Universidade Federal do Espírito Santo

CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA

Processamento Paralelo e Distribuído – Turmas 01 e 02 (EARTE) – 2022/1

Prof. Rodolfo da Silva Villaça – rodolfo.villaca@ufes.br

Laboratório V – Comunicação Indireta, Eleição e Coordenação Distribuída
Parte 2

1. Objetivos

- Experimentar a verificação de assinaturas de mensagens por meio de chaves públicas/ privadas;

2. Roteiro Básico

I. Instalação dos pré-requisitos de execução:
– Biblioteca pycryptodome 3.15.0
PyCryptodome

```
$ pip install -r requirements.txt
```

II. Geração das Chaves Pública/Privada (Public/Private Key):

```
$ bash generate_key.sh
Generating RSA private key, 1024 bit long modulus
...++++++
..++++++
e is 65537 (0x10001)

$python 0_export_public_key.py
(done)
```

III. Visualizar a Chave Pública:

```
cat public_key.txt
-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQC9wA27p3MJGj2uNLT8mpET4FjR
WS8pe2LTVx1owJmb0Q3WPKoDjVKG1EsluHXWLR72KvYsD1lr8S6XhFR5uLa8DY5J
vaBBY2XQrOqAvAbaNTPYebkj2346g3SnOCI25NnnItNv7xJKibo59ObgzbEr9Chk
n4HOp0FRi+P9ANDTawIDAQAB
-----END PUBLIC KEY-----
```

– Importante: a saída acima é aleatória e será diferente para cada geração de uma nova chave!

IV. Visualizar a Chave Privada:

```
$ cat private_key.pem
-----BEGIN RSA PRIVATE KEY-----
MIICXAIBAAKBgQC9wA27p3MJGj2uNLT8mpET4FjRWS8pe2LTVx1owJmb0Q3WPKoD
jVKG1EsluHXWLR72KvYsD1lr8S6XhFR5uLa8DY5JvaBBY2XQrOqAvAbaNTPYebkj
2346g3SnOCI25NnnItNv7xJKibo59ObgzbEr9Chkn4HOp0FRi+P9ANDTawIDAQAB
AoGAbPlNacYjNMkTP2cZwjDqvOWNXL3BbitkeEeBplVmYqxPLAivA1c4XCKz/bfQ
RO6o52uI8YjnLCzZOz9OXJGOr7d/mUtXRZpLOchbBQ2fVlatHizsyy3o77bMduND
ppJKELl3LRhLOnzLZKqHSSyHqwUblBU21WMbSzHG7zHue+ECQQDvOh1BhM8Ppngd
LWpnFv7wABb/umYES9QKQDKbL2YFUWVUDvDTUm0dypKq8YY/mdc532HbsZlhvWRO
gDpDywXpAkEAyw3hWPdSz7Hg0JD9TCIKDUhmVZYM0QGwhJN2gXrWDnJTqyeln3RE
J0SDrRJ6+Q/Byx86H/SlrQEhExijBfO2MwJAQN9Vh6roK6pM8DycmhAWWibsFbUK
```

```
bSfS/GSkVIjp9Y85Fy5mCChWBrokQ87zRNQHFV6KPvVOTr75MIB9TF0F4QJBALgO
9uv8FJar+MtWAiTW3iG4NPrzrt5uF6G1QcCQpiQmQe1tXNYzP+wQUQO7Mrk9LT+A
KOH6k4tE8MQsmum0Rr0CQAjY3egzQchUtYcjkcLgMIXiw5ASPUiM5K7UNfD6FvP7
ncpDLQplJopIIIQsHezlD96GjcbZz7awqms3vzuvdEs=
-----END RSA PRIVATE KEY-----
```

V. Criar uma assinatura para uma determinada chave:

```
$ python 1_sign.py
Enter a message: Rodolfo da Silva Villaca
Signature:
b1599cea8fd8ada2f133bd127a9b5c6dea1cedc4804d595b08df6bcd0ee15fca5c26b12c48e1ba83d38285980
5b5375bffe68c127a11b137d9ce8d20f18a13ec507f9177d23039f01ea96b95a8f16841105ee446f3bacc4c4a
1a97261c03a6168f879f4ace79ae42abd3c485f1dbb23ba2e683a7c3f7cf1c6f67204fbe955437
```

VI. Efetuar a validação da assinatura:

```
$ python 2_verify.py
Enter a message: Rodolfo da Silva Villaca
Enter Signature?
b1599cea8fd8ada2f133bd127a9b5c6dea1cedc4804d595b08df6bcd0ee15fca5c26b12c48e1ba83d38285980
5b5375bffe68c127a11b137d9ce8d20f18a13ec507f9177d23039f01ea96b95a8f16841105ee446f3bacc4c4a
1a97261c03a6168f879f4ace79ae42abd3c485f1dbb23ba2e683a7c3f7cf1c6f67204fbe955437
Successfully verified message
```

VII. Efetuar a validação da assinatura (com erro):
– Observe que a mensagem foi alterada para Rodolfo da Silva Villaca, com I maiúsculo.

```
$ python 2_verify.py
Enter a Message: Rodolfo da Silva VIllaca
Enter                                                                    Signature?
b1599cea8fd8ada2f133bd127a9b5c6dea1cedc4804d595b08df6bcd0ee15fca5c26b12c48e1ba83d38285980
5b5375bffe68c127a11b137d9ce8d20f18a13ec507f9177d23039f01ea96b95a8f16841105ee446f3bacc4c4a
1a97261c03a6168f879f4ace79ae42abd3c485f1dbb23ba2e683a7c3f7cf1c6f67204fbe955437
FAILED
```

3. Atividade

a) Criar 2 processos que se comunicam via uma infraestrutura Pub/Sub usando RabbitMQ (ou Mosquitto) por meio da fila “ppd/pubkey”. Esta fila deverá receber mensagens no formato:

```
NodeName:PubKey
```

Exemplo, se o nome do nó da rede Pub/Sub for `patolino` e a sua chave pública for `MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQC9wA27p3MJGj2uNLT8mpET4FjRWS8pe2LTVx1owJmb0Q3WPKoDj VKG1EsluHXWLR72KvYsD1lr8S6XhFR5uLa8DY5JvaBBY2XQrOqAvAbaNTPYebkj2346g3SnOCI25NnnItNv7xJKib o59ObgzbEr9Chkn4HOp0FRi+P9ANDTawIDAQAB` então a mensagem a ser enviada pelo nó na rede Pub/Sub será:

```
patolino:MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQC9wA27p3MJGj2uNLT8mpET4FjRWS8pe2LTVx1owJmb
0Q3WPKoDjVKG1EsluHXWLR72KvYsD1lr8S6XhFR5uLa8DY5JvaBBY2XQrOqAvAbaNTPYebkj2346g3SnOCI25NnnI
tNv7xJKibo59ObgzbEr9Chkn4HOp0FRi+P9ANDTawIDAQAB
```

b) Ao receber uma chave pública na fila “ppd/pubkey” o nó assinante deverá armazenar a chave pubkey do nó em um arquivo .txt no formato:

patolino.txt

```
-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQC9wA27p3MJGj2uNLT8mpET4FjR
WS8pe2LTVx1owJmb0Q3WPKoDjVKG1EsluHXWLR72KvYsD1lr8S6XhFR5uLa8DY5J
vaBBY2XQrOqAvAbaNTPYebkj2346g3SnOCI25NnnItNv7xJKibo59ObgzbEr9Chk
n4HOp0FRi+P9ANDTawIDAQAB
-----END PUBLIC KEY-----
```

– Observe que o nome do arquivo é o nome do node associado à chave pubkey!

c) Considerando os 2 nós na rede Pub/Sub, envie mensagens assinadas do nó1 para o nó2, e vice-versa, e sempre verifique a assinatura antes de aceitar a mensagem (exibir na tela). Lembrem-se: assinar com privkey e verificar com pubkey.

d) Crie um terceiro nó, nó3, que ***não*** possui a privkey do nó2, mas manda mensagens para o nó1 em nome do nó2 (tentando se passar por ele).

e) Faça com que o nó3 consiga acesso à chave privkey do nó2 e envie mensagens para o nó1 em nome do nó2 (tentando se passar por ele).

4. Instruções Gerais

1. O trabalho pode ser feito em grupos de 2 ou 3 alunos: não serão aceitos trabalhos individuais ou em grupos de mais de 3 alunos;
2. Os grupos deverão implementar os trabalhos usando Python;

Bom trabalho!