S4M Cipher - Matthew Fisher, Steven Jakubin, Sam Jarrett, Scott Luntz, Seamus Drury

The S4M Cipher is an experimental cipher to prove that creating your own cipher will produce many vulnerabilities.

Code can be found at: https://github.com/viiwee/S4M

Matrix

AA = Text
00 = Possible Padding
SS = Salt

```
[
[AA, AA, AA, AA, AA, AA, AA, AA],
[AA, AA, AA, AA, AA, AA, AA, AA],
[AA, AA, AA, AA, AA, AA, AA, AA],
[AA, AA, AA, AA, AA, AA, AA, AA],
[AA, AA, AA, AA, AA, AA, AA, AA],
[AA, AA, AA, AA, AA, AA, AA, AA],
[AA, AA, AA, AA, AA, AA, AA, AA],
[AA, 00, 00, 00, SS, SS, SS, SS]
]
```

Inputs

Input string: Text of any size

Input key: Text of any size

Constants

Block size: 64 Bytes

Salt: 4 Bytes

Repetitions: 21

Scrambling Functions:

SwitchColumn
SwitchRow
SwitchBlock
Bitwise XOR

Output

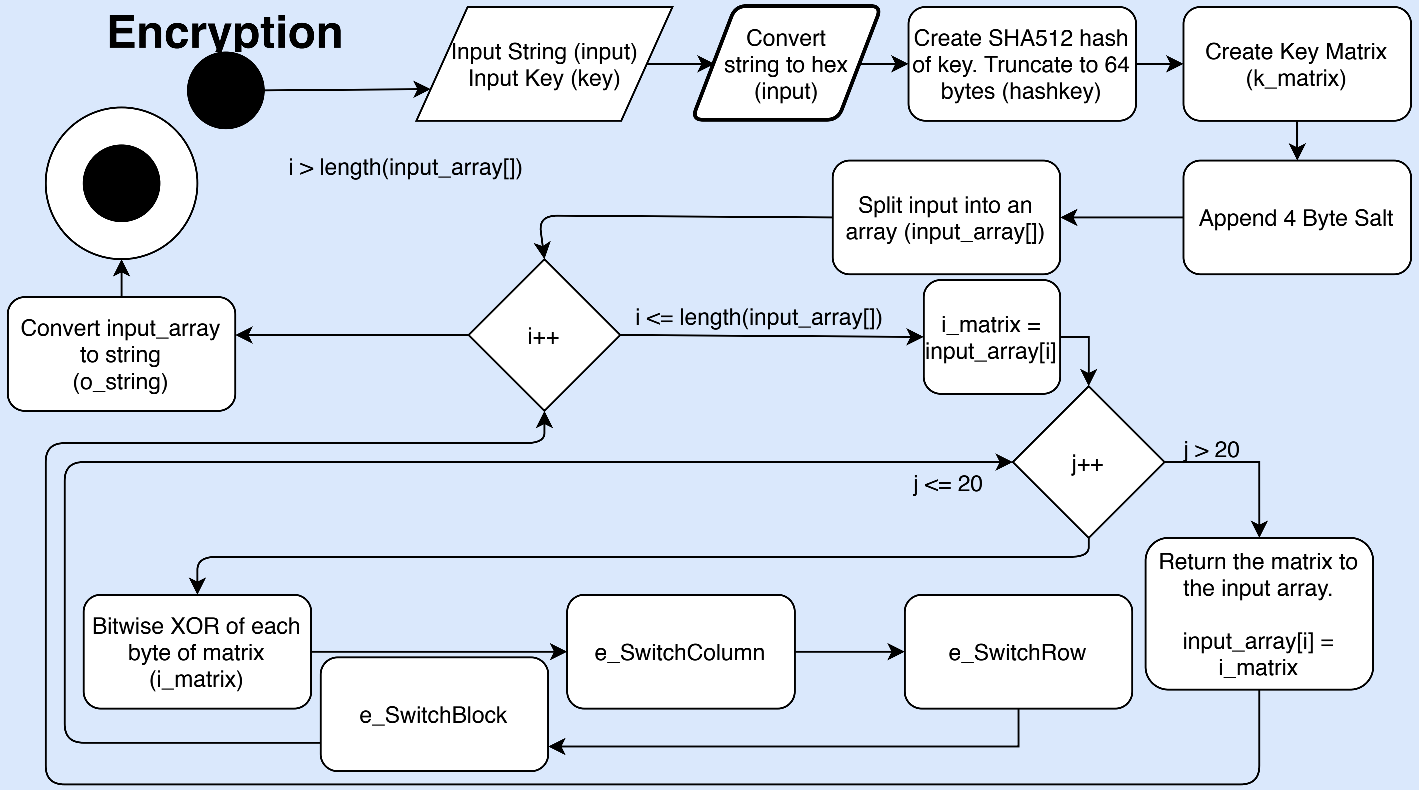
encrypt_matrix: Ciphertext string

decrypt_matrix: plaintext string

Usage

```
S4M.py [-h] [-e ENCRYPT | -d DECRYPT] [-v] key
```

Encryption



Input String (input)
Input Key (key)

Convert string to hex (input)

Create SHA512 hash of key. Truncate to 64 bytes (hashkey)

Create Key Matrix (k_matrix)

Append 4 Byte Salt

Split input into an array (input_array[])

i++

i <= length(input_array[])

i > length(input_array[])

i_matrix = input_array[i]

j++

j <= 20

j > 20

Bitwise XOR of each byte of matrix (i_matrix)

e_SwitchColumn

e_SwitchRow

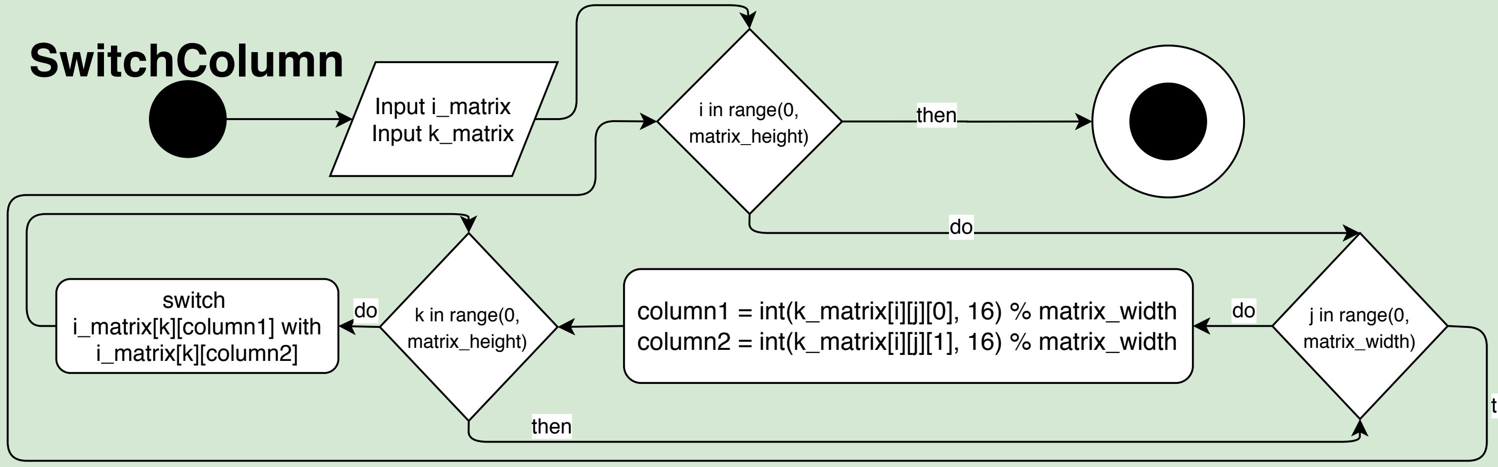
e_SwitchBlock

Return the matrix to the input array.

input_array[i] = i_matrix

Convert input_array to string (o_string)

SwitchColumn



Input i_matrix
Input k_matrix

i in range(0, matrix_height)

then

do

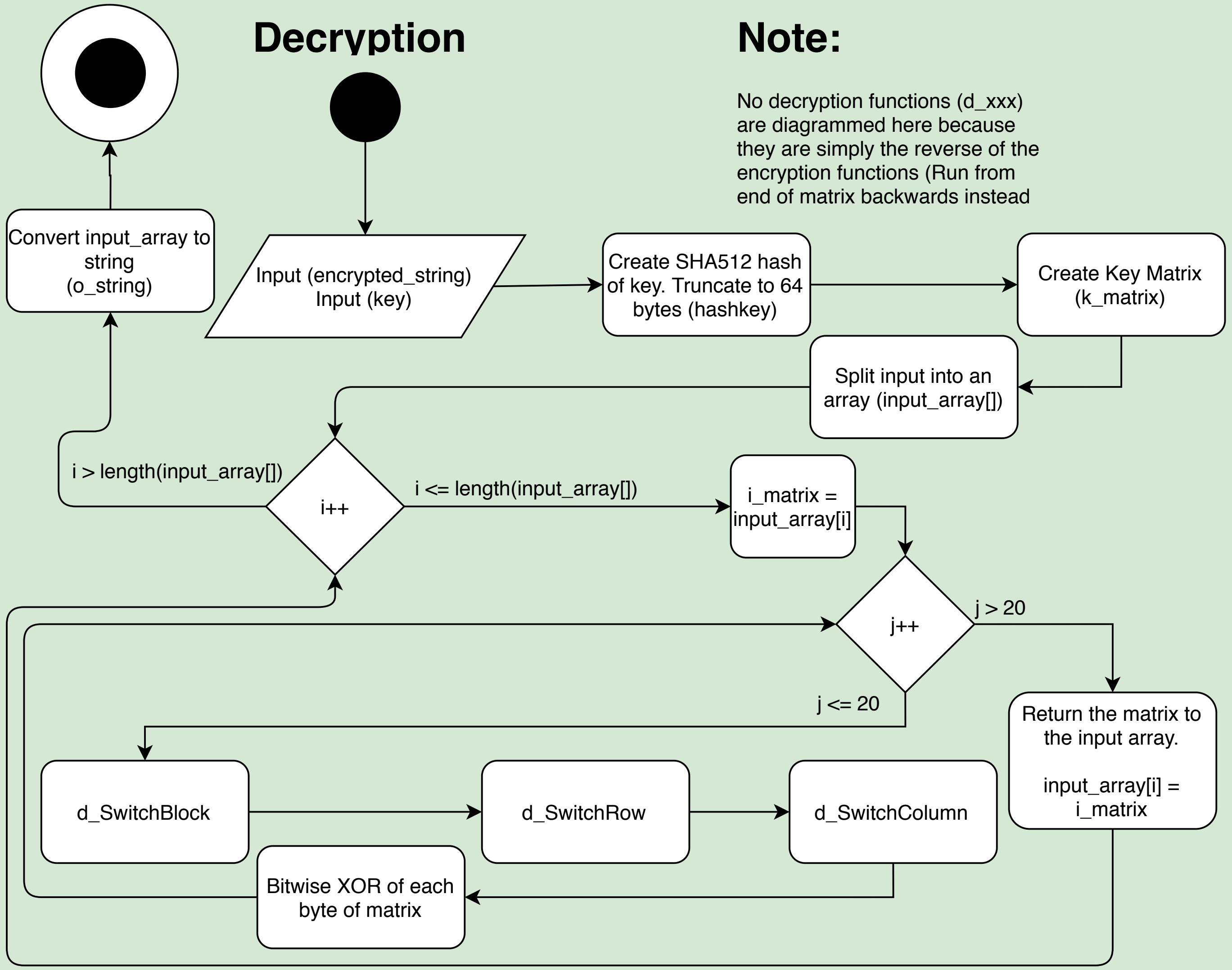
j in range(0, matrix_width)

column1 = int(k_matrix[i][j][0], 16) % matrix_width
column2 = int(k_matrix[i][j][1], 16) % matrix_width

k in range(0, matrix_height)

switch i_matrix[k][column1] with i_matrix[k][column2]

Decryption



Input (encrypted_string)
Input (key)

Create SHA512 hash of key. Truncate to 64 bytes (hashkey)

Create Key Matrix (k_matrix)

Split input into an array (input_array[])

i++

i <= length(input_array[])

i > length(input_array[])

i_matrix = input_array[i]

j++

j <= 20

j > 20

d_SwitchBlock

d_SwitchRow

d_SwitchColumn

Bitwise XOR of each byte of matrix

Return the matrix to the input array.

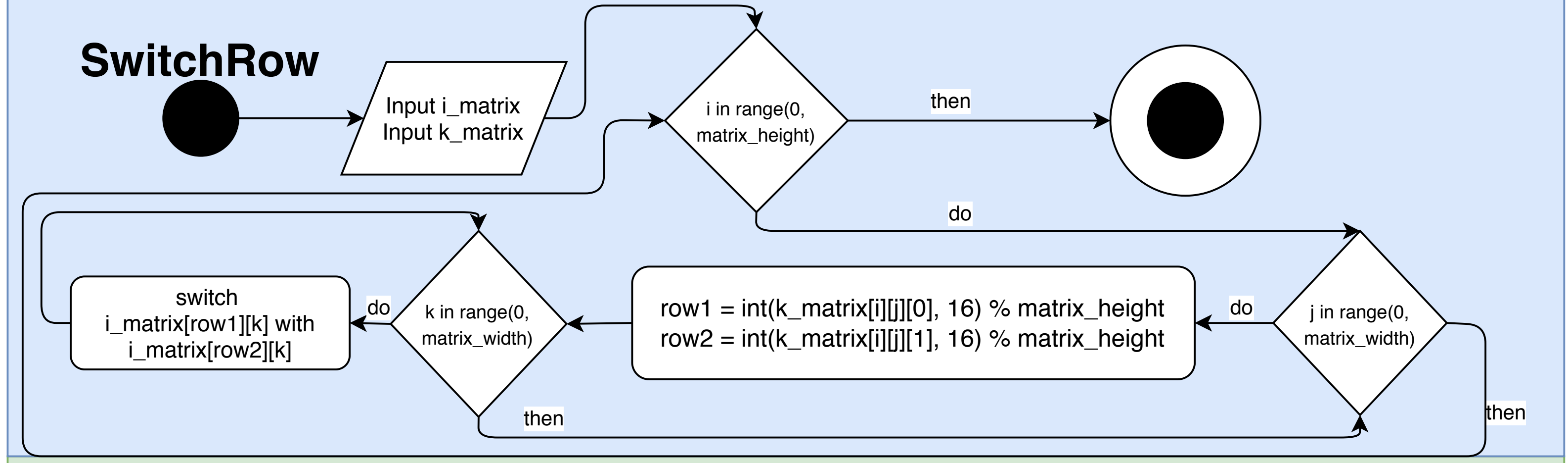
input_array[i] = i_matrix

Convert input_array to string (o_string)

Note:

No decryption functions (d_xxx) are diagrammed here because they are simply the reverse of the encryption functions (Run from end of matrix backwards instead

SwitchRow



Input i_matrix
Input k_matrix

i in range(0, matrix_height)

then

do

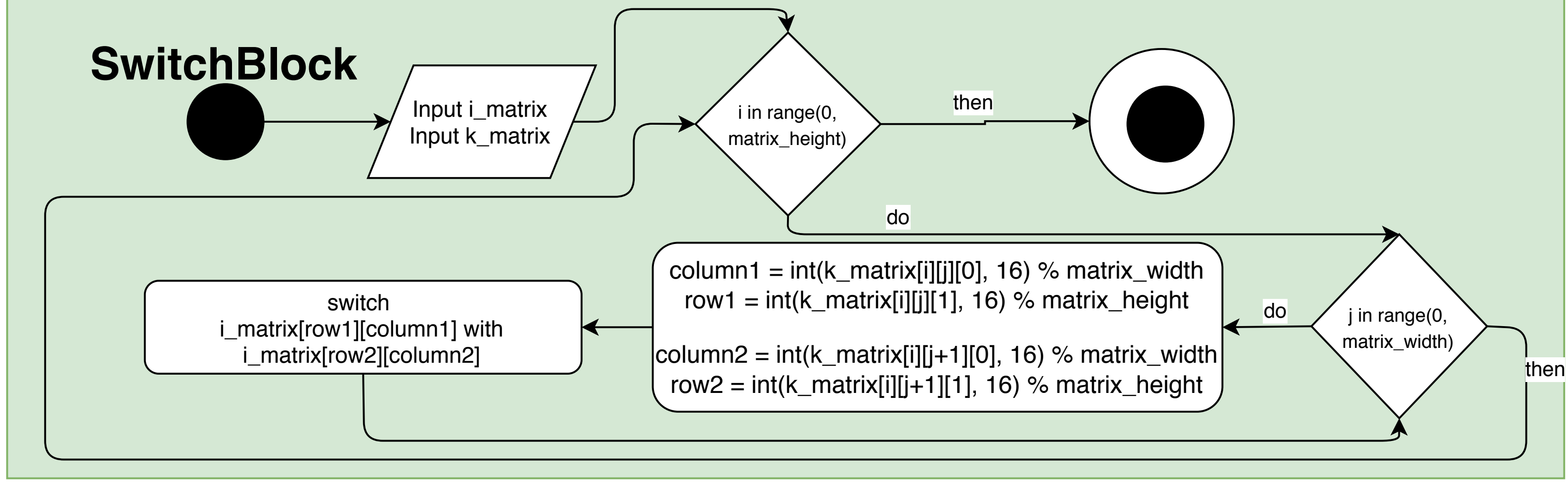
j in range(0, matrix_width)

row1 = int(k_matrix[i][j][0], 16) % matrix_height
row2 = int(k_matrix[i][j][1], 16) % matrix_height

k in range(0, matrix_width)

switch i_matrix[row1][k] with i_matrix[row2][k]

SwitchBlock



Input i_matrix
Input k_matrix

i in range(0, matrix_height)

then

do

j in range(0, matrix_width)

column1 = int(k_matrix[i][j][0], 16) % matrix_width
row1 = int(k_matrix[i][j][1], 16) % matrix_height

column2 = int(k_matrix[i][j+1][0], 16) % matrix_width
row2 = int(k_matrix[i][j+1][1], 16) % matrix_height

switch i_matrix[row1][column1] with i_matrix[row2][column2]