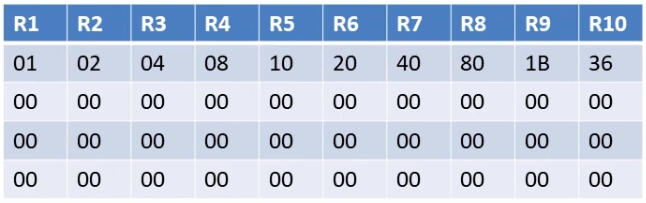
|  |
| --- |
| X1 |
| 62 |
| 55 |
| 47 |
| 68 |

De X1 wordt ook weer per byte in deze tabel opgezocht en vervangen. **62** wordt vervangen met **AA**, **55** met **FC**, **47** met **A0** en **68** met **45**.  
  
De uitkomst hiervan wordt een SubWord genoemd en is gedefinieerd als Y1.

|  |
| --- |
| Y1 |
| AA |
| FC |
| A0 |
| 45 |

Afbeelding 2 S-Box

Vervolgens wordt er een XOR-berekening uitgevoerd op de Y1 met een voor gedefinieerde tabel, de Round constant.



Afbeelding 3 Round constant

Bij iedere ronde (Round) wordt er per index de round kolom gebruikt met de XOR. Ronde 1 gebruikt R1, Ronde 2 R2 enz.

Voor Y1 in de eerste ronde wordt er R1 gebruikt, voor de makkelijke leesbaarheid wordt deze in binair opgeschreven, en eronder de binaire XOR uitkomst.

|  |
| --- |
| Y1 |
| AA |
| FC |
| A0 |
| 45 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | AA | FC | A0 | 45 |
| **Y1** | 10101010 | 11111100 | 10100000 | 01000101 |
| **R1** | 00000001 | 00000000 | 00000000 | 00000000 |
| **G** | 10101011 | 11111100 | 10100000 | 01000101 |

De uitkomst hiervan terug naar hexadecimaal is G(W3):

|  |
| --- |
| G |
| AB |
| FC |
| A0 |
| 45 |

|  |  |  |  |
| --- | --- | --- | --- |
| W0 | W1 | W2 | W3 |
| 47 | 64 | 5A | 68 |
| 52 | 7A | 50 | 62 |
| 39 | 4B | 39 | 55 |
| 36 | 59 | 33 | 47 |

Om W4, W5, W6 en W7 te bepalen, wordt het volgende gedaan:

* XOR W0 en G om W4 te bepalen;
* XOR W1 en W4 om W5 te bepalen;
* XOR W2 en W5 om W6 te bepalen;
* XOR W3 en W6 om W7 te bepalen.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| W0 | W1 | W2 | W3 | W4 | W5 | W6 | W7 |
| 47 | 64 | 5A | 68 | EC | 88 | D2 | BA |
| 52 | 7A | 50 | 62 | AE | D4 | 84 | E6 |
| 39 | 4B | 39 | 55 | 99 | D2 | EB | BE |
| 36 | 59 | 33 | 47 | 73 | 2A | 19 | 5E |

Dit wordt herhaald tot aan de V43 met iedere keer de laatste Word kolom als input van de G functie, in dit geval G(W7). W0-W43 maakt een totaal van 44 Words.

### **5.1.3 AddRoundKey**

Voordat de rondes beginnen, wordt er een XOR-operatie uitgevoerd op de eerste sub sleutel met de te versleutelen tekst.

De eerste ronde van de sleutel werd in hoofdstuk 5.1.2 berekend.

|  |  |  |  |
| --- | --- | --- | --- |
| W0 | W1 | W2 | W3 |
| 47 | 64 | 5A | 68 |
| 52 | 7A | 50 | 62 |
| 39 | 4B | 39 | 55 |
| 36 | 59 | 33 | 47 |

Het eerste tekstsegment dat versleuteld moet worden wat gedefinieerd staat in hoofdstuk 5.1.2, is: “**Dit is een gehei**”

In hexadecimaal: **44 69 74 20 69 73 20 65 65 6E 20 67 65 68 65 69**

Deze tekst kan ook weer omgezet worden naar een grit.

|  |  |  |  |
| --- | --- | --- | --- |
| M0 | M1 | M2 | M3 |
| 44 | 69 | 65 | 65 |
| 69 | 73 | 6E | 68 |
| 74 | 20 | 20 | 65 |
| 20 | 65 | 67 | 69 |

Per karakter (byte) wordt hier een XOR-operatie op uitgevoerd. Als voorbeeld **47** (**W00**) en **44** (**M00**).

|  |  |
| --- | --- |
| **W00** | 01000111 |
| **M00** | 01000100 |
| **AR00** | 00000011 |

De uitkomst als hexadecimaal is 03

Als deze XOR-operatie over iedere byte van de blokken wordt uitgevoerd ziet dat er uit als volgt, dit blok heeft als naam, de ‘State array’:

|  |  |  |  |
| --- | --- | --- | --- |
| A0 | A1 | A2 | A3 |
| 03 | 0D | 3F | 0D |
| 3B | 09 | 3E | 0A |
| 4D | 6B | 19 | 30 |
| 16 | 3C | 54 | 2E |

Met deze state array als input gaan de rondes van start, deze bestaan uit de SubBytes, ShiftRows, MixColumns en AddRoundKey stappen.

### **5.1.4 SubBytes**

In de eerste ronde de input voor deze stap is de state array uit hoofdstuk 5.1.3:

|  |  |  |  |
| --- | --- | --- | --- |
| A0 | A1 | A2 | A3 |
| 03 | 0D | 3F | 0D |
| 3B | 09 | 3E | 0A |
| 4D | 6B | 19 | 30 |
| 16 | 3C | 54 | 2E |

Bij de overige rondes zal dit de uitkomst zijn van de laatste stap uit de vorige ronde.

In deze stap wordt dezelfde S-Box tabel gebruikt eerder benoemd in hoofdstuk 5.1.2 (Afbeelding 2), elke waarde wordt daarmee vervangen. Als resultaat de volgende tabel:

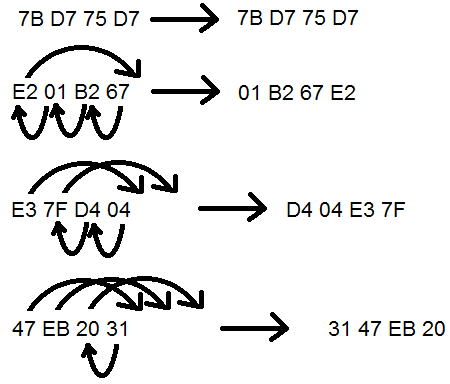
|  |  |  |  |
| --- | --- | --- | --- |
| B0 | B1 | B2 | B3 |
| 7B | D7 | 75 | D7 |
| E2 | 01 | B2 | 67 |
| E3 | 7F | D4 | 04 |
| 47 | EB | 20 | 31 |

### **5.1.5 ShiftRows**

In deze stap wordt de uitkomst van de vorige stap SubBytes gebruikt.

|  |  |  |  |
| --- | --- | --- | --- |
| B0 | B1 | B2 | B3 |
| 7B | D7 | 75 | D7 |
| E2 | 01 | B2 | 67 |
| E3 | 7F | D4 | 04 |
| 47 | EB | 20 | 31 |

In deze stap wordt per rij een circulaire left shift operatie uitgevoerd, iedere rij krijgt een byte shift erbij. De eerste rij veranderd niet, de tweede rij schuift één byte op, de derde rij twee bytes en de vierde rij drie bytes.



Afbeelding 4 ShiftRows

Het resultaat hiervan ziet er uit als volgt:

|  |  |  |  |
| --- | --- | --- | --- |
| R0 | R1 | R2 | R3 |
| 7B | D7 | 75 | D7 |
| 01 | B2 | 67 | E2 |
| D4 | 04 | E3 | 7F |
| 31 | 47 | EB | 20 |

### **5.1.6 MixColumns**

Deze stap neemt het resultaat van de ShiftRows stap als input.

|  |  |  |  |
| --- | --- | --- | --- |
| R0 | R1 | R2 | R3 |
| 7B | D7 | 75 | D7 |
| 01 | B2 | 67 | E2 |
| D4 | 04 | E3 | 7F |
| 31 | 47 | EB | 20 |

De MixColumns stap heeft voor het versleutelen en decoderen vaste tabellen waarmee de input vermenigvuldigd moet worden, de vastgestelde tabel voor het versleutelen ziet er uit als volgt:

|  |  |  |  |
| --- | --- | --- | --- |
| 02 | 03 | 01 | 01 |
| 01 | 02 | 03 | 01 |
| 01 | 01 | 02 | 03 |
| 03 | 01 | 01 | 02 |

Voor het decoderen is de vastgestelde tabel als volgt:

|  |  |  |  |
| --- | --- | --- | --- |
| 0E | 0B | 0D | 09 |
| 09 | 0E | 0B | 0D |
| 0D | 09 | 0E | 0B |
| 0B | 0D | 09 | 0E |

Voor dit voorbeeld wordt de tabel voor het versleutelen gebruikt.

Bij deze stap worden de rijen van de tabel voor het versleutelen, vermenigvuldigt met de kolommen van de input tabel en dat bij elkaar opgeteld, vervolgens een XOR-operatie over deze waardes.

De eerste rij van de versleutel tabel is:

|  |  |  |  |
| --- | --- | --- | --- |
| 02 | 03 | 01 | 01 |

De eerste kolom van de input tabel is:

|  |
| --- |
| R0 |
| 7B |
| 01 |
| D4 |
| 31 |

Het antwoord wat hier gezocht is, is (02 \* 7B) ⊕ (03 \* 01) ⊕ (01 \* D4) ⊕ (01 \* 31).

Het berekenen hiervan kan worden gedaan via polynomial multiplication MOD 2, dit wordt hieronder uitgelegd.

Hiervoor worden de volgende twee waardes gebruikt; De eerste byte van de eerste rij van de versleutel tabel (02) en de eerste byte van de eerste kolom van de input tabel (7B).

Deze waardes worden omgezet naar binair (7B, 01111011) en (02, 00000010), dan staan de 1’en of 0’en op een locatie in de binaire rij, deze locaties zijn als volgt:

01111011

x6 + x5 + x4 + x3 + x1 + x0

00000010

x1

Het optellen van deze twee waardes wordt in dit voorbeeld via een grit, de hierboven beschreven nummers staan nu in deze grit hieronder:

|  |  |
| --- | --- |
|  | 1 |
| 0 |  |
| 1 |  |
| 3 |  |
| 4  5  6 |  |

Vervolgens worden de waardes simpelweg met elkaar opgeteld:

|  |  |
| --- | --- |
|  | 1 |
| 0 | 1 |
| 1 | 2 |
| 3 | 4 |
| 4  5  6 | 5  6  7 |

Alle getallen die een even aantal voorkomen worden weggelaten, in dit geval zijn dit er geen.

|  |  |
| --- | --- |
|  | 1 |
| 0 | 1 |
| 1 | 2 |
| 3 | 4 |
| 4  5  6 | 5  6  7 |

Deze getallen worden terug geschreven naar binair op basis van de locatie.

x7 x6 x5 x4 x2 x1

11110110

In hexadecimaal: F6

Als de uitkomst hiervan onder een vast gedefinieerde waarde “100011011” is, dan is dit het antwoord, als dit antwoord groter is, dan moet er een XOR-operatie over gedaan worden met deze vaste waarde tot dat deze kleiner is.

Dit was enkel de 02 \* 7B van de (02 \* 7B) ⊕ (03 \* 01) ⊕ (01 \* D4) ⊕ (01 \* 31). De overige drie worden op dezelfde manier gedaan, die staan hieronder in verkorte versie.

0x03 \* 0x01

(00000011 \* 00000001)

|  |  |  |
| --- | --- | --- |
|  | 0 | 1 |
| 0 | 0 | 1 |

**00000010 (0x02)**

0x01 \* 0xD4

(00000001 \* 11010100)

|  |  |
| --- | --- |
|  | 0 |
| 2 | 2 |
| 4 | 4 |
| 6 | 6 |
| 7 | 7 |

**11010100 (0xD4)**

0x01 \* 0x31

(00000001 \* 00110001)

|  |  |
| --- | --- |
|  | 0 |
| 0 | 0 |
| 4 | 4 |
| 5 | 5 |

**00110001 (0x31)**

Over deze vier waardes samen moet een XOR-operatie uitgevoerd worden.

|  |  |
| --- | --- |
| **0xF6** | 11110110 |
| **0x02**  **0xD4**  **0x31** | 00000010  11010100  00110001 |
| **0x11** | 00010001 |

Het resultaat is 0x11, dit is het antwoord op het eerste vakje, al deze stappen moeten per rij gedaan worden op de kolommen.

|  |  |  |  |
| --- | --- | --- | --- |
| 11 | ? | ? | ? |
| ? | ? | ? | ? |
| ? | ? | ? | ? |
| ? | ? | ? | ? |

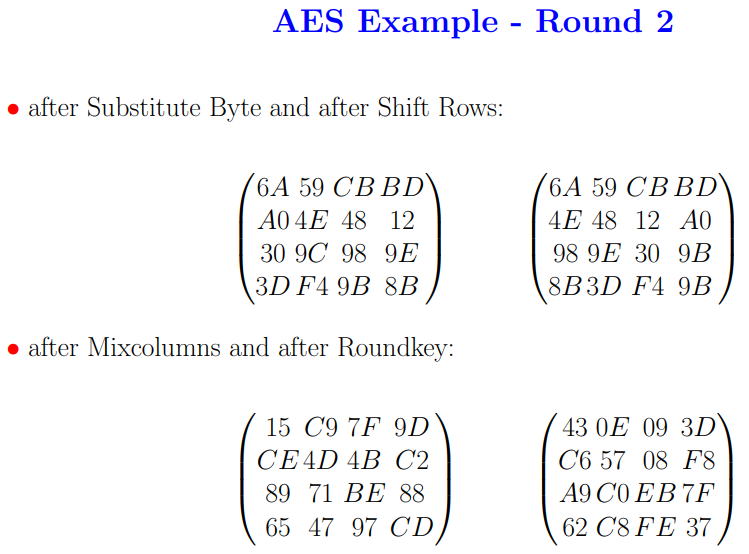
Al deze stappen worden herhaald per ronde, per te versleutelen tekstsegment. Dit voorbeeld ging over de werking per stap, het is niet relevant en erg veel werk om al de stappen door te gaan tot het uiteindelijke resultaat, dat wordt daarom ook niet uitgevoerd.

## **5.2 Welke voorbeelden voor het handmatig toepassen bestaan er al?**

In deze deelvraag is onderzoek gedaan naar al bestaande voorbeelden voor het handmatig toepassen van het AES-algoritme.

### **5.2.1 Kavaliro**

(Kivaliro, 2014) heeft een voorbeeld waar er eerst uitleg gegeven wordt over het AES-algoritme zelf, de uitleg is vrij abstract waardoor veel diepgaande details weggelaten zijn. Hierna wordt dit ook daadwerkelijk uitgevoerd met de uitkomst per ronde.



Afbeelding 5 Kavaliro AES-example

Zoals in de bovenstaande afbeelding te zien is, wordt hier geen verdere uitleg gegeven, de antwoorden staan wel per ronde opgeschreven zodat dit gemakkelijk nagedaan worden.

**Voordelen**

* Er is per ronde een antwoord zichtbaar, dit maakt het gemakkelijk te controleren of het gevonden antwoord klopt.

**Nadelen**

* Er is geen gedetailleerde uitleg over het algoritme zelf, waardoor dit van een externe bron gehaald moet worden.

5.2.2 Satish C J

## **5.3 Wat voor methodes kunnen gebruikt worden om het toe te passen?**

De MixColumns stap is voor vele een lastige stap en er zijn verschillende manieren om dit te berekenen. Er wordt in dit voorbeeld één manier gebruikt.

# **6 Conclusie**

# **7 Aanbeveling**

# **8 Planning**

# **Literatuurlijst**

Kavaliro. (2014, maart). *'* *AES Example - Input (128 bit key and message)'.* Geraadpleegd op 7 januari 2021 van <https://kavaliro.com/wp-content/uploads/2014/03/AES.pdf>

# **Afkortingen en woordenlijst**

**AES** Advanced Encryption Standard

**MOD** Modulus

**XOR** Exclusieve disjunctie

**⊕** XOR-teken

**Bit** Waarde uit het binaire stelsel, een 1 of 0

# **Bijlagen**