Segurança em Sistemas de Computação

Desafio 05
(AES - 128 bits)

INF01045 – Comunicação – Turma U – Prof. Raul Weber

Luiz Gustavo Frozi de Castro e Souza - Cartão 96957
Mário César Gasparoni Jr. - Cartão 151480

Setembro de 2013

Texto Cifrado (notação hexadecimal):

E6A2C84DE51FA7619BA0A3DA6C306F4B
137CB914B18F1DD6E7BF45C5634FE31E
801F7E836873C2843BBB19AB5DA1BDAE
5953C59D28F702E00EE59FB883DE9C67
14DAAA8628744A46A5E39E2A909AFF4D
37CCF46ED9B29423E986B53B498D9562
1610AAD8EBF8E2805B19F58F9F5930A9
52C88E7E5054B7C0F5A93688FF863E00
AF0335ECC7FA9B3AA8895D986A1FC72E
232064A8C864608D46E619763776EA4E
1580712BE23F5C623C595510B9397146
7C04CE1EB8CD98EF5069EF001010A7E8
5ACB7367B5A09EAAEE48332BC04207E3
90075010AB98FC75BB4EF1B8A021136B
0C5A3FB8084865F8F20909D4F7364684
0950E37C20DAF88DCB1D61BF7169FFE8
96AF75E1AFF231FC9F02B3F58E1713C7

Texto Decifrado:

O proximo desafio vai usar conceitos de criptografia de chave publica. Vai ser preciso utilizar uma biblioteca de grandes numeros, e sera necessario quebrar um esquema Diffie-Hellman multiplicativo de troca de chaves de sessao.
LOMKMGOKDCHKCCMMGGCBFIIMMNPCF

Chave:

Key2Group25[‘3s.

Passo a Passo da Análise efetuada:

Foi informado que a chave a ser quebrada iniciava por **Key2GroupNNXXXXX,** onde NN era o número do grupo e XXXXX uma string de 5 caracteres em código ASCII cujo valor de cada caractere poderia variar de 33 a 126 em decimal. Com base nisso o problema consistia em encontrar uma string de 5 caracteres que completasse a chave, de modo que o texto descriptografado fosse legível.

O método usado foi de **força bruta,** ou seja, foram tentadas todas as combinações possíveis

de caracteres, variando-se em ordem crescente de **!!!!!** (menor valor ASCII, ou seja, 33 em decimal) a **~~~~~** (maior valor ASCII, ou seja, 126 em decimal).
A heurística usada era verificar se um bloco descriptografado possuía caracteres legíveis e sem acentos (caracteres com código ASCII entre 32 e 126, ou seja, considerando o caractere de espaço em branco, cujo valor ASCII é 32). Quando o programa detectava uma string decifrada com essas características, a mesma era exibida na tela.
A linguagem escolhida para o programa de força bruta foi **Python**. A escolha da linguagem teve dois fatores muito importantes:

1. Ela é uma linguagem fácil de de desenvolver algoritmos e muito prática, como outras linguagens de script;
2. Apesar de interpretada, as bibliotecas de funções, como a de criptografia AES são compiladas e altamente otimizadas para o hardware onde está sendo executada.

Com base nisso o programa deveria ser o mais simples possível (menos código interpretado), deixando para as funções de biblioteca a maior parte do trabalho.
Para verificar os parâmetros de entrada da biblioteca de funções de criptografia AES, foi feito um pequeno programa de teste, onde criptografamos o texto **"Texto para teste"** (hexadecimal: 546578746F20706172617261207465737465) tendo como chave "**essasenhaehfraca**" (hexadecimal: 6573736173656E686165686672616361), de modo a obter como texto cifrado (em hexadecimal): A506A19333F306AC2C62CBE931963AE7. Obtendo esses resultados conseguimos a configuração do modo de operação da biblioteca.
O computador utilizado foi um **Ultrabook** com processador **Intel Core i3** com dois núcleos que suportam duas threads cada, velocidade de **2,4 GHz**, **6GB** de memória RAM. O computador levou **15h para encontrar a chave**.
Abaixo seguem os códigos-fonte utilizados.

Programa de Teste (testeaes.py):

```
1.  ## Universidade Federal do Rio Grande do Sul - Instituto de Informatica
2.  ## Departamento de Informatica Aplicada
3.  ## Seguranca em Sistemas de Computacao - 2013/2
4.  ## Professor: Raul Fernando Weber
5.  ## Alunos: Luiz Gustavo Frozi e  Mario Gasparoni Junior
6.  ##
7.  ##
8.  ##
9.  ## Script para executar o teste de algoritmo do professor.
10.
11. from Crypto.Cipher import AES
12. from Crypto import Random
13.
14.
15. cyphered = [] #cada bloco eh um elemento da lista
16. cyphered.append("Texto para teste")
17.
18. key = b'essasenhaehfraca'
19. cipher = AES.new(key, AES.MODE_ECB)
20. for block in cyphered:
21.     cyphered = cipher.encrypt(block)#criptografa bloco
22.     for c in cyphered: #imprime cada caracter em ascii
23.         print hex(ord(c))
```

Programa de Força Bruta (aes.py):

```
## Universidade Federal do Rio Grande do Sul - Instituto de Informatica
## Departamento de Informatica Aplicada
## Seguranca em Sistemas de Computacao - 2013/2
## Professor: Raul Fernando Weber
## Alunos: Luiz Gustavo Frozi e  Mario Gasparoni Junior
##
## Desafio 5
##
##
## Script que executa forca bruta sobre o AES.
from Crypto.Cipher import AES
from Crypto import Random


cyphered = [] #cada bloco eh um elemento da lista
cyphered.append("E6A2C84DE51FA7619BA0A3DA6C306F4B".decode("hex"))
cyphered.append("137CB914B18F1DD6E7BF45C5634FE31E".decode("hex"))
cyphered.append("801F7E836873C2843BBB19AB5DA1BDAE".decode("hex"))
cyphered.append("5953C59D28F702E00EE59FB883DE9C67".decode("hex"))
cyphered.append("14DAAA8628744A46A5E39E2A909AFF4D".decode("hex"))
cyphered.append("37CCF46ED9B29423E986B53B498D9562".decode("hex"))
cyphered.append("1610AAD8EBF8E2805B19F58F9F5930A9".decode("hex"))
cyphered.append("52C88E7E5054B7C0F5A93688FF863E00".decode("hex"))
cyphered.append("AF0335ECC7FA9B3AA8895D986A1FC72E".decode("hex"))
cyphered.append("232064A8C864608D46E619763776EA4E".decode("hex"))
cyphered.append("1580712BE23F5C623C595510B9397146".decode("hex"))
cyphered.append("7C04CE1EB8CD98EF5069EF001010A7E8".decode("hex"))
cyphered.append("5ACB7367B5A09EAAEE48332BC04207E3".decode("hex"))
cyphered.append("90075010AB98FC75BB4EF1B8A021136B".decode("hex"))
cyphered.append("0C5A3FB8084865F8F20909D4F7364684".decode("hex"))
cyphered.append("0950E37C20DAF88DCB1D61BF7169FFE8".decode("hex"))
cyphered.append("96AF75E1AFF231FC9F02B3F58E1713C7".decode("hex"))

minChar = 33
maxChar = 126
minCharWhitespace = 32

def isBadString(text):
    for c in text:
        if ((ord(c) <minCharWhitespace) or (ord(c) >maxChar)):
            return 1
    return 0

for i in range(minChar,maxChar):
    for j in range(minChar,maxChar):
        for k in range(minChar,maxChar):
            for l in range(minChar,maxChar):
                for m in range(minChar,maxChar):
                    key = b'Key2Group25'+chr(i)+chr(j)+chr(k)+chr(l)+chr(m)
                    cipher = AES.new(key, AES.MODE_ECB)
                    for block in cyphered:
                        plain = cipher.decrypt(block)#decifra bloco
                        if (isBadString(plain) == 0):
                            print "Key: " +key
                            print plain
                        else:
                            break
```