Whitebox Cryptography

By Sanyog Chhetri

Content

- What is White-box and what’s it used for?
- Intro to AES algorithm
- Side-channel attacks
- Differential fault injection attack
- DFA on White-box AES
- Great Reads

What is White-box and what's it used for?

- Whitebox cryptography is a software-based method to protect cryptographic keys and algorithm from being exposed or tampered with in an untrusted environment. This is usually done by mixing key addition with S-boxes. It then further uses techniques such as:

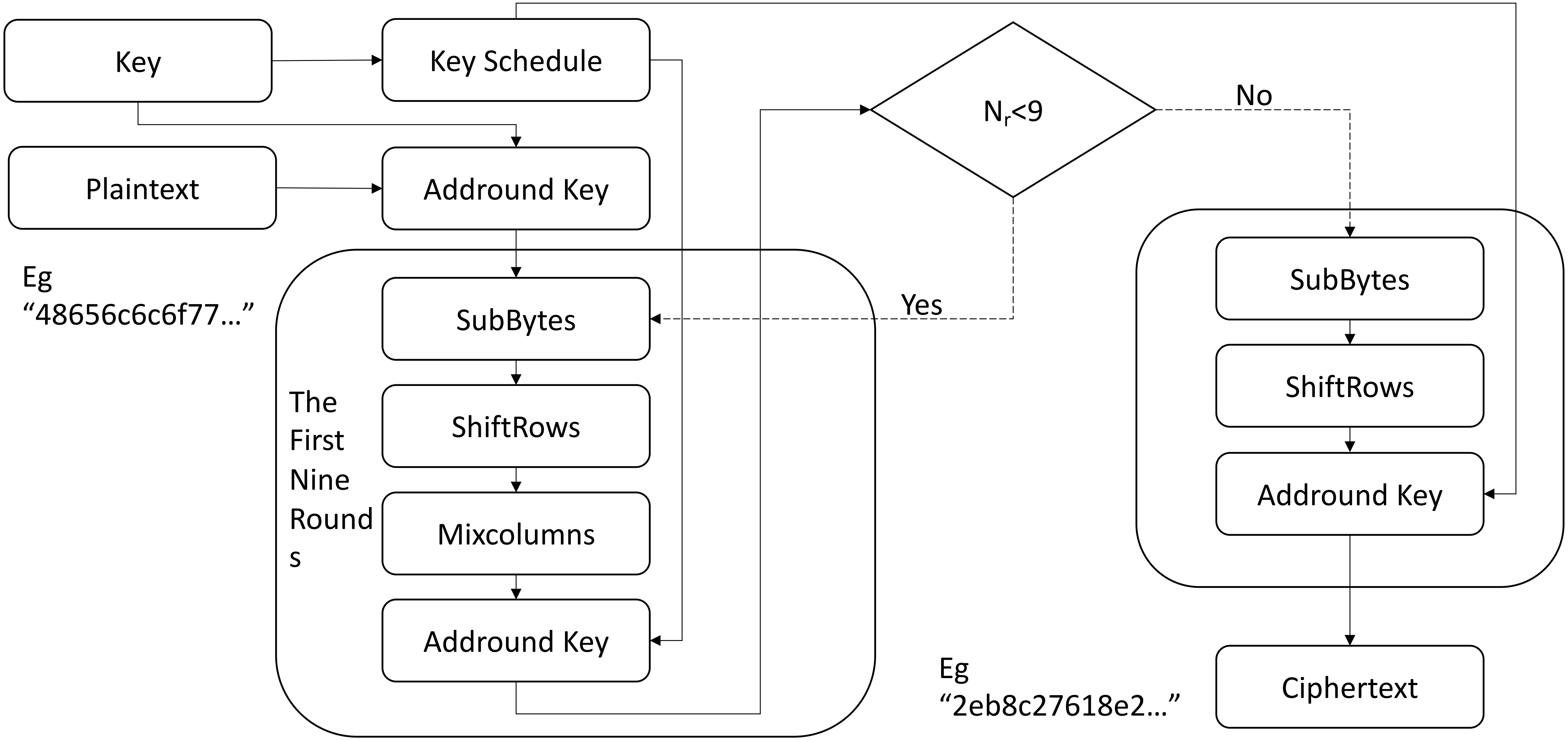
Obfuscation

encryption

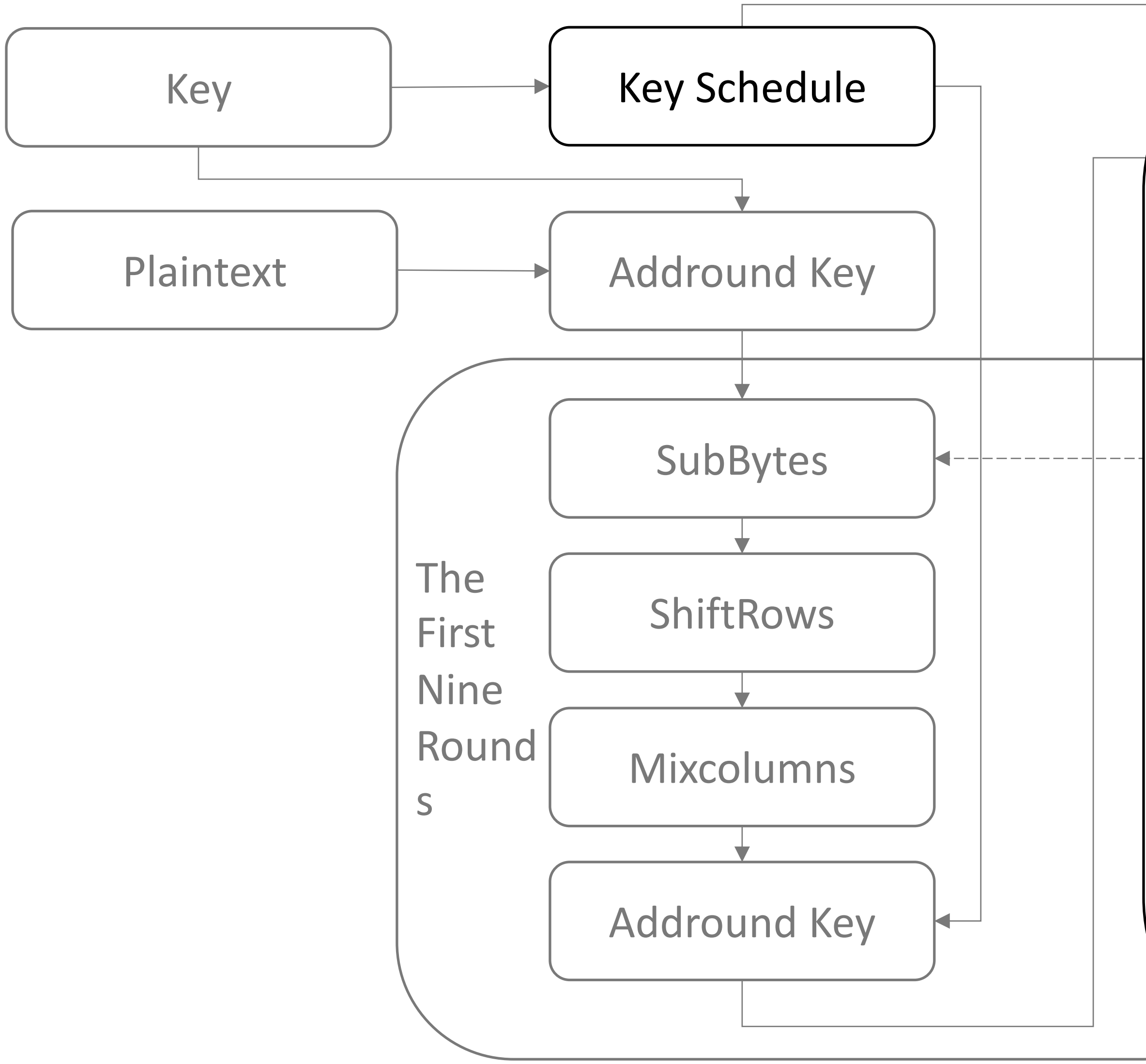
Mathematical Transformation

- White-box cryptography is useful for securing open devices, such as smartphones, that are vulnerable to analysis or rooting.

Intro to AES algorithm



Intro to AES algorithm



Key

Key Schedule

Plaintext

Addround Key

SubBytes

ShiftRows

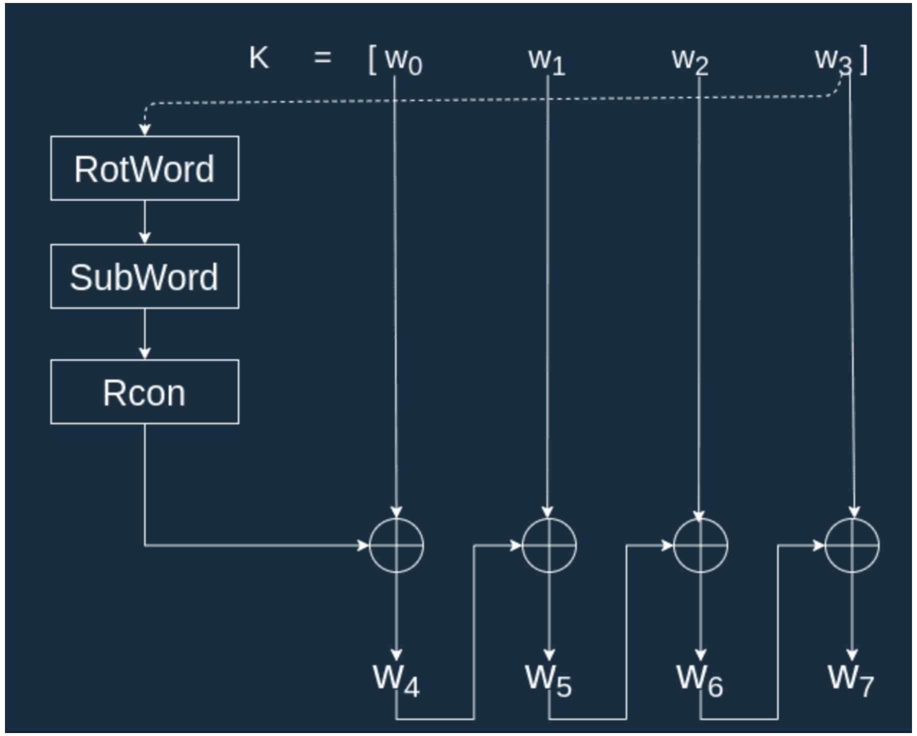
Mixcolumns

Addround Key

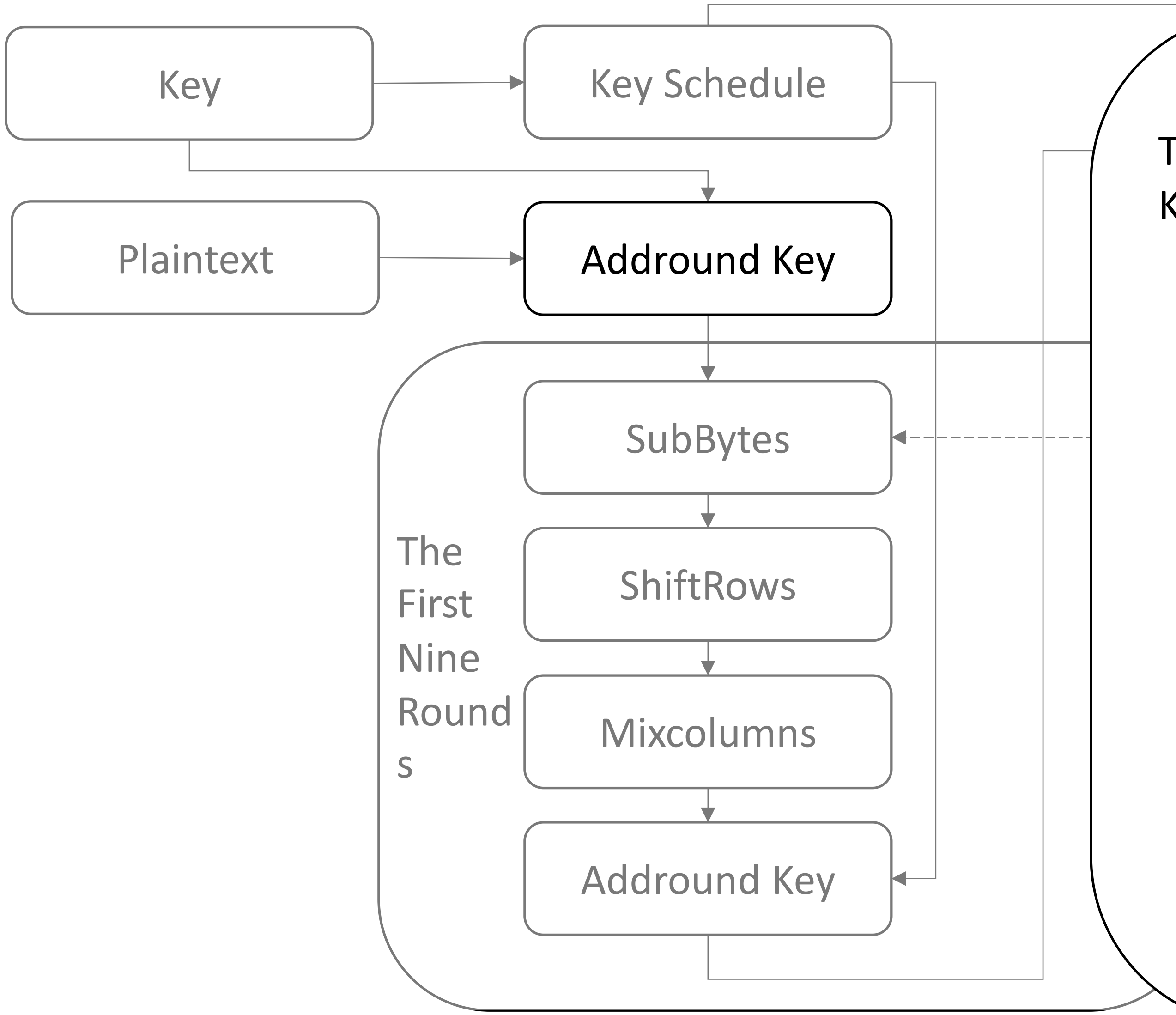
The First Nine Rounds

Key Schedule

New keys are derived for each round from the original. The key is separated into 4 word bytes.

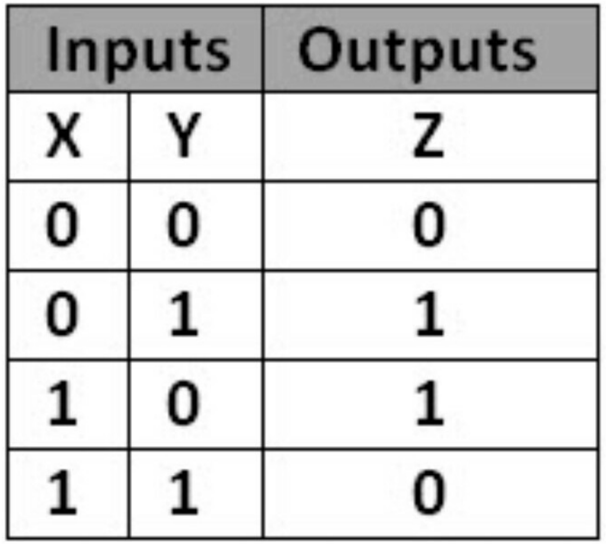


Intro to AES algorithm



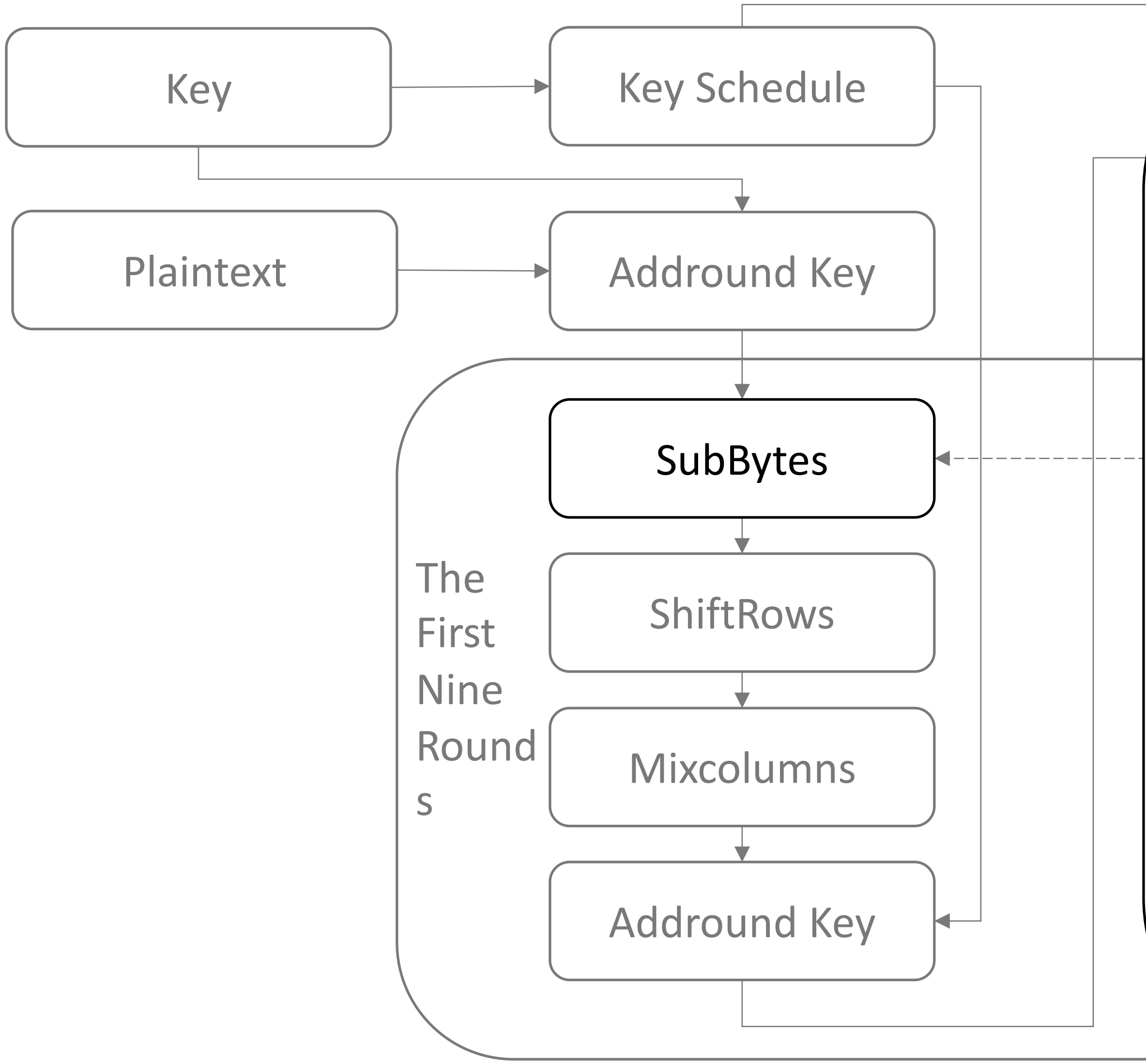
Addround Key

This is just XOR’ing the Plaintext or input with the Key.



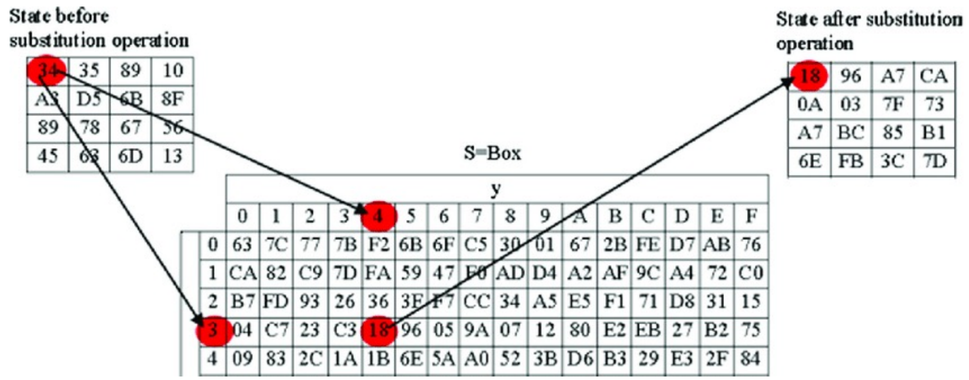
Inputs		Outputs
X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0

Intro to AES algorithm



SubBytes

In this each byte is substituted by another byte. It’s performed using a lookup table called the S-box. This substitution is done in a way that a byte is never substituted by itself and also not substituted by another byte which is a compliment of the current byte.



State before substitution operation

34	35	89	10
A3	D5	6B	8F
89	78	67	56
45	63	6D	13

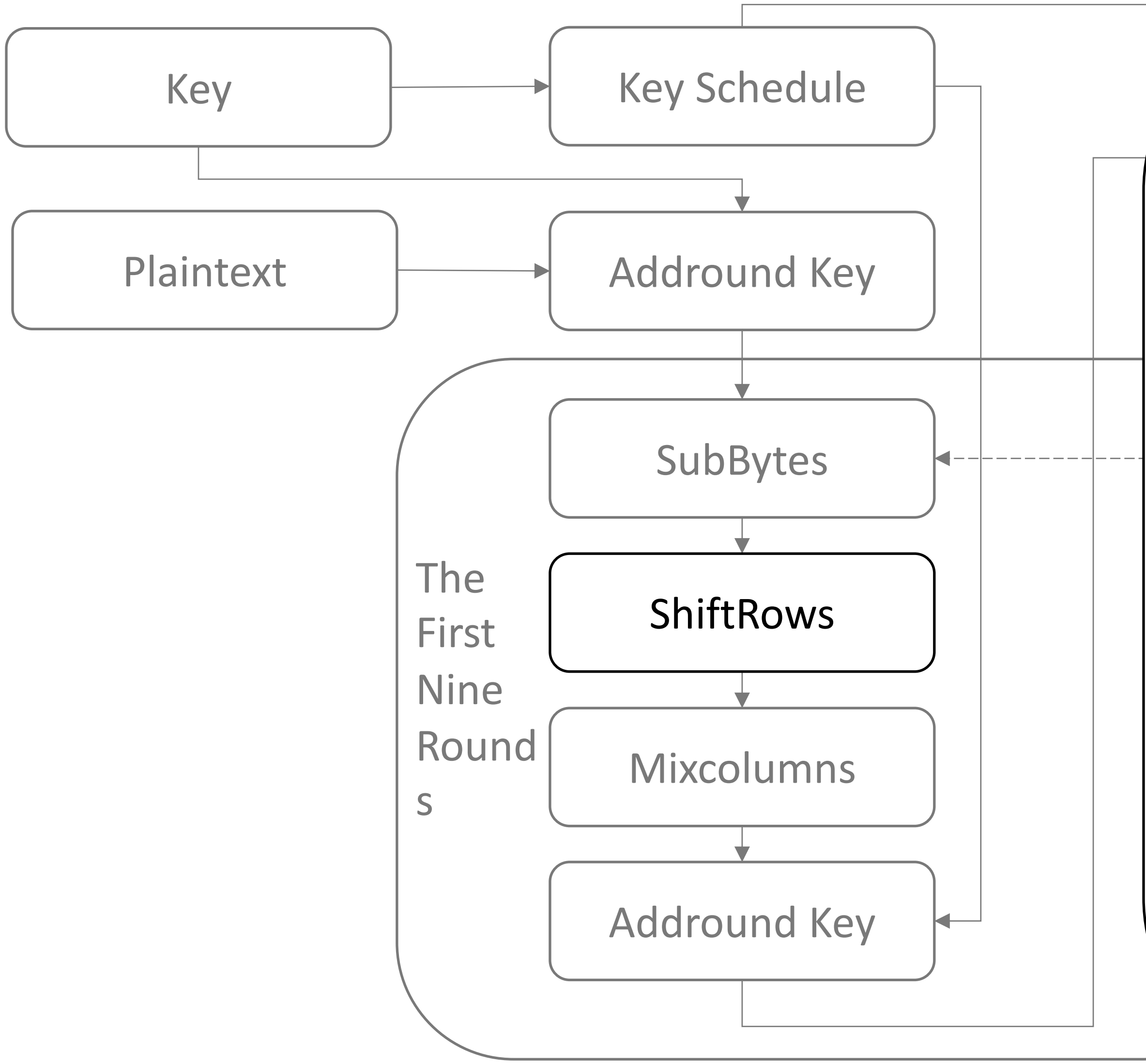
S=Box

	y															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
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

State after substitution operation

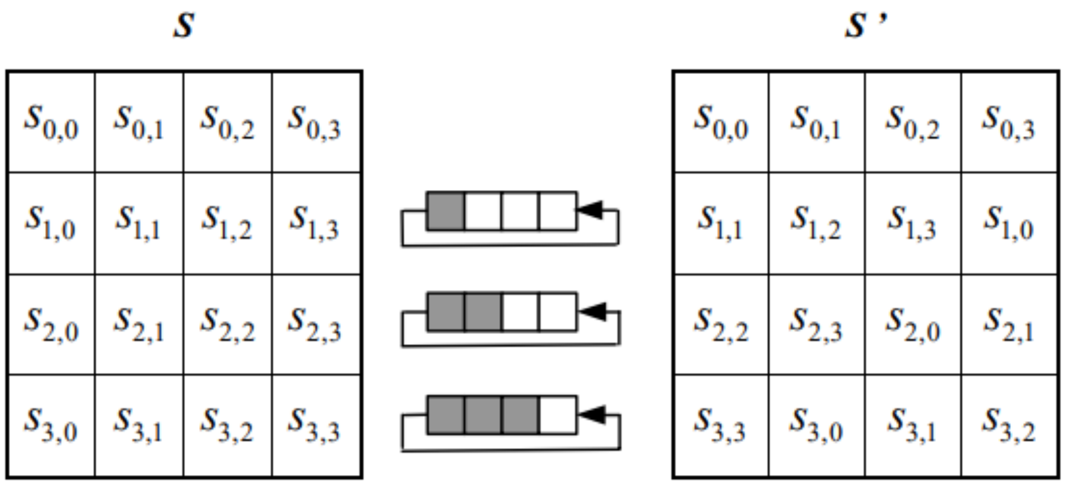
18	96	A7	CA
0A	03	7F	73
A7	BC	85	B1
6E	FB	3C	7D

Intro to AES algorithm



ShiftRows

It is just as it sounds, Each row is shifted a particular number of times.



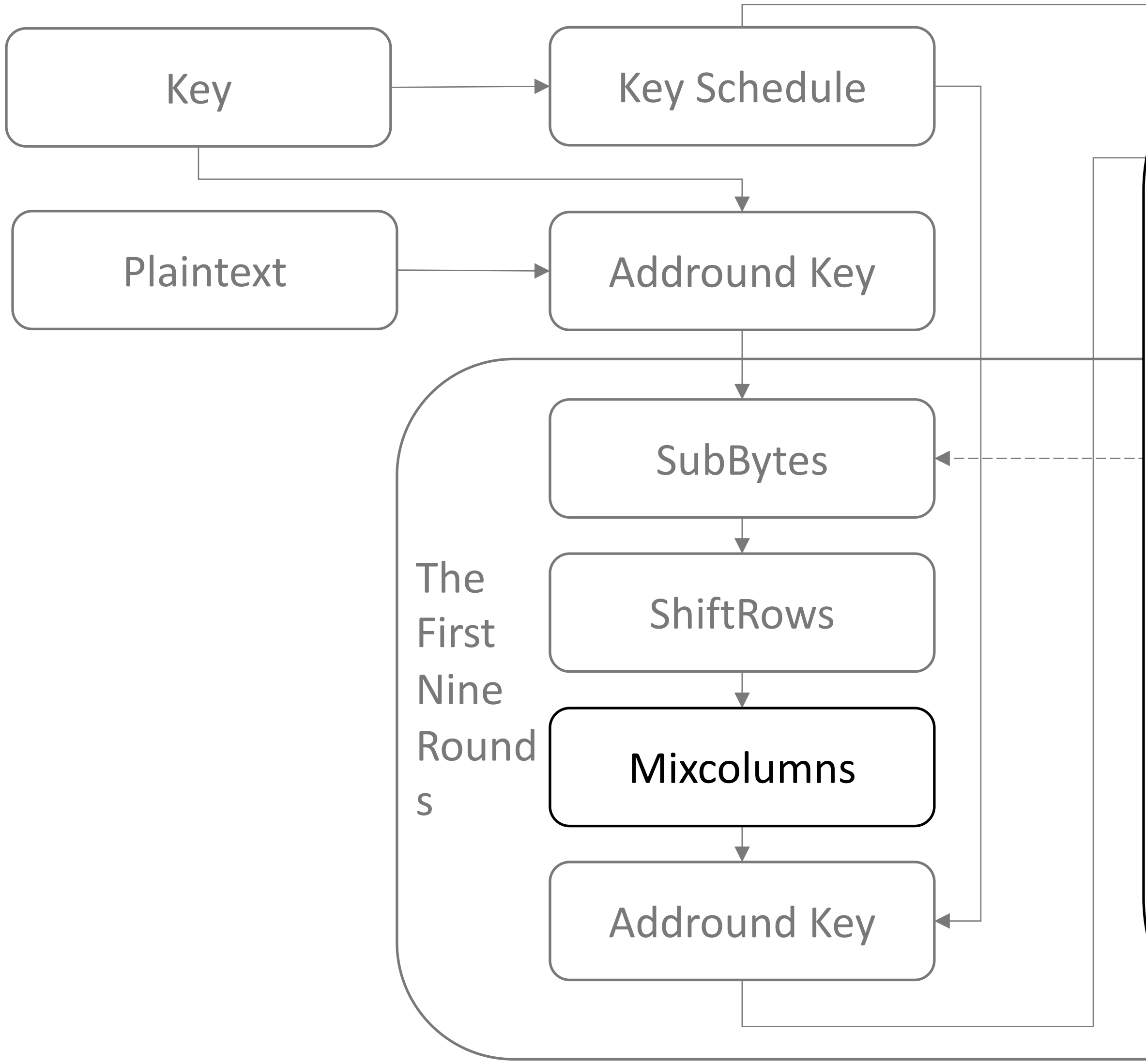
s

$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,0}$	$s_{1,1}$	$s_{1,2}$	$s_{1,3}$
$s_{2,0}$	$s_{2,1}$	$s_{2,2}$	$s_{2,3}$
$s_{3,0}$	$s_{3,1}$	$s_{3,2}$	$s_{3,3}$

s'

$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,1}$	$s_{1,2}$	$s_{1,3}$	$s_{1,0}$
$s_{2,2}$	$s_{2,3}$	$s_{2,0}$	$s_{2,1}$
$s_{3,3}$	$s_{3,0}$	$s_{3,1}$	$s_{3,2}$

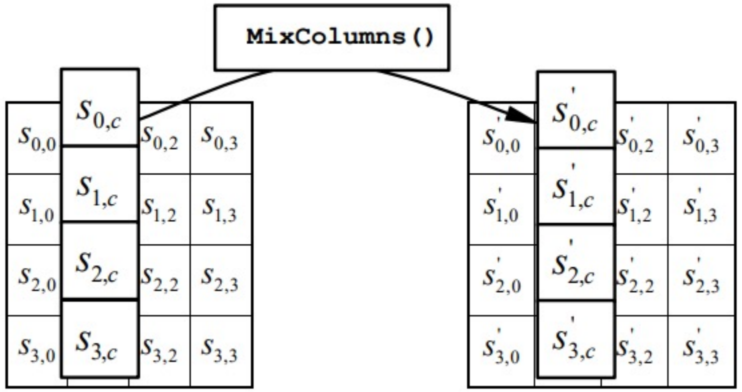
Intro to AES algorithm



Mixcolumns

This step is basically a matrix multiplication. Each column is multiplied with a specific matrix and thus the position of each byte in the column is changed as a result.

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$



Side-channel attacks

DCA

Differential Computational Analysis:

- It is software derived version of the differential Power Analysis (DPA) attack.
- The statistical analysis of the data managed in memory or registers during obtained when executing a cryptographic primitive with different inputs might correlate to and reveal information about the secret key material used by the algorithm.

DFA

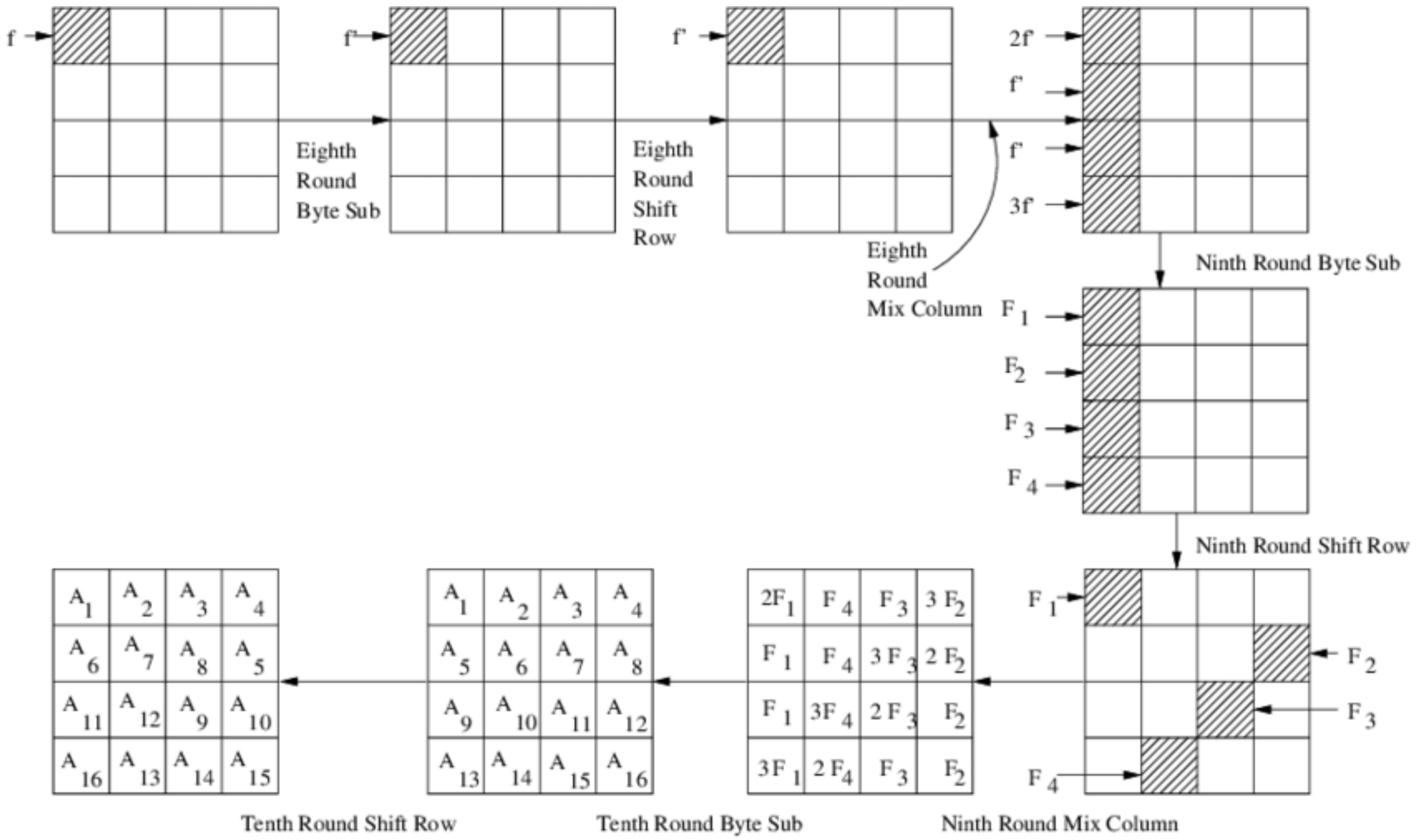
Differential Fault Analysis:

- It comes from a hardware background, and it is based in the induction of faults during the execution of a cryptographic algorithms.
- The statistical analysis of an original trace together with traces obtained using the same input and injecting faults during its execution can give the secret key of the software White-Box Cryptography implementation.

Differential fault injection attack

- Where to inject the faults?
  - Finding out where to inject faults requires understanding of the code/binary and understanding of how Cryptographic algorithm works.
  - This can however also be done via automation and there are some nice tools for it. This is one of the tools used for the purpose of attempting to automate the DFA attacks.
    - https://github.com/SideChannelMarvels/JeanGrey

Differential fault injection attack



Ninth Round Mix Column

$2F_1$	F_4	F_3	$3F_2$
F_1	F_4	$3F_3$	$2F_2$
F_1	$3F_4$	$2F_3$	F_2
$3F_1$	$2F_4$	F_3	F_2

Tenth Round Byte Sub

A_1	A_2	A_3	A_4
A_5	A_6	A_7	A_8
A_9	A_{10}	A_{11}	A_{12}
A_{13}	A_{14}	A_{15}	A_{16}

Tenth Round Shift Row

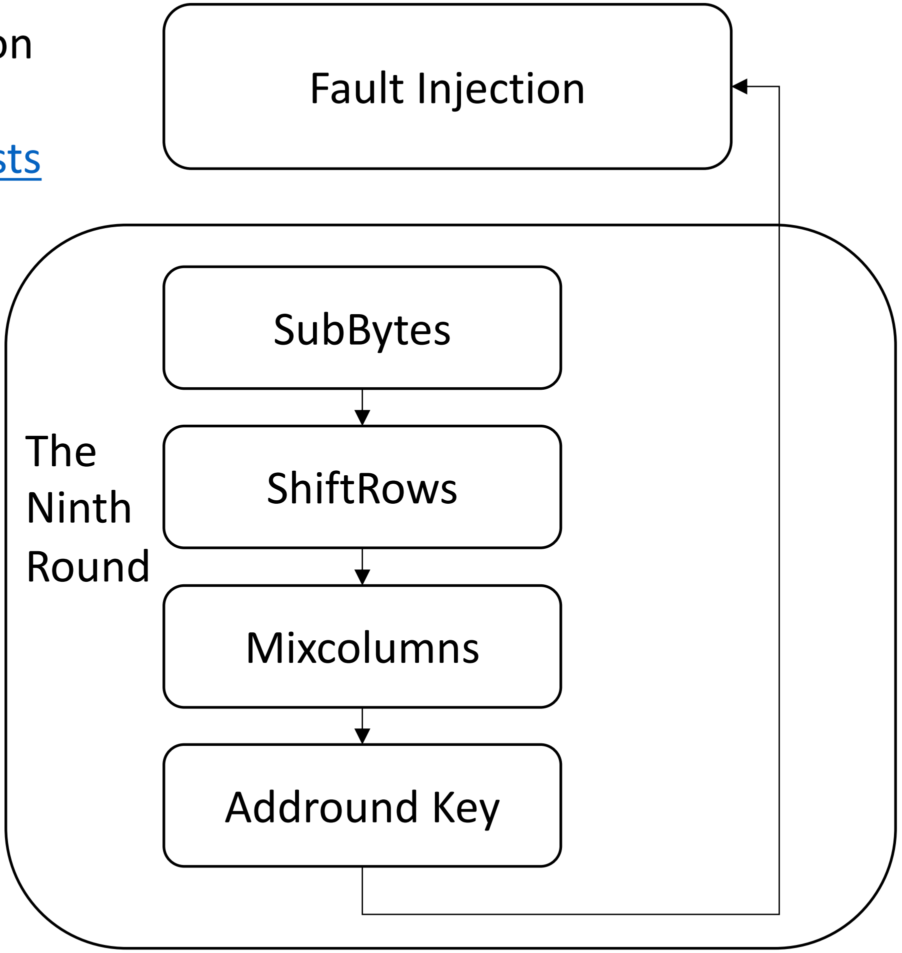
A_1	A_2	A_3	A_4
A_6	A_7	A_8	A_5
A_{11}	A_{12}	A_9	A_{10}
A_{16}	A_{13}	A_{14}	A_{15}

Tools for retrieving the last round key:

- https://github.com/SideChannelMarvels/JeanGrey/tree/master/phoenixAES

DFA on White-box AES

Whitebox implementation link: https://whibox.io/contests/2019/cadidate/26.c



Plaintext:
48656c6c6f776f726c646f
6f6f6f6f6f

Ciphertext:

2e	B8	C2	76
18	E2	97	48
94	Cb	65	7d
B7	57	B6	c4

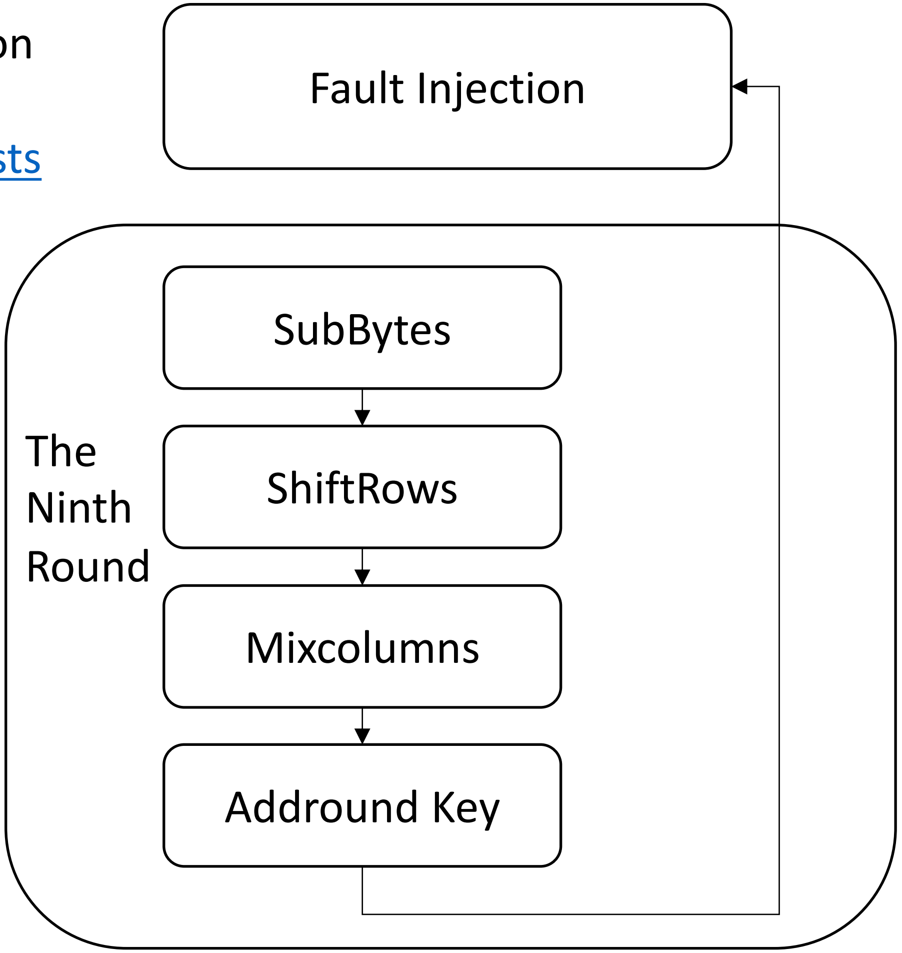
Fault Injected Ciphertext:

3a	B8	C2	76
18	E2	97	Fa
94	Cb	B6	7d
B7	bd	B6	c4

DFA on White-box AES

Whitebox implementation link: https://whibox.io/contests/2019/cadidate/26.c

This fault should be 2 per column, if the injection occurs in the 8th round this only needs to be done for one column, if the injection is in the 9th round each column needs to be addressed.



Fault Injection

The Ninth Round

SubBytes

ShiftRows

Mixcolumns

Addround Key

Fault Injection

Following is the state of the input before the injection and after the injection:

9b	51	6e	52
Aa	D0	85	93
C0	Df	56	33
33	46	79	b2

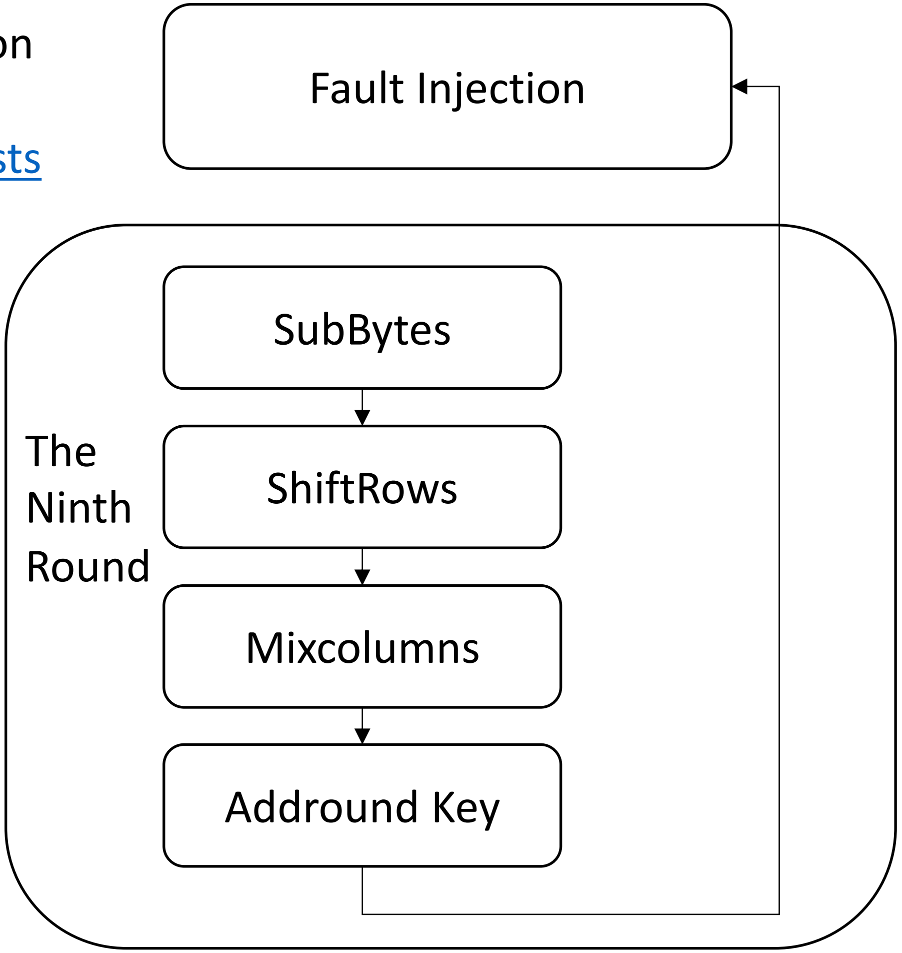
After fault injection

00	51	6e	52
Aa	D0	85	93
C0	Df	56	33
33	46	79	b2

DFA on White-box AES

Whitebox implementation link: https://whibox.io/contests/2019/cadidate/26.c

This fault should be 2 per column, if the injection occurs in the 8th round this only needs to be done for one column, if the injection is in the 9th round each column needs to be addressed.



Key scheduling

After the 10th round key is achieved, following tool can be used to get a hold of the original key:

- https://github.com/SideChannelMarvels/Stark

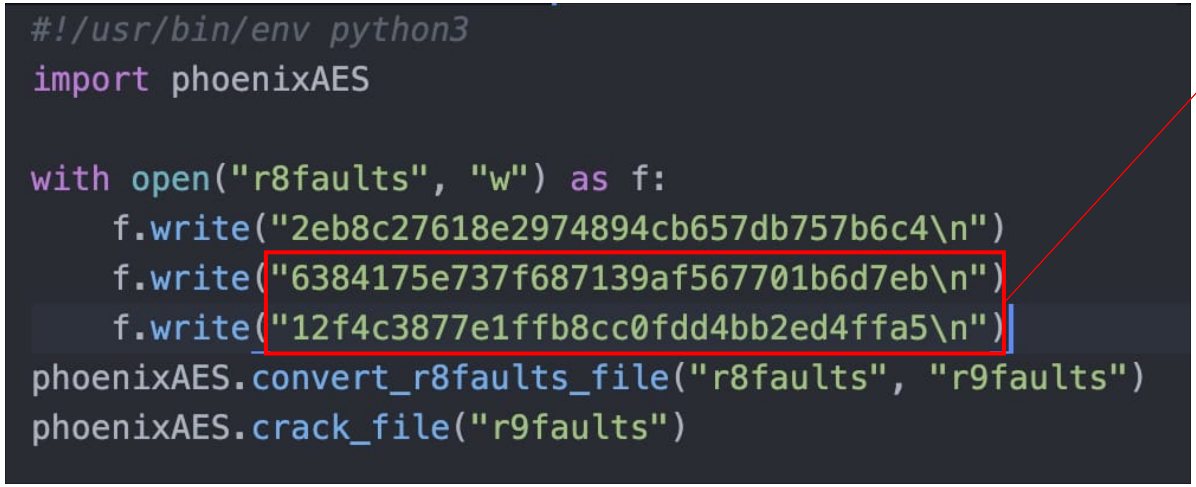
This is because the key scheduling algorithm can be inversed to get the original key.

Fault Injected Ciphertext:

3a	B8	C2	76
18	E2	97	Fa
94	Cb	B6	7d
B7	bd	B6	c4

DFA on White-box AES

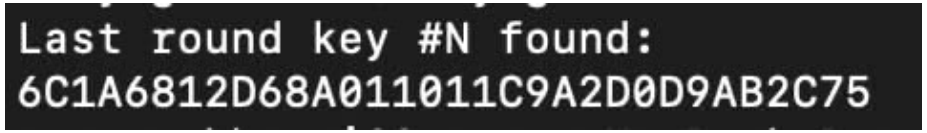
Following is the a demo on getting the 10th round key:



```
#!/usr/bin/env python3
import phoenixAES

with open("r8faults", "w") as f:
    f.write("2eb8c27618e2974894cb657db757b6c4\n")
    f.write("6384175e737f687139af567701b6d7eb\n")
    f.write("12f4c3877e1ffb8cc0fdd4bb2ed4ffa5\n")
phoenixAES.convert_r8faults_file("r8faults", "r9faults")
phoenixAES.crack_file("r9faults")
```

Faulty ciphertext after injecting on 8th round.

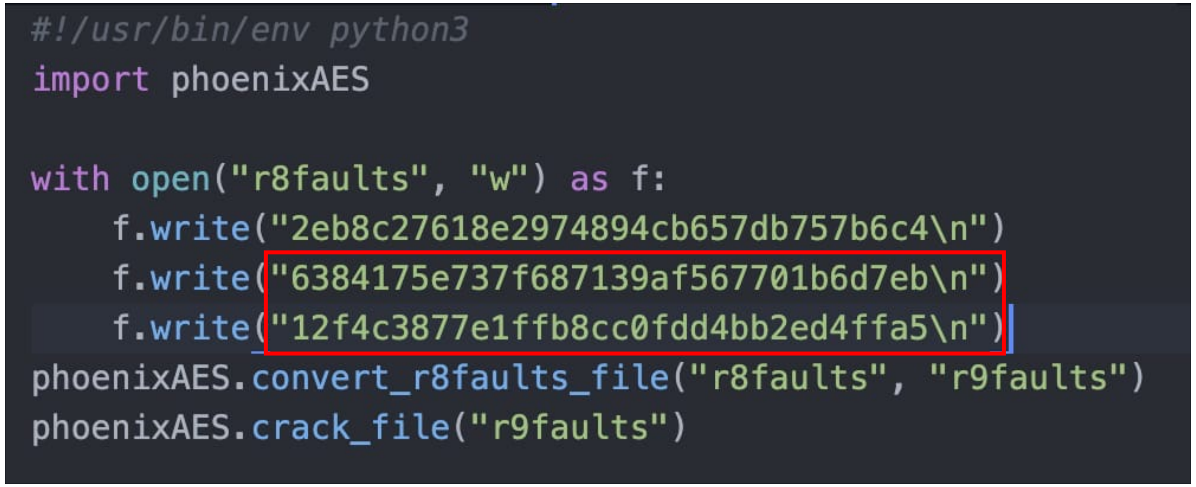
Using the tool PhoenixAES in python script, I was able to get ahold of the 10th key.



```
Last round key #N found:
6C1A6812D68A011011C9A2D0D9AB2C75
```

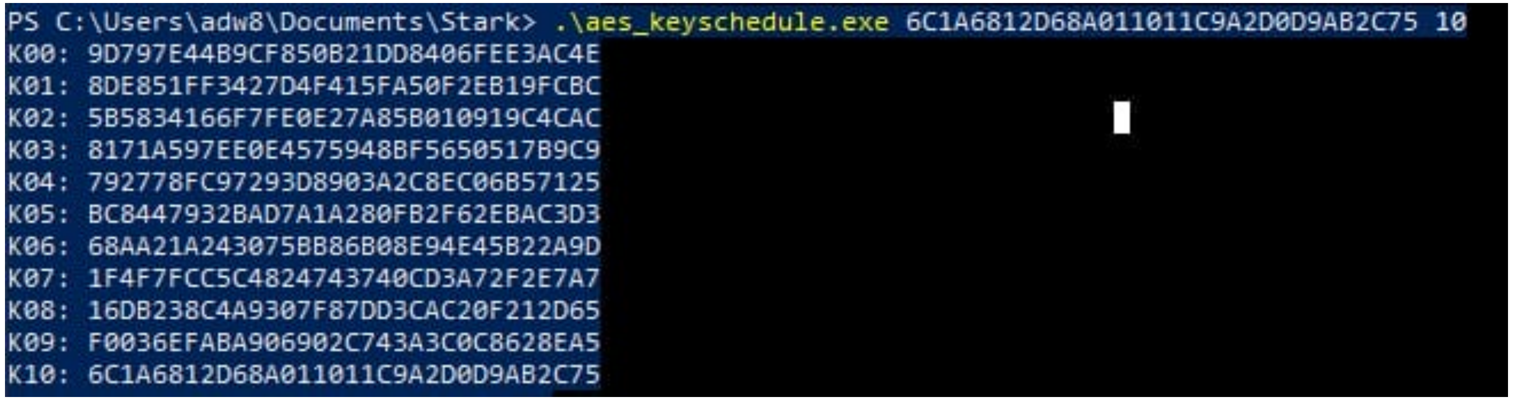
DFA on White-box AES

Following is the a demo on getting the 10th round key:



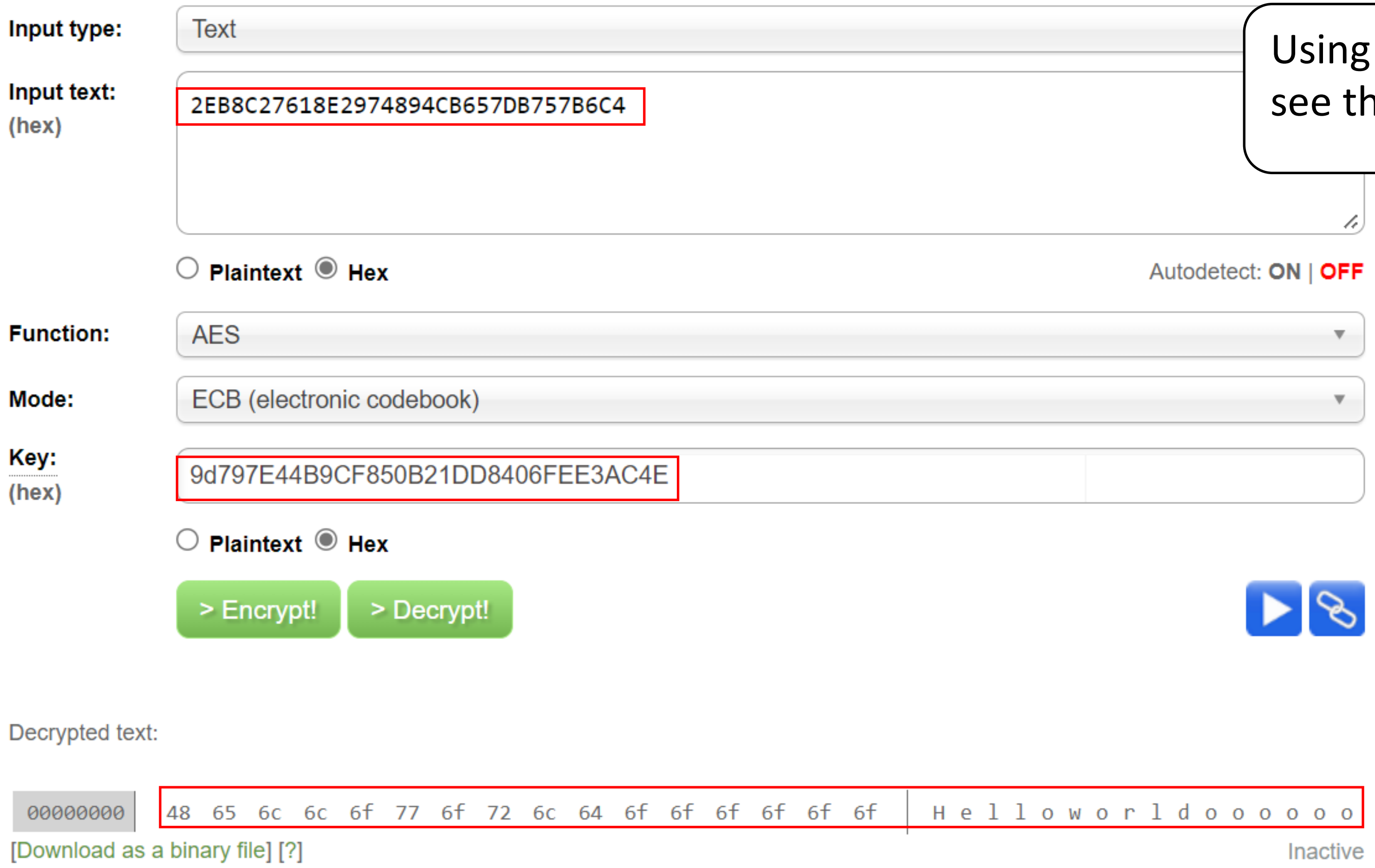
```
#!/usr/bin/env python3
import phoenixAES

with open("r8faults", "w") as f:
    f.write("2eb8c27618e2974894cb657db757b6c4\n")
    f.write("6384175e737f687139af567701b6d7eb\n")
    f.write("12f4c3877e1ffb8cc0fdd4bb2ed4ffa5\n")
phoenixAES.convert_r8faults_file("r8faults", "r9faults")
phoenixAES.crack_file("r9faults")
```

Using the tool stark(https://github.com/SideChannelMarvels/Stark) on the 10th key, I was able to obtain the original key.



```
PS C:\Users\adw8\Documents\Stark> .\aes_keyschedule.exe 6C1A6812D68A011011C9A2D0D9AB2C75 10
K00: 9D797E44B9CF850B21DD8406FEE3AC4E
K01: 8DE851FF3427D4F415FA50F2EB19FCBC
K02: 5B5834166F7FE0E27A85B010919C4CAC
K03: 8171A597EE0E4575948BF5650517B9C9
K04: 792778FC97293D8903A2C8EC06B57125
K05: BC8447932BAD7A1A280FB2F62EBAC3D3
K06: 68AA21A243075BB86B08E94E45B22A9D
K07: 1F4F7FCC5C4824743740CD3A72F2E7A7
K08: 16DB238C4A9307F87DD3CAC20F212D65
K09: F0036EFABA906902C743A3C0C8628EA5
K10: 6C1A6812D68A011011C9A2D0D9AB2C75
```

DFA on White-box AES



Input type: Text

Input text: (hex)

2EB8C27618E2974894CB657DB757B6C4

○ **Plaintext** ● **Hex**

Autodetect: **ON** | **OFF**

Function: AES

Mode: ECB (electronic codebook)

Key: (hex)

9d797E44B9CF850B21DD8406FEE3AC4E

○ **Plaintext** ● **Hex**

Decrypted text:

```
00000000  48 65 6c 6c 6f 77 6f 72 6c 64 6f 6f 6f 6f 6f 6f  H e l l o w o r l d o o o o o o
```

[Download as a binary file] [?]

Inactive

Using a aes decryption tool online we can see that, the keys found is valid.

Site used:
[AES Encryption – Easily encrypt or decrypt strings or files (online-domain-tools.com)]

Great Reads

- https://blog.quarkslab.com/differential-fault-analysis-on-white-box-aes-implementations.html
- https://eprint.iacr.org/2015/753
- https://www.geeksforgeeks.org/advanced-encryption-standard-aes/
- https://braincoke.fr/blog/2020/08/the-aes-key-schedule-explained/#rotword
- An introduction to white-box cryptography - Security Boulevard