S-AES

Mestrado em Cibersegurança

Sofia Teixeira Vaz

S-AES

Criptografia Aplicada

Mestrado em Cibersegurança

Sofia Teixeira Vaz
(92968) sofiateixeiravaz@ua.pt

27/11/2021

Conteúdo

1 Introdução **1**
1.1 AES 1
1.2 S-AES 2

2 Implementação **3**
2.1 AES 3
2.2 S-AES 3
2.2.1 Escolha de ronda 3
2.2.2 AddRoundKey 4
2.2.3 SubBytes 4
2.2.4 ShiftRows 4
2.2.5 MixColumns 5

3 Aplicações **5**
3.1 Encrypt 5
3.2 Decrypt 5
3.3 Speed 6

Capítulo 1

Introdução

Este documento explicita como a implementação do Shuffled Advanced Encryption Standard (S-AES) foi feita.

1.1 AES

A cifra Rijndael é a cifra que, em 2000, o National (americano) Institute of Standards and Technology (NIST) elegeu como Advanced Encryption Standard (AES). Foi escolhido numa competição anunciada em 1997, e 15 cifras competiram. Rijndael, foi escolhida devido à sua rapidez, segurança, entre outros atributos.

AES é uma cifra em bloco, sendo este de 128 bits. A implementação varia com o comprimento da chave, mas o foco do documento será a versão com uma chave de 128 bits.

Antes do processo de cifra começar, a chave será expandida, de modo a ser do tamanho de 11 chaves de 128 bits. Cada ronda de cifra (ou decifra) usará uma chave diferente desta *pool*.

Depois disto, o *input* será XOR'd com a primeira chave da expansão (chave 0). Isto constitui a ronda 0.

Da ronda 1 a 9, o processo será sempre o mesmo. Cada byte do estado atual é trocado pelo seu equivalente na *s-box* definida pela cifra (fase SubBytes). Depois disto, cada linha do input (sendo este uma tabela 4x4) será *shifted* - ShiftRows. A fase MixColumns é vista como a computacionalmente mais exigente. Nesta, cada coluna é multiplicada por uma matriz 4x4, sendo o resultado a coluna que tomará o lugar da usada no seu cálculo. Finalmente, o estado atual é XOR'd com a chave relativa à ronda - AddRoundKey.

Na ronda 10, todas as operações mencionadas acima são feitas, excetuando MixColumns. Isto conclui o processo de cifra.

O processo de decifra é equivalente, sendo as operações feitas na ordem oposta. As operações também serão substituídas pelas suas inversas - excetuando a AddRoundKey, uma vez que um XOR é negado por um XOR com o

mesmo valor.

A cifra discutida neste documento é uma adaptação do AES.

1.2 S-AES

O S-AES tem alguns pontos de divergência com o AES. Primeiramente, recebe uma segunda chave de 128 bits, sendo esta a *shuffling key*. Esta irá definir qual das rondas, de 1 a 9, será substituída pela ronda alterada.

Nesta ronda alterada, durante o AddRoundKey, a *shuffling key* será usada para adicionar um *offset* numa das matrizes usadas. Na SubBytes, a *s-box* usada é uma versão *shuffled* pela segunda chave fornecida. Isto significa que a *s-box* usada na decifra será, necessariamente, diferente. Na fase ShiftRows, os *shifts* efetuados serão permutados pela *shuffling key*. Durante o MixColumns, a SK será usada para adicionar um *offset* às colunas escolhidas.

Também foi especificado que o input será processado usando o modo Electronic Codebook (ECB) e o padding será PKCS#7.

Capítulo 2

Implementação

2.1 AES

A implementação do AES utilizado neste projeto foi inteiramente feita pela autora do documento. A *s-box* usada é a Nyberg *s-box*. A RCon usada na expansão da chave é tabelada.

Para perceber certos pontos do algoritmo, foram explorados apontamentos de uma aula de uma universidade externa (`https://engineering.purdue.edu/kak/compsec/NewLectures/Lecture8.pdf`), e ainda foi consultado um repositório - `https://github.com/moserware/AES-Illustrated`.

A implementação do AES foi bem sucedida, uma vez que a cifra e posterior decifra de um *plaintext* qualquer resulta nesse mesmo *plaintext*.

2.2 S-AES

A implementação da nova versão do AES também foi feita, na íntegra, pela autora do documento. Para minimizar o número de comparações feitas, as funções que implementam S-AES são diferentes das que implementam AES. Istou significa que cada ronda apenas fará uma comparação, em vez de 4, que seria o caso ao fazer estas comparações em cada função.

Alguns dos testes são visíveis no ficheiro test.py.

2.2.1 Escolha de ronda

Uma das especificações da nova cifra é que a escolha da ronda alterada será equiprovável para qualquer valor entre 1 e 9. Por isso, esta é calculada durante a inicialização da cifra, e será igual à soma de cada byte da SK em módulo 9, somando 1. Isto terá resultados equiprováveis entre 1 e 9.

2.2.2 AddRoundKey

Na fase AddRoundKey, é descrito que haverão até 16 *offsets* possíveis. Para o offset ser definido, este será igual ao valor que estiver na posição X da SK em módulo 16, sendo X o número da ronda atual. O offset será aplicado na chave de cifra ou decifra, em vez do estado atual, uma vez que a chave apresenta-se como uma lista enquanto que o estado será um *array* bidimensional. Ao aplicar o *offset* na chave, menos cálculos serão necessários.

2.2.3 SubBytes

Durante a inicialização da cifra, é definida a *s-box* e *s-box* inversa *shuffled*. Para a criação da primeira, a *s-box* inicial irá passar por uma função que troca cada posição desta com o valor que estiver na posição equivalente da SK. O código da função é visível abaixo:

```
def generates_box(self):
        box = [i for i in self.s_box]
        current = 0
        idx = 0

        for i in range(0, 256):
            current = self.sk[i%len(self.sk)]
            box[current], box[i] = box[i], box[current]


        return box
```

A nova *s-box* usada na decifra será a inversa desta. O código usado para a gerar encontra-se abaixo:

```
def invert(self, sbox):
        toRet = []
        while(len(toRet)!= len(sbox)):
            toRet.append(0);
        for idx, val in enumerate(sbox):
            toRet[val] = idx
        return toRet
```

2.2.4 ShiftRows

Na definição da cifra, é gerado um *array* de *offsets* = [0,1,2,3]. Caso a SK seja introduzida, isto é, se esteja num caso de cifra com S-AES, este array será *shuffled*. O método de *shuffle* será como o descrito na secção SubBytes.

Durante a decifra, o *offset* aplicado a cada linha será igual a 4 - offset[linha].

2.2.5 MixColumns

Na fase de MixColumns, as matrizes de cifra e decifra mantêm-se como no AES, no entanto, as colunas do estado serão escolhidas com um *offset*. Este será calculado usando o mesmo método de AddRoundKey, e será subtraído (em vez de somado) durante a decifra.

Capítulo 3

Aplicações

Este capítulo visa a descrever o funcionamento e modo de utilização de cada uma das aplicações criadas. De notar que os comandos devem ser efetuados no diretório que terá todos os ficheiros (saes.py, encrypt.py, decrypt.py e speed.py).

3.1 Encrypt

Para executar esta aplicação, o utilizador deverá correr o comando "python3 encrypt.py X Y", sendo que o utilizador deverá substituir X pela chave de cifra e, se desejar usar o S-AES, Y pela *shuffling key*. Ambas as chaves deverão ter, no mínimo, 16 caracteres.

Será pedido ao utilizador um *plaintext* que será cifrado. O *ciphertext* gerado surgirá como uma cadeia de caracteres e, para mais fácil leitura, um *array* de inteiros.

3.2 Decrypt

Para executar a aplicação, o utilizador deverá correr o comando "python3 decrypt.py X Y", sendo X a chave de cifra e, se desejar usar o S-AES, Y a *shuffling key*. As especificações definidas na secção da aplicação Encrypt são aplicáveis aqui. Se o *input* inserido não tiver comprimento múltiplo de 16 caracteres, a execução da aplicação não ocorrerá, uma vez que isto dará origem a situações difíceis em termos de *padding*.

	SAES	AES
1	0.10409283638000488	0.0007524490356445312
2	0.10508561134338379	0.0007488727569580078
3	0.10573673248291016	0.000743865966796875
4	0.10546422004699707	0.0007443428039550781
5	0.10436701774597168	0.0007457733154296875

Tabela 3.1: Medidas de velocidade da implementação de S-AES e de uma implementação de AES externa

	SAES	AES
1	0.10123825073242188	0.0007035732269287109
2	0.10412859916687012	0.0007300376892089844
3	0.10399365425109863	0.0007338523864746094
4	0.1027064323425293	0.0007312297821044922
5	0.10294771194458008	0.0007319450378417969

Tabela 3.2: Medidas de velocidade da implementação de AES presente na biblioteca criada e de uma implementação de AES externa

3.3 Speed

Para executar esta aplicação, execute o comando "python3 speed.py".

Foi notado que a implementação de S-AES é, sensivelmente, 145 vezes mais lenta do que a implementação de AES presente no módulo Crypto.Cipher. Apesar da grande diferença, esta será de esperar, uma vez que que não foram feitas otimizações ao nível de execução de C, o que provavelmente acontece na outra implementação. Na tabela 3.1 estão os resultados de 5 intervalos de tempo que cada módulo demorou a cifrar e decifrar 256 blocos de 16 bytes, sendo que cada um destes será o melhor caso (ou seja, menor intervalo de tempo) de 50 ciclos. Todos os intervalos de tempo foram medidos em segundos.

Em termos da implementação presente do AES, os dados são os visíveis na tabela 3.2.

A execução do AES em vez do S-AES é mais rápida, mas continua bastante longe da velocidade da implementada na biblioteca PyCrypto. A diferença entre os melhores casos é 0.003 segundos, o que equivale a 3%.