Segurança Computacional
Advanced Encryption Standard

Pedro Henrique de Brito Agnes, 18/0026305
Pedro Pessoa Ramos, 18/0026488

Dep. Ciência da Computação - Universidade de Brasília (UnB)

1 Implementação do AES

A implementação do AES foi feita fixa para a versão de 128 bits e para isso foi criado o arquivo `aes.py` na pasta `src` usando a linguagem python preferencialmente na versão 3.6 ou acima. Para executar o programa para cifrar o arquivo `sample/dupla.jpg`, por exemplo com uma chave pseudo-aleatória usando o padrão de *rounds* no modo ECB e salvar o criptograma no arquivo `sample/10r.jpg`, deve ser usado o seguinte comando:

```
1 python src/aes.py sample/dupla.jpg -o sample/10r.jpg
```

O primeiro argumento que o programa recebe é o arquivo que contém a mensagem. Logo em seguida, está sendo passado o argumento `-o`, que é obrigatório e representa o arquivo onde será impresso o criptograma. Como não foi passada uma chave pré-existente, será gerada a chave durante a execução e ela será salva na pasta `keys` e será impressa uma mensagem informando o nome exato do arquivo que a contém. Existem outros argumentos opcionais que podem ser listados com o `-h`:

```
1 python src/aes.py -h
```

Segue a lista de argumentos aceitos:

- **-k** - Arquivo com a chave para criptografar/descriptografar. Argumento obrigatório se for acionada a opção para descriptografar.
- **-o** - Arquivo onde será feito o output. Obrigatório.
- **-r** - Número positivo que representa a quantidade de *rounds* que o AES irá usar. Se não passado um valor, será usado o padrão de 10.
- **-d** - Argumento que indica que o programa vai descriptografar. Deve ser passado no final do comando sem parâmetros adicionais.

Portanto, como exemplo, para cifrar o mesmo arquivo do exemplo acima usando a mesma chave (ex.: `keys/key_1`) com 1 *round* e colocando o output no arquivo `sample/1r.jpg`, pode ser usado o seguinte comando:

```
python src/aes.py sample/dupla.jpg -k keys/key_1 -r 1 -o sample/1r.jpg
```

Da mesma forma, para decifrá-lo no arquivo `sample/decoded.jpg` por exemplo, pode-se usar o comando abaixo:

```
python src/aes.py sample/1r.jpg -k keys/key_1 -r 1 -o
    sample/decoded.jpg -d
```

1.1 Aspectos Técnicos

Para a implementação descrita acima, temos 2 arquivos principais que realizam as operações do AES, ambos na pasta `src/symmetric`. O `cipher` é usado para cifrar e o `decipher` é usado para decifrar. De forma resumida, o funcionamento da cifração se dá na seguinte forma:

```
self.addRoundKey()          # realiza o XOR com a chave inicial
for i in range(rounds):     # itera sobre os rounds
    self.expandKey(i)       # transforma a chave
    self.subBytes()
    self.shiftRows()
    if i != rounds-1:       # não realiza o mixColumns no último round
        self.mixColumns()

    self.addRoundKey()      # Realiza o XOR com a chave da rodada
```

Já para a decifração, é feito o contrário das operações, sendo que algumas foram modificadas apesar de ter o mesmo nome e segue a seguinte rotina:

```
for x in range(rounds):     # vai até a chave final obtida na cifração
    self.expandKey(x)

for i in range(rounds):     # itera sobre os rounds
    self.addRoundKey()      # realiza o XOR com a chave da rodada
    if i != 0:              # não realiza o mixColumns no round inicial
        self.mixColumns()

    self.shiftRows()
    self.subBytes()
    self.shrinkKey(rounds-1-i) # operação reversa da expandKey

self.addRoundKey()          # Realiza o XOR com a chave inicial
```