CYBER SECURITY

4. forduló

Continental The Future in Motion

A kategória támogatója: Continental Automotive Hungary Kft.

Ismertető a feladathoz

A 4. forduló után elérhetőek lesznek a helyezések %-os formában: azaz kiderül, hogy a kategóriában a versenyzők TOP 20% - 40% -60% -ához tartozol-e!

Szeretnénk rá felhívni figyelmedet, hogy a játék nem Forma-1-es verseny! Ha a gyorsaságod miatt kilököd a rendesen haladó versenyzőket, kizárást vonhat maga után!

4.forduló

A különböző kriptográfiai aláírások használata nagy mértékben elterjedt a modern járműiparban is. A 4. fordulóban egy kriptografikus módon aláírt üzenet segítségével fogjuk ezt a tudást próbára tenni.

A forduló megoldásához erősen ajánlott egy kriptográfiai eszköztár használata (pl. Python környezet PyCryptodome libraryvel).

Felhasznált idő: 00:00/25:00 **Elért pontszám:** 0/15

Indítás utáni csatolmányok

1. feladat 0/15 pont

Adott a következő mondat (a másolhatóság miatt jelenítjük meg kódként):

```
Continental - The Future in Motion
```

Az esetleges új sor / sortörés már nem része a stringnek!

Add meg azt a RSA PSS aláírást, amelyik a fenti jelmondatra lett generálva!

Hash-ként SHA256-os funkció volt használva. Az RSA kulcs hossza 2048 bit. A kulcspár PEM formátumban, illetve a 4 aláírás a csatolmányban található.

Válasz

- (●) option_1_sign.txt
  Ez a válasz helyes, de nem jelölted meg.
- ( ) option_2_sign.txt

option_3_sign.txt

option_4_sign.txt

Magyarázat

A feladat megoldása során a „signiture verification” kriptográfiai módszert kell használni.

Elegendő a publikus kulcs használata a feladat megoldásához.

Lehetséges megoldás Python programnyelven.

```python
from Crypto.PublicKey import RSA
from Crypto.Signature import pss
from Crypto.Hash import SHA256


public_key = RSA.importKey(open('public_key.pem').read())


message = b'Continental - The Future in Motion'

signature = bytes.fromhex(("35801A55F73577393F504E0A79FA19BE2148FA2E479FF1A22B8E082244A51B56E9F5013EBDCE8BFE904C693C5EE6AC55B0ABF3A5A8295BB7986CB9D3C68

#verify signiture
h = SHA256.new(message)
verifier = pss.new(public_key)
try:
    verifier.verify(h, signature)
    print("The signature is authentic.")
except (ValueError, TypeError):
    print("The signature is not authentic.")
```