AES Algorithm

Configuration

Choose Key Standard:

```
keyProperties = 128
```

```
keyProperties = 128
```

```
config = Config();
if keyProperties == 128
    config.Nk = 4;
    config.Nb = 4;
    config.Nr = 10;
elseif keyProperties == 192
    config.Nk = 6;
    config.Nb = 4;
    config.Nr = 12;
elseif keyProperties == 256
    config.Nk = 8;
    config.Nb = 4;
    config.Nr = 14;
end
```

SBox configuration:

```
sBoxConfig = "Test"
```

```
sBoxConfig =
"Test"
```

```
if sBoxConfig == "Test"
    config.sBox = TestSBox();
    config.invSBox = TestInvSBox();
elseif sBoxConfig == "Random"
    % To change the random configuration, change the variables
    % in SBox.m and InvSBox.m files
    config.sBox = SBox();
    config.invSBox = InvSBox();
end
```

Enter the plaintext and cipherkey:

```
plainText = "sasank"
```

```
plainText =
"sasank"
```

```
cipherKey = "SpaceX"
```

```
cipherKey =
"SpaceX"
```

```
textInts = uint8(char(plainText));
% Test for AES-128
% textInts = uint8([0x32 0x43 0xf6 0xa8 0x88 0x5a 0x30 0x8d 0x31 0x31 0x98 0xa2 0xe0 0x37 0x07
% Test for AES-192
% textInts = uint8([0x00 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88 0x99 0xaa 0xbb 0xcc 0xdd 0xee
% Test for AES-256
% textInts = uint8([0x00 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88 0x99 0xaa 0xbb 0xcc 0xdd 0xee
nblocks = int32(length(textInts)) / (config.Nb * 4);
npad = uint8(config.Nb * 4 ...
    - (length(textInts) - config.Nb * 4 * nblocks));
textInts = [textInts repmat(npad, 1, npad)]
```

```
textInts = 1×16 uint8 row vector
   115    97   115    97   110   107    10    10    10    10    10    10    10 ···
```

```
keyInts = uint8(char(cipherKey));
% Test for AES-128
% keyInts = uint8([0x2b 0x7e 0x15 0x16 0x28 0xae 0xd2 0xa6 0xab 0xf7 0x15 0x88 0x09 0xcf 0x4f 0
% Test for AES-192
% keyInts = uint8([0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x0a 0x0b 0x0c 0x0d 0x0e 0
%                  0x10 0x11 0x12 0x13 0x14 0x15 0x16 0x17]);
% Test for AES-256
% keyInts = uint8([0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x0a 0x0b 0x0c 0x0d 0x0e 0
%                  0x10 0x11 0x12 0x13 0x14 0x15 0x16 0x17 0x18 0x19 0x1a 0x1b 0x1c 0x1d 0x1e 0
npad = uint8(config.Nk*4 - length(keyInts));
keyInts = [keyInts repmat(npad, 1, npad)]
```

```
keyInts = 1×16 uint8 row vector
    83   112    97    99   101    88    10    10    10    10    10    10    10 ···
```

```
config.K = keyInts;
```

Encryption

```
cipherText = [];
for i = 1:(config.Nb * 4):length(textInts)
    fprintf("\n\n-------------------------------------------------------------------------\n");
    fprintf("Process for blob %i:", int32(i/(config.Nb * 4) + 1));
    fprintf("\n-------------------------------------------------------------------------\n");
    blob = Cipher( ...
        textInts(i:i+config.Nb * 4-1), ...
        config.K, ...
        config.Nk, ...
```

```
        config.Nb, ...
        config.Nr ...
    );
    cipherText = [cipherText blob];
end
```

```
------------------------------------------------------------------------
Process for blob 1:
------------------------------------------------------------------------
Initial Round:
Initial State  :73 61 73 61 6e 6b a a a a a a a a a
FirstRoundKey  : 53 70 61 63 65 58 a a a a a a a a a
State          :20 11 12 2 b 33 0 0 0 0 0 0 0 0 0

Round 1:
SubBytes      : b7 82 c9 77 2b c3 63 63 63 63 63 63 63 63 63
ShiftRows     : b7 c3 63 63 2b 63 63 77 63 63 c9 63 63 82 63 63
MixColumns    : 2b ec 17 a4 e7 3f 17 93 c9 86 2c c9 5b ba 82 82
RoundKey      : 35 17 6 4 50 4f c e 5a 45 6 4 50 4f c e
AddRoundKey   : 1e fb 11 a0 b7 70 1b 9d 93 c3 2a cd b f5 8e 8c

Round 2:
SubBytes      : 72 f 82 e0 a9 51 af 5e dc 2e e5 bd 2b e6 19 64
ShiftRows     : 72 51 e5 64 a9 2e 19 e0 dc e6 82 5e 2b f af bd
MixColumns    : 96 80 5e ea c2 3e 8e c 4e c8 c7 a7 55 62 bd bc
RoundKey      : b3 e9 ad 57 e3 a6 a1 59 b9 e3 a7 5d e9 ac ab 53
AddRoundKey   : 25 69 f3 bd 21 98 2f 55 f7 2b 60 fa bc ce 16 ef

Round 3:
SubBytes      : 3f f9 d 7a fd 46 15 fc 68 f1 d0 2d 65 8b 47 df
ShiftRows     : 3f 46 d0 df fd f1 47 7a 68 8b d fc 65 f9 15 2d
MixColumns    : bb 7 b8 72 d4 b7 c 5e a7 8e e6 dd e2 9e c1 19
RoundKey      : 26 8b 40 49 c5 2d e1 10 7c ce 46 4d 95 62 ed 1e
AddRoundKey   : 9d 8c f8 3b 11 9a ed 4e db 40 a0 90 77 fc 2c 7

Round 4:
SubBytes      : 5e 64 41 e2 82 b8 55 2f b9 9 e0 60 f5 b0 71 c5
ShiftRows     : 5e b8 e0 c5 82 9 71 e2 b9 b0 41 2f f5 64 55 60
MixColumns    : 4a cb 69 2b 97 e1 54 3a cc 2e fa 7f 68 a2 9b f5
RoundKey      : 84 de 32 63 41 f3 d3 73 3d 3d 95 3e a8 5f 78 20
AddRoundKey   : ce 15 5b 48 d6 12 87 49 f1 13 6f 41 c0 fd e3 d5

Round 5:
SubBytes      : 8b 59 39 52 f6 c9 17 3b a1 7d a8 83 ba 54 11 3
ShiftRows     : 8b c9 a8 3 f6 7d 11 52 a1 54 39 3b ba 59 17 83
MixColumns    : e6 e2 c e1 33 6d 5f c9 a7 79 ca e3 10 b2 53 86
RoundKey      : 5b 62 85 a1 1a 91 56 d2 27 ac c3 ec 8f f3 bb cc
AddRoundKey   : bd 80 89 40 29 fc 9 1b 80 d5 9 f 9f 41 e8 4a

Round 6:
SubBytes      : 7a cd a7 9 a5 b0 1 af cd 3 1 76 db 83 9b d6
ShiftRows     : 7a b0 1 d6 a5 3 9b 9 cd 83 a7 af db cd 1 76
MixColumns    : e8 d4 a9 88 c6 1c 90 7e 17 8d f1 2d 96 2f 8e 56
RoundKey      : 76 88 ce d2 6c 19 98 0 4b b5 5b ec c4 46 e0 20
AddRoundKey   : 9e 5c 67 5a aa 5 8 7e 5c 38 aa c1 52 69 6e 76

Round 7:
SubBytes      : b 4a 85 be ac 6b 30 f3 4a 7 ac 78 0 f9 9f 38
ShiftRows     : b 6b ac 38 ac 7 9f be 4a f9 85 f3 0 4a 30 78
MixColumns    : 3f a 6b aa 6b a6 57 10 f2 c4 ac 5f 96 bc a2 8a
RoundKey      : 6c 69 79 ce 0 70 e1 ce 4b c5 ba 22 8f 83 5a 2
AddRoundKey   : 53 63 12 64 6b d6 b6 de b9 1 16 7d 19 3f f8 88
```

```
Round 8:
SubBytes      : ed fb c9 43 7f f6 4e 1d 56 7c 47 ff d4 75 41 c4
ShiftRows     : ed f6 47 c4 7f 7c 41 43 56 75 c9 1d d4 fb 4e ff
MixColumns    : 43 17 c2 e 78 7 44 3a e7 e1 8d 7c 14 14 a9 37
RoundKey      : 0 d7 e bd 0 a7 ef 73 4b 62 55 51 c4 e1 f 53
AddRoundKey   : 43 c0 cc b3 78 a0 ab 49 ac 83 d8 2d d0 f5 a6 64

Round 9:
SubBytes      : 1a ba 4b 6d bc e0 62 3b 91 ec 61 d8 70 e6 24 43
ShiftRows     : 1a e0 61 43 bc ec 24 6d 91 e6 4b 3b 70 ba 62 d8
MixColumns    : 2d 21 fd 29 5 7e af cd 78 a0 ac 73 8f 61 7d e3
RoundKey      : e3 a1 e3 a1 e3 6 c d2 a8 64 59 83 6c 85 56 d0
AddRoundKey   : ce 80 1e 88 e6 78 a3 1f d0 c4 f5 f0 e3 e4 2b 33

Final Round:
SubBytes      : 8b cd 72 c4 8e bc a c0 70 1c e6 8c 11 69 f1 c3
ShiftRows     : 8b bc e6 c3 8e 1c f1 c4 70 69 72 c0 11 cd a 8c
RoundKey      : 42 10 93 f1 a1 16 9f 23 9 72 c6 a0 65 f7 90 70
AddRoundKey   : c9 ac 75 32 2f a 6e e7 79 1b b4 60 74 3a 9a fc

Final State   : c9 ac 75 32 2f a 6e e7 79 1b b4 60 74 3a 9a fc
```

```
display(cipherText);
```

```
cipherText = 4×4 int16 matrix
   201    47   121   116
   172    10    27    58
   117   110   180   154
    50   231    96   252
```

Decryption

```
decryptedInts = [];
for i = 1:(config.Nb * 4):length(textInts)
    fprintf("\n\n------------------------------------------------------------------------\n");
    fprintf("Process for blob %i:", int32(i/(config.Nb * 4) + 1));
    fprintf("\n------------------------------------------------------------------------\n");
    decryptedBlob = InvCipher( ...
        cipherText(i:i+config.Nb * 4-1), ...
        config.K, ...
        config.Nk, ...
        config.Nb, ...
        config.Nr ...
    );
    decryptedInts = [decryptedInts decryptedBlob];
end
```

```
------------------------------------------------------------------------
Process for blob 1:
------------------------------------------------------------------------
Initial Round:
Initial State  :c9 ac 75 32 2f a 6e e7 79 1b b4 60 74 3a 9a fc
FirstRoundKey  : 42 10 93 f1 a1 16 9f 23 9 72 c6 a0 65 f7 90 70
```

```
State          :8b bc e6 c3 8e 1c f1 c4 70 69 72 c0 11 cd a 8c

Round 1:
InvShiftRows  : 8b cd 72 c4 8e bc a c0 70 1c e6 8c 11 69 f1 c3
InvSubBytes   : ce 80 1e 88 e6 78 a3 1f d0 c4 f5 f0 e3 e4 2b 33
RoundKey      : e3 e3 a8 6c a1 6 64 85 e3 c 59 56 a1 d2 83 d0
AddRoundKey   : 2d 21 fd 29 5 7e af cd 78 a0 ac 73 8f 61 7d e3
InvMixColumns : 1a e0 61 43 bc ec 24 6d 91 e6 4b 3b 70 ba 62 d8

Round 2:
InvShiftRows  : 1a ba 4b 6d bc e0 62 3b 91 ec 61 d8 70 e6 24 43
InvSubBytes   : 43 c0 cc b3 78 a0 ab 49 ac 83 d8 2d d0 f5 a6 64
RoundKey      : 0 0 4b c4 d7 a7 62 e1 e ef 55 f bd 73 51 53
AddRoundKey   : 43 17 c2 e 78 7 44 3a e7 e1 8d 7c 14 14 a9 37
InvMixColumns : ed f6 47 c4 7f 7c 41 43 56 75 c9 1d d4 fb 4e ff

Round 3:
InvShiftRows  : ed fb c9 43 7f f6 4e 1d 56 7c 47 ff d4 75 41 c4
InvSubBytes   : 53 63 12 64 6b d6 b6 de b9 1 16 7d 19 3f f8 88
RoundKey      : 6c 0 4b 8f 69 70 c5 83 79 e1 ba 5a ce ce 22 2
AddRoundKey   : 3f a 6b aa 6b a6 57 10 f2 c4 ac 5f 96 bc a2 8a
InvMixColumns : b 6b ac 38 ac 7 9f be 4a f9 85 f3 0 4a 30 78

Round 4:
InvShiftRows  : b 4a 85 be ac 6b 30 f3 4a 7 ac 78 0 f9 9f 38
InvSubBytes   : 9e 5c 67 5a aa 5 8 7e 5c 38 aa c1 52 69 6e 76
RoundKey      : 76 6c 4b c4 88 19 b5 46 ce 98 5b e0 d2 0 ec 20
AddRoundKey   : e8 d4 a9 88 c6 1c 90 7e 17 8d f1 2d 96 2f 8e 56
InvMixColumns : 7a b0 1 d6 a5 3 9b 9 cd 83 a7 af db cd 1 76

Round 5:
InvShiftRows  : 7a cd a7 9 a5 b0 1 af cd 3 1 76 db 83 9b d6
InvSubBytes   : bd 80 89 40 29 fc 9 1b 80 d5 9 f 9f 41 e8 4a
RoundKey      : 5b 1a 27 8f 62 91 ac f3 85 56 c3 bb a1 d2 ec cc
AddRoundKey   : e6 e2 c e1 33 6d 5f c9 a7 79 ca e3 10 b2 53 86
InvMixColumns : 8b c9 a8 3 f6 7d 11 52 a1 54 39 3b ba 59 17 83

Round 6:
InvShiftRows  : 8b 59 39 52 f6 c9 17 3b a1 7d a8 83 ba 54 11 3
InvSubBytes   : ce 15 5b 48 d6 12 87 49 f1 13 6f 41 c0 fd e3 d5
RoundKey      : 84 41 3d a8 de f3 3d 5f 32 d3 95 78 63 73 3e 20
AddRoundKey   : 4a cb 69 2b 97 e1 54 3a cc 2e fa 7f 68 a2 9b f5
InvMixColumns : 5e b8 e0 c5 82 9 71 e2 b9 b0 41 2f f5 64 55 60

Round 7:
InvShiftRows  : 5e 64 41 e2 82 b8 55 2f b9 9 e0 60 f5 b0 71 c5
InvSubBytes   : 9d 8c f8 3b 11 9a ed 4e db 40 a0 90 77 fc 2c 7
RoundKey      : 26 c5 7c 95 8b 2d ce 62 40 e1 46 ed 49 10 4d 1e
AddRoundKey   : bb 7 b8 72 d4 b7 c 5e a7 8e e6 dd e2 9e c1 19
InvMixColumns : 3f 46 d0 df fd f1 47 7a 68 8b d fc 65 f9 15 2d

Round 8:
InvShiftRows  : 3f f9 d 7a fd 46 15 fc 68 f1 d0 2d 65 8b 47 df
InvSubBytes   : 25 69 f3 bd 21 98 2f 55 f7 2b 60 fa bc ce 16 ef
RoundKey      : b3 e3 b9 e9 e9 a6 e3 ac ad a1 a7 ab 57 59 5d 53
AddRoundKey   : 96 80 5e ea c2 3e 8e c 4e c8 c7 a7 55 62 bd bc
InvMixColumns : 72 51 e5 64 a9 2e 19 e0 dc e6 82 5e 2b f af bd

Round 9:
InvShiftRows  : 72 f 82 e0 a9 51 af 5e dc 2e e5 bd 2b e6 19 64
InvSubBytes   : 1e fb 11 a0 b7 70 1b 9d 93 c3 2a cd b f5 8e 8c
RoundKey      : 35 50 5a 50 17 4f 45 4f 6 c 6 c 4 e 4 e
AddRoundKey   : 2b ec 17 a4 e7 3f 17 93 c9 86 2c c9 5b ba 82 82
InvMixColumns : b7 c3 63 63 2b 63 63 77 63 63 c9 63 63 82 63 63
```

```
Final Round:
InvShiftRows : b7 82 c9 77 2b c3 63 63 63 63 63 63 63 63 63 63
InvSubBytes  : 20 11 12 2 b 33 0 0 0 0 0 0 0 0 0 0
RoundKey     : 53 70 61 63 65 58 a a a a a a a a a a
AddRoundKey  : 73 61 73 61 6e 6b a a a a a a a a a a

Final State   : 73 61 73 61 6e 6b a a a a a a a a a a
```

```
decryptedInts = decryptedInts(:);
if range(decryptedInts(end-decryptedInts(end)+1:end)) == 0
    fprintf("Decryption successful!\nPlain text: %s", ...
        char(decryptedInts(1:end-decryptedInts(end))));
else
    fprintf("Decryption failed :(");
end
```

```
Decryption successful!
Plain text: sasank
```