p-t.py.pdf

piedad_pg

Seguridad de la Información

3º Grado en Ingeniería Informática

Escuela Técnica Superior de Ingeniería Informática
Universidad de Málaga

```python
from Crypto.Hash import SHA256, HMAC
import base64
import json
import sys
from socket_class import SOCKET_SIMPLE_TCP
import funciones_aes

# Paso 0: Crea las claves que T comparte con B y A
#####################################################

# Crear Clave KAT, guardar a fichero
KAT = funciones_aes.crear_AESKey()
FAT = open("KAT.bin", "wb")
FAT.write(KAT)
FAT.close()

# Crear Clave KBT, guardar a fichero
KBT = funciones_aes.crear_AESKey()
FBT = open("KBT.bin", "wb")
FBT.write(KBT)
FBT.close()

# Paso 1) B->T: KBT(Bob, Nb) en AES-GCM
#######################################

# Crear el socket de escucha de Bob (5551)
print("Esperando a Bob...")
socket_Bob = SOCKET_SIMPLE_TCP('127.0.0.1', 5551)
socket_Bob.escuchar()

# Crea la respuesta para B y A: K1 y K2
K1 = funciones_aes.crear_AESKey()
K2 = funciones_aes.crear_AESKey()

# Recibe el mensaje
cifrado = socket_Bob.recibir()
cifrado_mac = socket_Bob.recibir()
cifrado_nonce = socket_Bob.recibir()

# Descifro los datos con AES GCM
datos_descifrado_ET = funciones_aes.descifrarAES_GCM(KBT, cifrado_nonce, cifrado,
cifrado_mac)

# Decodifica el contenido: Bob, Nb
json_ET = datos_descifrado_ET.decode("utf-8" ,"ignore")
print("B->T (descifrado): " + json_ET)
msg_ET = json.loads(json_ET)

# Extraigo el contenido
t_bob, t_nb = msg_ET
t_nb = bytearray.fromhex(t_nb)

# Paso 2) T->B: KBT(K1, K2, Nb) en AES-GCM
##########################################

mensaje=[]
mensaje.append(K1.hex())
mensaje.append(K2.hex())
mensaje.append(t_nb.hex())
jsonTB = json.dumps(mensaje)
```

```
60 print("T -> B (cifrado): " + jsonTB)
61
62 # Cifra los datos con AES GCM
63 aes_engine = funciones_aes.iniciarAES_GCM(KBT)
64 cifradoB, cifrado_macB, cifrado_nonceB =
   funciones_aes.cifrarAES_GCM(aes_engine,jsonTB.encode("utf-8"))
65
66 # Envia los datos
67 socket_Bob.enviar(cifradoB)
68 socket_Bob.enviar(cifrado_macB)
69 socket_Bob.enviar(cifrado_nonceB)
70
71
72
73 # (A realizar por el alumno/a...)
74
75 # Cerramos el socket entre B y T, no lo utilizaremos mas
76 socket_Bob.cerrar()
77
78 # Paso 3) A->T: KAT(Alice, Na) en AES-GCM
79 ##########################################
80 # Crear el socket de escucha de Bob (5551)
81 print("Esperando a Alice...")
82 socket_Alice = SOCKET_SIMPLE_TCP('127.0.0.1', 5551)
83 socket_Alice.escuchar()
84
85 # Crea la respuesta para B y A: K1 y K2
86 #ya la creamos antes, son la misma k1 y k2
87
88 # Recibe el mensaje
89 cifrado2 = socket_Alice.recibir()
90 cifrado_mac2 = socket_Alice.recibir()
91 cifrado_nonce2 = socket_Alice.recibir()
92
93 # Descifro los datos con AES GCM
94 datos_descifrado_AT = funciones_aes.descifrarAES_GCM(KAT, cifrado_nonce2, cifrado2,
   cifrado_mac2)
95
96 # Decodifica el contenido: Bob, Nb
97 json_AT = datos_descifrado_AT.decode("utf-8" ,"ignore")
98 print("A->T (descifrado): " + json_AT)
99 msg_AT = json.loads(json_AT)
100
101 # Extraigo el contenido
102 t_alice, t_na = msg_AT
103 t_na = bytearray.fromhex(t_na)
104
105 # (A realizar por el alumno/a...)
106
107 # Paso 4) T->A: KAT(K1, K2, Na) en AES-GCM
108 ##########################################
109 mensajeA=[]
110 mensajeA.append(K1.hex())
111 mensajeA.append(K2.hex())
112 mensajeA.append(t_na.hex())
113 jsonTA = json.dumps(mensajeA)
114 print("T -> A (cifrado): " + jsonTA)
115
116 # Cifra los datos con AES GCM
117 aes_engine = funciones_aes.iniciarAES_GCM(KAT)
```

```
118 cifradoA, cifrado_macA, cifrado_nonceA =
    funciones_aes.cifrarAES_GCM(aes_engine,jsonTA.encode("utf-8"))
119
120 # Envia los datos
121 socket_Alice.enviar(cifradoA)
122 socket_Alice.enviar(cifrado_macA)
123 socket_Alice.enviar(cifrado_nonceA)
124
125
126 # (A realizar por el alumno/a...)
127 # Cerramos el socket entre A y T, no lo utilizaremos mas
128 socket_Alice.cerrar()
```