p-a.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
from Crypto.Random import get_random_bytes

# Paso 0: Inicializacion
########################

# Lee clave KBT
KAT = open("KAT.bin", "rb").read()


# Paso 3) A->T: KAT(Alice, Na) en AES-GCM
########################################

# Crear el socket de conexion con T (5551)
print("Creando conexion con T...")
socket = SOCKET_SIMPLE_TCP('127.0.0.1', 5551)
socket.conectar()

# Crea los campos del mensaje
t_n_origen = get_random_bytes(16)

# Codifica el contenido (los campos binarios en una cadena) y contruyo el mensaje JSON
msg_TE = []
msg_TE.append("Alice")
msg_TE.append(t_n_origen.hex())
json_ET = json.dumps(msg_TE)
print("A -> T (descifrado): " + json_ET)

# Cifra los datos con AES GCM
aes_engine = funciones_aes.iniciarAES_GCM(KAT)
cifrado, cifrado_mac, cifrado_nonce = funciones_aes.cifrarAES_GCM(aes_engine,json_ET.encode("utf-8"))

# Envia los datos
socket.enviar(cifrado)
socket.enviar(cifrado_mac)
socket.enviar(cifrado_nonce)

# Paso 4) T->A: KAT(K1, K2, Na) en AES-GCM
########################################
cifradoA = socket.recibir()
cifrado_macA = socket.recibir()
cifrado_nonceA = socket.recibir()
datos_claro=funciones_aes.descifrarAES_GCM(KAT, cifrado_nonceA, cifradoA,cifrado_macA)

# Decodifica el contenido: Bob, Nb
json_AT = datos_claro.decode("utf-8" ,"ignore")
print("A->T (descifrado): " + json_AT)
msg_AT = json.loads(json_AT)

# Extraigo el contenido
K1,K2, t_nb = msg_AT
```

```
58 K1 = bytearray.fromhex(K1)
59 K2 = bytearray.fromhex(K2)
60 t_nb = bytearray.fromhex(t_nb)
61
62 if(t_nb==t_n_origen):
63     print("El nonce es el mismo")
64 else:
65     print("El nonce no es el mismo")
66     exit
67
68
69 # Cerramos el socket entre A y T, no lo utilizaremos mas
70 socket.cerrar()
71
72 # Paso 5) A->B: KAB(Nombre) en AES-CTR con HMAC
73 ################################################
74 print("Creando conexion con Bob...")
75 socket = SOCKET_SIMPLE_TCP('127.0.0.1', 5553)
76 socket.conectar()
77
78 #envio el nombre
79 nombre="Pepito"
80 aes_cifrado, nonce_16_ini = funciones_aes.iniciarAES_CTR_cifrado(K1)
81 datos_cifrado = funciones_aes.cifrarAES_CTR(aes_cifrado, nombre.encode("utf-8"))
82 #creo el hmac
83 hsend = HMAC.new(K2, msg=nombre.encode("utf-8"), digestmod=SHA256)
84 mac = hsend.digest()
85
86 mensaje = []
87 mensaje.append(datos_cifrado.hex())
88 mensaje.append(nonce_16_ini.hex())
89 mensaje.append(mac.hex())
90 json_paquete = json.dumps(mensaje)
91 socket.enviar(json_paquete.encode("utf-8"))
92
93
94
95
96 # Paso 6) B->A: KAB(Apellido) en AES-CTR con HMAC
97 ################################################
98
99 #recibe el apellido
100 paquete = socket.recibir()
101 # Decodifica el contenido:
102 json_BT = paquete.decode("utf-8" ,"ignore")
103 print("A->B (descifrado): " + json_BT)
104 msg_BT = json.loads(json_BT)
105
106 # Extraigo el contenido
107 datos_cifrado, nonce, mac = msg_BT
108 datos_cifrado= bytearray.fromhex(datos_cifrado)
109 nonce = bytearray.fromhex(nonce)
110
111 #lo descifro
112 aes_descifrado=funciones_aes.iniciarAES_CTR_descifrado(K1, nonce)
113 datos_claro=funciones_aes.descifrarAES_CTR(aes_descifrado, datos_cifrado)
114 mensaje_claro_json = datos_claro.decode("utf-8")
115 print("A -> B (descifrado): " + mensaje_claro_json)
116
117
```

```
118 hmacB = HMAC.new(K2, digestmod=SHA256)
119 hmacB.update(mensaje_claro_json .encode("utf-8"))
120 try:
121     hmacB.hexverify(mac)
122     print("Mensaje correcto")
123 except ValueError:
124     print("Mensaje manipulado")
125     socket.cerrar()
126     exit()
127
128
129
130
131
132 # Paso 7) A->B: KAB(END) en AES-CTR con HMAC
133 ############################################
134 #envio el nombre
135 end="END"
136 aes_cifrado, nonce_16_ini = funciones_aes.iniciarAES_CTR_cifrado(K1)
137 datos_cifrado = funciones_aes.cifrarAES_CTR(aes_cifrado, end.encode("utf-8"))
138 #creo el hmac
139 hsend = HMAC.new(K2, msg=nombre.encode("utf-8"), digestmod=SHA256)
140 mac = hsend.digest()
141
142 mensaje = []
143 mensaje.append(datos_cifrado.hex())
144 mensaje.append(nonce_16_ini.hex())
145 mensaje.append(mac.hex())
146 json_paquete = json.dumps(mensaje)
147 socket.enviar(json_paquete.encode("utf-8"))
148 socket.cerrar()
149 # (A realizar por el alumno/a...)
150
```