Teoria de Números Computacional 21/22

Trabalho Prático 1

Grupo:

- Ivo Miguel Gomes Lima (A90214)
- Tiago dos Santos Silva Peixoto Carriço (A91695)

Contextualização

Neste trabalho foi-nos pedida a implementação e explicação de um dos ataques feitos ao RSA, o famoso ataque de $Hastad$.

Para o efeito recorremos ao SageMath (https://www.sagemath.org) e a alguns documentos bibliográficos o D. Boneh, Twenty years of attacks on the RSA cryptosystem (http://crypto.stanford.edu/~dabo/pubs/papers/RSA-survey.pdf) e Glenn Durfee, Cryptanalysis of RSA Using Algebraic and Lattice Methods (pag 24) (http://theory.stanford.edu/~gdurf/durfee-thesis-phd.pdf) para conseguirmos exemplificar e adquirir a precessão de como o ataque funciona.

No final do documento apresentamos alguns exemplos a chaves RSA com expoente 3.

Criação do RSA

O algoritmo de criptografia Rivest-Shamir-Adleman (RSA) é um algoritmo de criptografia assimétrico amplamente utilizado para transmissão segura de dados. A criptografia assimétrica usa um par de chaves matematicamente ligadas para criptografar e descriptografar dados.

O RSA envolve um par de chaves, uma chave pública gerada através de dois números primos grandes p e q, tendo estes valores a ordem de 10^{100}, esta chave pode ser conhecida por todos, já a chave privada deve ser mantida em sigilo. Toda mensagem cifrada usando uma chave pública só pode ser decifrada usando a respectiva chave privada. Após a geração destes valores é calculado um n que é a somente a multiplicação dos valores p e q, isto é $n = p \times q$. Em seguida devemos calcular a Função totiente de Euler (https://pt.wikipedia.org/wiki/Função_totiente_de_Euler) que geralmente é apresentada como $\phi(n) = (p_1 - 1) \times p_1^{k_1 - 1} \dots (p_r - 1) \times p_r^{k_r - 1}$. Para o nosso caso, como $k = 1$, a expressão terá o valor de $(p - 1) \times (q - 1)$, sendo que este valor m deverá satisfazer a condição de $MDC(m, e) = 1$, tendo o valor de e sido estipulado no enunciado como 3, caso contrário, terão de ser recalculados os valores de p, q e m. Por fim, temos de calcular o inverso multiplicativo de e em $\mod m$ que será guardado numa variável d.

Implementação

In [1]:
```
def RSA(nbits = 512, e = 3):
    p = random_prime(2^(nbits//2), lbound=2^(nbits//2-1))
    q = random_prime(2^(nbits//2+1), lbound=2^(nbits//2))
    n = p*q
    m = (p-1)*(q-1)
    while gcd(e, m) != 1:
        p = random_prime(2^(nbits//2), lbound=2^(nbits//2-1))
        q = random_prime(2^(nbits//2+1), lbound=2^(nbits//2))
        n = p*q
        m = (p-1)*(q-1)
    d = power_mod(e, -1, m) # é a chave privada, inverso de e mod m
    return (n, e), d
```

Encriptação e Decriptação

Para transformar uma mensagem $mens$, numa mensagem $cripto$, recorremos à iteração de cada um dos caracteres nela contida por forma a encriptá-la através da potenciação modular. Neste processo utilizamos a chave pública do destinatário, o n e o e, por fim acrescentamento o resultado à lista, a fórmula será algo assim $mens[i]^e = crypto[i] \mod n$.

Para recuperar a mensagem original, fazemos o mesmo para quando encriptamos utilizamos a potenciação modular mas desta vez utilizamos a chave privada do recetor. A fórmula fica muito semelhante à anteriormente, isto é $crypto[i]^d \equiv mens[i] \mod n$.

Implementação

In [2]:
```
def RSA_encriptar(mens, ch_pub):
    n, e = ch_pub
    cripto = []
    for ch in mens:
        cripto.append( power_mod(ord(ch), e, n) )
    return cripto

def RSA_desencriptar(cripto, ch_pub, ch_priv):
    n, _ = ch_pub
    decif = []
    for ch in cripto:
        decif.append( chr(power_mod(ch, ch_priv, n)) )
    return "".join(decif)
```

Ataque de $Hastad$

Este ataque ocorre quando o expoente é pequeno, temos uma mensagem longa e o remetente envia a mesma mensagem para vários destinatários usando o mesmo e.

Para que o ataque ocorra precisaremos de capturar pelo menos e mensagens encriptadas da mensagem m.

Utilizando o nosso caso de estudo onde e = 3, podemos concluir que $M = m^3$, isto é ficaremos com um sistema de equações semelhante ao apresentado abaixo:

$$\begin{cases} M \equiv c_1[n_1] \\ M \equiv c_2[n_2] \\ M \equiv c_3[n_3] \end{cases}$$

De seguida, teremos de resolver o sistema por forma a encontrar uma solução, neste ponto usaremos o Teorema chinês do resto (https://pt.wikipedia.org/wiki/Teorema_chinês_do_resto) pois este garante a existência de uma solução $\mod N = \prod_{i=1}^{e} n_i = n_1 \times n_2 \times n_3,\ para\ MDC(n_i, n_j) = 1\ e\ i \neq j$. Esta condição é satisfeita pois, caso contrário, seria possível calcular um fator de um dos n_i através do $MDC(n_i, n_j)$.

Para encontrar a solução definimos o $N_i = \frac{N}{n_i}$.

Sabendo que o $MDC(N_i, n_i) = 1$, podemos assumir que $(u_i \times N_i) + (v_i \times n_i) = 1$ onde u_i é o inverso de $N_i \mod n_i$. Em suma, $u_i \times N_i \equiv 1 \mod n_i$.

Percebemos ainda que $u_i \times N_i \equiv 0 \mod n_j$, para $j \neq i$ porque N_i é múltiplo de n_j por definição.

Agora podemos construir uma solução para o sistema de equações, findando em $M \equiv \sum_{i=1}^{e} c_i \times u_i \times N_i \mod N$.

Todo o processo enunciado acima já se encontra implementado na biblioteca do SageMath (https://www.sagemath.org) pela função `crt`.

Por fim como $m < n_i$ então $m^3 < N$, temos de calcular a raiz cúbica da solução por forma a obter a mensagem original, que será feito através da função `nth_root(3)` da mesma biblioteca.

In [3]:
```
def hastad(cifras, chaves_publicas):
    res = []
    for i in range(len(cifras[0])):
        x = crt([x[i] for x in cifras], [x[0] for x in chaves_publicas]) # Teorema Chinês dos Restos
        res.append(x.nth_root(3)) # Raiz de grau 3 de x
    return "".join(map(chr, res))
```

Alternativa

Após a primeira abordagem tentamos otimizar a função de encritação, aplicando a tradução de cada caráter da mensagem para o valor $ASCII$ (https://www.asciitable.com) correspondente, de seguida agrupamo-los sequencialmente e por fim convertemo-los para inteiro. Esta intrepertação permitiu a transformação da mensagem com um todo.

Ilustração da abordagem: Suponhando que a mensagem a ser enviada é "OLA" fazemos a conversão para 079076065.

Porém quando fizemos alguns testes descobrimos uma limitação neste método que se dá quando o valor inteiro calculado é maior que o N mais pequeno das chaves públicas.

In [4]:
```
def RSA_encriptar2(mens, ch_pub):
    n, e = ch_pub
    plain = ""
    for ch in mens:
        x = ord(ch)
        plain += format(x, '03')
    cripto = power_mod(int(plain), e, n)
    return cripto
```

In [5]:
```
def convertToString(decif):
    decifString = str(decif)

    plain = []
    i = len(decifString)

    while i > 0:
        if i >= 3:
            plain.append(chr(int(decifString[i-3:i])))
        else:
            plain.append(chr(int(decifString[0:i])))
        i -= 3
    return "".join(plain[::-1])
```

In [6]:
```
def RSA_desencriptar2(cripto, ch_priv, ch_pub):
    n, e = ch_pub
    decif = power_mod(cripto, ch_priv, n)
    return convertToString(decif)
```

In [7]:
```
def hastad2(cifras, ch_pub):
    x = crt(cifras, [x[0] for x in ch_pub])
    return convertToString(x.nth_root(3))
```

Exemplos

Função Auxiliar:

In [8]:
```
def gerar_cifras(mensagem):
    chaves_publicas = []
    chaves_privadas = []
    cifras = []
    cifras2 = []
    for i in range(3):
        publica, privada = RSA()
        chaves_publicas.append(publica)
        chaves_privadas.append(privada)
        cifras.append(RSA_encriptar(mensagem, publica))
        cifras2.append(RSA_encriptar2(mensagem, publica))

    return chaves_publicas, chaves_privadas, cifras, cifras2
```

Exemplo 1

A Alice decidiu enviar o famoso texto $Lorem\ ipsum$ (https://pt.wikipedia.org/wiki/Lorem_ipsum) para a sua equipa de desenvolvimento web com o intuito destes utilizarem-no para testar e ajustar aspetos visuais no site. A partilha desta informação deu-se através de um sistema de RSA com $e = 3$. O Bob que também pertence a uma empresa de $webdesign$ concorrente estava a tentar obter informações sobre o conteúdo do site e conseguiu interceptar a mensagem através do ataque de $Hastad$.

In [9]:
```
plaintext = '''Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed vitae tortor in quam bibendum iacul

Nullam lacinia augue non ipsum sagittis, id lobortis massa vestibulum. Donec dignissim eleifend ante eu ultrice

Vivamus vel lacus ac diam sollicitudin rutrum sed at libero. Aliquam semper purus eros, vel placerat turpis ege

Proin lobortis quam in libero fermentum lacinia. Nunc varius ligula at erat convallis ultricies. Integer egesta

Nam in turpis fringilla, venenatis lorem eu, congue enim. Phasellus commodo, leo in placerat tempor, felis quam
```

In [10]:
```
chaves_publicas, chaves_privadas, cifras, cifras2 = gerar_cifras(plaintext)

mensO1 = hastad(cifras,chaves_publicas)

print("A mensagem interceptada pelo Zachary foi: \n\n'" + mensO1  +"'\n\nutilizando o 1º método.")
```

```
A mensagem interceptada pelo Zachary foi:

'Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed vitae tortor in quam bibendum iaculis non et e
lit. Suspendisse pellentesque nibh ut ex porta finibus. Ut ut sapien tempor, ullamcorper risus a, porta ex.
Duis arcu est, iaculis non mi eu, interdum condimentum nunc. Sed eu porta libero. Praesent fermentum mi eu a
uctor consequat. Morbi in lectus metus. Donec ex massa, fermentum ac bibendum a, porttitor a felis. Praesent
nisl nulla, dignissim sit amet ligula a, rutrum tincidunt sem.

Nullam lacinia augue non ipsum sagittis, id lobortis massa vestibulum. Donec dignissim eleifend ante eu ultr
ices. Duis luctus elit leo, laoreet maximus ipsum ornare ac. In hac habitasse platea dictumst. Integer turpi
s odio, venenatis in mattis venenatis, aliquet at tortor. Suspendisse molestie molestie arcu hendrerit imper
diet. Pellentesque ex est, vehicula vitae lobortis eget, laoreet sit amet enim. Aliquam posuere sagittis eni
m, a accumsan lectus. Suspendisse vulputate nibh ac mi pulvinar, non blandit neque varius. Sed rutrum arcu u
t turpis volutpat, a cursus neque ultrices. Nulla nec congue nisl. Duis in urna bibendum enim consectetur vi
verra in nec est. Cras malesuada metus a viverra lobortis. Aliquam eu imperdiet diam.

Vivamus vel lacus ac diam sollicitudin rutrum sed at libero. Aliquam semper purus eros, vel placerat turpis
egestas et. Cras vel dignissim quam. Pellentesque in tempus purus, a aliquam lacus. Sed a justo quis lectus
euismod tincidunt eu vitae dui. Ut luctus neque ac tellus dictum fermentum. Sed in diam eget velit ornare ve
hicula. Vivamus sit amet dignissim eros. Curabitur venenatis vehicula dolor et mattis. Cras convallis eros e
```

Neste exemplo não foi utilizado o método alternativo, pois após a conversão para inteiro, o seu valor é maior que o n_i mais pequeno.

Exemplo 2

Luís Filipe, famoso empresário português queria enviar um email para os seus colegas de "$negócios$" e foi-lhe sugerido o uso do sistema RSA com expoente $e = 3$. Sabendo as vulnerabilidades deste método, o hacker Rui Pinto, conseguiu desencriptar o email enviado, que colocou o Luís em grandes sarilhos.

In [11]:
```
email = 'Olá amigos José Silva, Júlio Loureiro e Paulo Gonçalves, seria importante para mim falar um pouco com
```

In [12]:
```
chaves_publicas, chaves_privadas, cifras, cifras2 = gerar_cifras(email)

mensO1 = hastad(cifras,chaves_publicas)

print("A mensagem enviada por Luís Filipe foi: \n\n'" + mensO1  +"'\n\nutilizando o 1º método.\n")
```

```
A mensagem enviada por Luís Filipe foi:

'Olá amigos José Silva, Júlio Loureiro e Paulo Gonçalves, seria importante para mim falar um pouco com vocês r
elativamente às camisolas, convites e bilhetes que prometi. A verdade é que estou aqui com um problema e preci
samos de nos reunir. Digam-me o melhor dia e hora, pois precisamos de falar algumas horas. Abraço amigo, Luí
s.'

utilizando o 1º método.
```

Neste exemplo não foi utilizado o método alternativo, pois após a conversão para inteiro, o seu valor é maior que o n_i mais pequeno.

Exemplo 3

O núcleo de estudantes de Engenharia Informática da Universidade Do Minho utiliza encriptação RSA com expoente $e = 3$ para partilhar dentro da organização as datas dos eventos que planeiam realizar. Ao descobrir isto, Bruno $Wicked$ decidiu interceptar a mensagem e informar o Núcleo de Estudantes de Ciências da Computação.

In [13]:
```
mensagem = 'JOIN - 28, 29, 30 junho'
```

In [14]:
```
chaves_publicas, chaves_privadas, cifras, cifras2 = gerar_cifras(mensagem)

mensO1 = hastad(cifras,chaves_publicas)
mensO2 = hastad2(cifras2,chaves_publicas)

print("A mensagem original interceptada pelo Bruno foi: \n\n'" + mensO1  +"'\n\nutilizando o 1º método.\n")
print("A mensagem original interceptada pelo Bruno foi: \n\n'" + mensO2  +"'\n\nutilizando o 2º método.")
```

```
A mensagem original interceptada pelo Bruno foi:

'JOIN - 28, 29, 30 junho'

utilizando o 1º método.

A mensagem original interceptada pelo Bruno foi:

'JOIN - 28, 29, 30 junho'

utilizando o 2º método.
```