COSAS-IMPORTANTES-PARCIAL-PRACTI...

Fredis

Seguridad de la Información

3º Grado en Ingeniería Informática

Escuela Técnica Superior de Ingeniería Informática
Universidad de Málaga

-ECB->no hace falta IV
-CBC-OFB-CFB->IV
-CTR->Comprueba nonce
-GCM->NONCE,MAC

HMAC->
macc=HMAC.new(k1,msg.encode(),digestmod=SHA256)
mac.hexdigest()
socket.enviar(mac.encode())
macc=new
macc.hexverify(mac)

JSON
enviar
mensaje.append(n4.hex())
json.dumps(mensaje)
funcion_aes(jstr.encode())

recibir
msg=funcion
json.loads(msg)
nonce=bytearray.fromhex(noncea)

ENCRIPTAR Y DESENCRIPTAR
encrypt(pad(msg.encode(),BLOCK_SIZE_AES))
unpad(decrypt(msg),BLOCK_SIZE))

FIRMAR->RSA con kpriv y se des con kpublic

```
def firmarRSA_PSS(texto, key_private):
  # La firma se realiza sobre el hash del texto (h)
  h = SHA256.new(texto.encode("utf-8"))
  print(h.hexdigest())
  signature = pss.new(key_private).sign(h)
return signature

def comprobarRSA_PSS(texto, firma, key_public):
 # Comprobamos que la firma coincide con el hash (h)
 h = SHA256.new(texto.encode("utf-8"))
 print(h.hexdigest())
 verifier = pss.new(key_public)
try:
 verifier.verify(h, firma)
 return True
except (ValueError, TypeError):
 return False
```