Task 1 Alice is using the following "textbook" RSA-3072 digital signature scheme: sig = (msg) d mod N for message authentication. Given the public modulus N:

1(a) Given Alice's private key d and the message msg, compute Alice's signature sig:

Using the RSA-3072 digital signature scheme: sig (msg)^(d)mod N, by inputting the public modulus N, Alice's private key d, the message msg, we can calculate Alice's signature.

1. Define variables N, d, msg
2. Write the equation for sig=power_mod(msg,d,N)
3. Evaluate sig=power_mod(msg,d,N)
4. Print sig, giving Alice's signature print(hex(sig))

sage:
N=0xe55c85be1e8f31b8cfa79da46e313545fe58d51308f427be1798373cce2304c0cee692f4
ab78387dc5d3161b5a1f33df90858c5c0a8fe906579257043a527f33e37b3466b7929be81abe
c6e9979215abf92d71032caf5fffe4a5f1c176172d8fb62da7beecc255e45b75a44e30ebbeb91
ecb97de7dc51a0c1d19f1cb0e5658b4a66cd4500252dc8f50076c357f5dece3f94ef1133cd2c5
92a5c9eb22a2e818f95252f0917caf47737807ece3a0f508f1af03b8eabd2f3d6cc881b27627e
3cb5eda7862c25213592ebf1f8470dff22d7603d299ee69628101c75133d65618692aad5f3b2f
fb3a22e1084a900cb0543107b02f8062737181eab4870cf25f0ed473cf4095530702314dd0a8
cace3a6fd0169f2dfea254d3ab152381c3ae535f780a1b532fe040eae7ba864bf28543a6dec71
1e62878ec4471341c8ee00824e9cae7627c29de36f3678cbfe046dce37bd6c7639c51f9387e1
b756bda7622efb9ee49fb258266b19fb359ef3f959ffabb0ebf3747bb923cf69899bfdcaba18bd
4dbb7

sage: N

sage:
d=0x541af9701e04a45700ce962015c835a0d503fe1e5cca2b48a99e47a32473f2ea40f48c2e
ec31c98555657255d5565bcf3f4fb98886d6febc34a0950817dae88a3e808f569b3a47b1751d
4013a861095166ae2322e6dfe8740d844c8284ab3b29d7c4261efcf2c64c56bd6ce2bf4db342
6ee879683cf669f6c7351c55398cb03a8e4c9a0e3ccbe5d527a3912a8cea045414b7bdef2ffe9
a348c56dec274ba676e05a224553543910fe6940169f73be36bbca1c0cd53525f53e4b2aa9e
69423ef077b2d1bfe8d45927a677f74418240b95ed5c698e62fb429ece5fabbdcff8f64c480bff4
6bb6a448ae350739795abd156a5814378248b7100bfbaa07b039bc105a32a6fe74e0768857
7edecd515bd452a41cbfc017b9d26e76a5bec2ce433714a02f0f2c3784b65738adc849c3c31f
8a731132e4bd8c2b2c0b33de87403c2b7ff12ab3d9582453844b4ff03142f899b256e407c330
1adc46794d14bd668beac877e9cb5aa0602c447b75d3424d3a71495ed55a86fb1b01b5fbae2
a766f6172301

sage: d

sage:
msg=0x6b948843b86adb04a834cba6a76d5753da8ffbdcd01782a49d395f52f4c37a0cc39ee
b41646ebc2b2003bacb203328e210604f248e02fa95aa6eec50751abe267f5c0b70f60901a4f
d338f61bb2000acb3f2cf80d602acf85c5ee2f015667e9520e2d5c1aa84dcc69c9358a376846d
2a0e9b52877fe17a76ce4bf6c46c7a46f61102d42869e0a594c4ad71a699a603654e4d6bdf83
fc09b9741b70e82013302517efceebc9be49a7bc86ab89653f3281ffcc20824970410461510f4

a9b538f8d5468b872cbef23a348b61576ae1f840138f14e7f8f13643aae1467cd534803555f8b
2facb34fae15d53dc8c954bc8af0561597bfbb5a82c3b08bc83d349962aaaa6e164a138045b9
6dd9730aa7e1bb440838c42296ff2bdf53ca69f09c7e74c5e855455ffb052399e82e7e182d8ef
a08c96bdd166a00381d3fc53bb2a3d46b0aa6e2af8a45cd00e8bffe34fb7bafd20dade1efece7
331b417136e2ed971c8f16e193948f3c6595e9f63a948610f1d3e2246e6603d0b039f9bdd50fc
50baadef

Solution:

Alice's Signature:

274e84c26582b994d2fda202d6774579e9124d4f07e0620c35df8f499fdf9ed1736703d0d613bb805ec1
030f6a9130fbdcf5a0f2dacf23bd9382b8112309b0fbccae7115d4a68d4660f2b86758b8b16a0affa96ac9
776934d4cf55d9ad64bf3a4b3adee1cc0633401319ac77d76cf6b79835b1c831a787b4e1f78d615b6a5
0856f15d08dcd94215a99644b822437ba1bb27dc4ef5e3d30e661bed7c598159854683ded1da06939e
ce2239daf6d3207d8f093a10210c12e3c405f3f7d4d89e2fca7c9d8b2272ddd6eaba0ff8f553f9e0a3440e
82db55bc053727474f0701e45da59a5ef02f5307e9ae8ade39862e8ccdcc14d3107485bcf5c7bb01600f
f8932f126561a474c41fc23d9af4651f2c8415d19a2514c37f37b360243149573afabdcc4aa1db33f9346
e89006d5577dcbb1ddd31db28b89f48c4f8f4bb0c61bb66b9c2998a1b7229f372e1e2e3d14905f1c6f9e
37dd0acbabe1fad962fd2d12d162a8aeb9879702984ecdbf4978a5bfdf2b6204eeb077f0dab98d1e815e
dd26a66f49

1(b) Given Alice's public key (e, N) where N is given in (a) and e =0x10001 in hexadecimal, and the following two message/signature pairs (msg1,sig1) and (msg2,sig2) Bob received from Alice, show how Bob verifies the digital signatures. Do both messages pass the verification?

To verify the signature we use the same decryption algorithm to decrypt the signature and compare the decrypted signature to the msg. Therefore for a message, we must decrypt signature appended to it and compare with message, if they match then msg was sent by the Alice.

Sage

1. Define variables N, msg1, msg2, sig1, sig2, e
2. To verify msg1, we decrypt sig1 call this msg1_verify
3. Compare using if msg1 == message1_verifying: print "verified" Else: print "not verified"

4) Repeat for msg2 and sig 2

sage:

msg1=0x9cca26cb7a1713c2c95ee703089b84bd27311c5750c2a817586e7b1fed6e12a8051
a4626b48656a229eec292e30f0751d59ce1544300919801303cef1a08d08015ba5ea047e434
8b340aca99b3fcbcde9663b1a8d59eed23b6e1e834889729e53a691343f589babff5c8f8fe661
bd0a643644dde1ba9f37023bb01af97dd12bfbc5517a2a1a3e67108f0b287e7dbe9c9d81fdc5
c1c7875c8577c6acd7eeaee2a46a5ce3c2d5123b085cdb554c36fa3a2f756f3e4515a0b0bb5a
a504ebcdf8d9e8837b2d3b8b60eea5658be5dc9f3f9cddf1449b226d668591144a7bc4f17bf4e
51a56ea29b0f6d7321054f208e965b022a0668e2563a058626ea7347d9fb776e5596198fc99
1b1dd450b0e621ab9b11e10a0ab5bdb9f572f8f4b82f9edb5b17d154371118d3be51b28d085
42c2d7ff5e99674a070d0c08fcb55f8cfdc8a3739194b0d2ed99a34bbc70b0c43dc709be2bafc
e3255a6c747461b5bd24798160549d3ea9b37691039d4d5482640bcf3297cce75435867076
1f22876c78a4ac69ead

Cyber Security Project

sage:
sig1=0x2be8a53416b161b0b7e76e28046a7a8b923417536fc6b27f2cb1666daaf6ab0292b69
6d9946134f3de1e4a9db1a0d47c5c888b0699bc29bf497dbcc49bd643bf151a06514d45f418a
ca2198a6220d56970d7c15bc65caaac568148fa03f1d39d8cbf3b6e919e8ed3ff25655f15e18e
0b89109f8f337dcb80853dd22d73461dd3956e83e97debf6a5878f1219682cac4d2a71a2fc66
2743221d2faba4fe8ab2f02055bbc895c38475c95590c9b08beeec49217bbc52a3771012022
a9c537477892b1eeadf17ef5f9471d1d40abd097ca9026fa321df01add8289bf611bd4c029eca
b9c0539a22b9360f241950bcb9b32dc339ca9c94054c0be4e1772629b7344a544c0eb56689
5d64ee773d220e417f9805bb36457515474146a22264c05c1a8f1a23c156c87111d4198dbee
9e89fc14e059e0e9030c5d1ce2327ebbe99099d006c99e5d0cfb9ea2cd94b944c8933277e53
f63a0e0664c9a652711ac0438bab41c6107924afaa77a5953c02e235a18f320822cdbb2621e
a0912631d55aefd1a4d

sage:
msg2=0x31c276bb243008ad8ee81ee029e80aad7e9ff16ed54dafe20649756f6bae1b57fb095
865c4a902d55892d0d22e3ba2ca62d7bc3a069a18f9c8df8e5b09a640ab1dd35bac240e6fbe
c27d7089abdcf943d3894cbe1e13a2db39dae0a1409259b76ead144ec7a8c308c4bd1ca1cc8
133de63c46d8092510ffa422bf8827d81e377bf70a07e6e82ed8b863cdc2aa6705731237f79f3
6aa6c35ca3f03542f0a7d5c56e3711b96b20c7bbb0da837c6cc3abec24783d2b95de9e6bc05
2b81d21955912d40a18b03ba9fcd206c37ccac389524b4ef4835822ea0cb3524ecd1a47ca2b
aa4bc66fd3dc4ba174aab59088019f9932102709519f8146d3d5af858077f351c97d277aaacf9
a832e0b4271475bf5fd29e380149e2bbd443f0e0363f7e96d2a3f02e384a01caf6b11afde5516
96f411f26e603225fde420deec3f4f715abb5e445180d2717870b2285f761f0baa91775151244
2ddbfd05e529a15b649f6d45aa93fb31626ab9a498d98612e225140c98551d0851057c33236
6e60f39234e0c711 sage:
sig2=0x79c8b72ca72f4c6363b3e29c1a6267ede2a1740dde90a071b8600f98f14eeee19b580
213a872c00fb5146a851285945bda4728437622ab9e0f800881a7ebaf71cfcb558c8de0a150e
1452443808614f0e96dbbeedfabcdbf01e41b4b9601935bf9f12c5947d7a066c236d6843bcf05
d136e1cd480eda39a40f3fa9e5a1b26033643859ad5b5bc91b185bd980d2efa223c5025c133
89e542167999282c8cb5aa180cfa89746f377bb3b2923bde3be1b6fa05980b6a80a9b52136d
d3b933dcd54a095dacb8d1c9fa7c8dbf96e421ae713440bba2f3c82c31c356a268d3623e7c15
10a8a6ca506943f843682c73179eaa35a9678ad599b2a41881a2b47234deb25640771b9ee8
ca4f488b21d735bf3adc1ae37a786dbc0622d7ba31a218d02567355af578b187eeef9de6b37f
eb408f3ac296d5410c9d4d2920f452e30cf215227075756ff2f4fb0a15741102c31c1e5966276
7d78691bf864f3fbda722da50b8e02e88f7b6029eb85da47c44439b90e9cd4d027d171960b9
438d9c9c73d211e555

sage: e = 0x10001

sage: N =
0xe55c85be1e8f31b8cfa79da46e313545fe58d51308f427be1798373cce2304c0cee692f4ab
78387dc5d3161b5a1f33df90858c5c0a8fe906579257043a527f33e37b3466b7929be81abec6
e9979215abf92d71032caf5fffe4a5f1c176172d8fb62da7beecc255e45b75a44e30ebbeb91ec
b97de7dc51a0c1d19f1cb0e5658b4a66cd4500252dc8f50076c357f5dece3f94ef1133cd2c592
a5c9eb22a2e818f95252f0917caf47737807ece3a0f508f1af03b8eabd2f3d6cc881b27627e3c
b5eda7862c25213592ebf1f8470dff22d7603d299ee69628101c75133d65618692aad5f3b2ffb
3a22e1084a900cb0543107b02f8062737181eab4870cf25f0ed473cf4095530702314dd0a8ca
ce3a6fd0169f2dfea254d3ab152381c3ae535f780a1b532fe040eae7ba864bf28543a6dec711e
62878ec4471341c8ee00824e9cae7627c29de36f3678cbfe046dce37bd6c7639c51f9387e1b7

56bda7622efb9ee49fb258266b19fb359ef3f959ffabb0ebf3747bb923cf69899bfdcaba18bd4dbb7

Solution:
Msg1, sig1: Verified
Msg2, sig2: Not verified ----------

1(c) Assume the attacker Marvin intercepted the following two message/signature pairs (msg1,sig1) and (msg2,sig2) signed by Alice using the same private key (d, N) from (a). Explain how Marvin can perform an efficient forgery attack that, given (msg1,sig1), (msg2,sig2) and Alice's public key (N, e), computes a forgery message/signature pair (msg3,sig3), where sig3 is a valid signature for the new message msg3 (here, new means that msg3 is not equal to either msg1 or msg2). NOTE: Marvin's forgery method CANNOT use Alice's secret key d, which is assumed to be unknown to Marvin. Show a numerical example that the signatures forged by Marvin can pass the verification.

Marvin can create a message by multiply msg1 and msg 2 together, and the signature appended to the message would be created by multiplying the sig1 and sig2.
Sage:

1. Define variables msg1, sig1, msg2, sig2
2. msg3=msg1*msg2
3. sig3=sig1*sig2
4. m3=power_mod(sig3,e,N)
5. hex(m3)
6. if msg3 == m3: print "verified" Else: print "not verified"

Even though Marvin doesn't know the digital scheme algorithm key, his messages can still be verified using the messages he intercepted from Alice and without the need of Alice's secret.

sage:
msg1=0xa9edfdd28f7b79039fc041e8244d3d06bbac0e2cd108e1c5fd5691ae03545cf27a465bee1216e6f97f5d2443767a32a6dcb46f012aa0ac19cf5e6b81097e2846cea4eab599620fc876ef6071d92b67cec573d91366301a60f90efab964ccfbda9b5fc197ba86bdfb9e7a5380f7e28c90e7149a3d8ca5d443e22df5e284cbb5700c2f89df1c6a7d21dca87a856523bf1a37e44e85e76a03be641ff9366b6aa8552bee1763bca3cdba155899137fb8fae54fa4558a9cfcb5dfabcd1473068b93ca sage:
sig1=0x75b4bfc2420836f896bde21d195204a875867d43765194babbf67a3b9803515c7c1779ecf9ab20266071493a3b2e12272e7ae1a1f030055c51eb1076814a7bdde56b9381644e892a0a32d5b176af4428db1bddc53df9b28e0d7949c660559672fe497f051c978f407bbb961bfed8841e5bd46d523010355535d01246da0d821a9389537e1747f8c296dff83ba22bbd5993300c92846ea288aade9fb0591c3bb3dd25372c15224a5ac3588734144190fce710a2e07493cc6bb0ad80f205667a4264ab0d1b139b8ac8fc2f35a89ec2f9c6c159f683ef111796eb8d6b0d02d73f5cc890af958c9ea7fc9ada016bb53c80191babb37facd0ead5093a969cc5947775f48b5b80797533d1cd40987eb537d63221c51dc87863d5ba6fe30fea2d1831fda5334e16b6e2c6db117d7eb5bcc918f6718213f6902fce55b498ca1f381fba98f49c6858c65168de2416cb8408f6838d12e5c99c6d1e0aea0e6c3dcbda5075dea991ac42c759b835e9bd4ff303eab6702e798ed284ebeda6e50754828cad511b sage:
msg2=0x4eaacda480337afeea561e82087deb98a9b16e84a7fdb9f6c586e37e04f74c42a891d2dbb397154a0e76b7df72c0975702965a1ca2a70bad04a7285b0a5618ef9bca9070f76c1930225f56f58d17b15b8954a35e08223f505c9c2e93cf8bf7f28c50647d385863dc36bd9e52556c

896e89e9073015a7c38a59f2914124701451844450a2e4c792b70e99bc730f82005154f7e6c
79b4ef394aaa155c524c18da44ea8615ca0cce07aa822c0ec6704902bef72db21f3302cfccf3a
b79458c1f197 sage:
sig2=0x77dd1b15bcf1cf5dc65d64408a18771a7cfb2af1833de20d7802ddd85d2c09429f52bf9
e1c2700a39f22f2d98933e22fc57876f30a1d86696a6970fae76d89e5556046fe1f91a0701571
4fc3d302e326bbe364f5784ffb0a6c68f33a693a758e442e1e2410731d2f6e77776e0435f5d80
6c8b149aef99fcc8f49620b487a6703178839d3658b4c5008e2db841383a9aceb0862aec852f
6af50926f9045afff5e40ab52929993e3b79bca23777f06ea2914a284b90254798d131dfd0e8b
97d0c94bfa589ed9a75a4d4ea045fecf7325eb6e91ca6536113d6acbe6e83a1dfac9833edafb
8d2d25439a8eb0ad6cec289d8628cc0733ca231409065c3215e3ad5e2f5ad61756e6d4bd5b
14fffb78e9d1f470057061f0b9ab988f3be141435a8a71f70f7084a20f6ce13ac2aceba4546afc6
fd28b305dbde81c7b54be192c69f118ce5d131b5f217e2ec804c1e11578cbff51270a4fd16632
2dd69aa48a18466f608c2621baf828002d55c8361cbb1f7b3279c986bdf1c3da2c12fe565e7c
15d7250 sage: e =
0x10001 sage:
N=0xe55c85be1e8f31b8cfa79da46e313545fe58d51308f427be1798373cce2304c0cee692f4
ab78387dc5d3161b5a1f33df90858c5c0a8fe906579257043a527f33e37b3466b7929be81abe
c6e9979215abf92d71032caf5fffe4a5f1c176172d8fb62da7beecc255e45b75a44e30ebbeb91
ecb97de7dc51a0c1d19f1cb0e5658b4a66cd4500252dc8f50076c357f5dece3f94ef1133cd2c5
92a5c9eb22a2e818f95252f0917caf47737807ece3a0f508f1af03b8eabd2f3d6cc881b27627e
3cb5eda7862c25213592ebf1f8470dff22d7603d299ee69628101c75133d65618692aad5f3b2f
fb3a22e1084a900cb0543107b02f8062737181eab4870cf25f0ed473cf4095530702314dd0a8
cace3a6fd0169f2dfea254d3ab152381c3ae535f780a1b532fe040eae7ba864bf28543a6dec71
1e62878ec4471341c8ee00824e9cae7627c29de36f3678cbfe046dce37bd6c7639c51f9387e1
b756bda7622efb9ee49fb258266b19fb359ef3f959ffabb0ebf3747bb923cf69899bfdcaba18bd
4dbb7

Solution:

In this task, id Marvin wants to perform an efficient forgery attack, he can create a new (**msg3,sig3**) by just getting the two pair of (msg1,sig1) and (msg2,sig2), and then by multiplying the **msg1 * msg2** create a new **msg3,** In addition, by multiplying **sig1*sig2** create a new **sig3.**

So after doing it, by calculating the we will get the exact message 3, which will show the verification.

Msg3 (msg1 * msg2 à)

0x3437e3e17b5c499ebb6b4c9bdb0f2902a0ee7d5df7320f93b8f1301d155893d3880558c737
49d1cc4a9b7d2b4af6fe6df1d2bfcaa74407a98ca4ac159a5e7ba24d8a1383dad89927e8aaa4
8dbe341cecb376a3b07e4cbda5bbbe6a089db1b04cc0e52caa377cb1843d76f38f3b601da69
915d859f897d0e5db156968647c93e251abe1bde110be28703dd8fd9d74652641b2ff540f648
9f23aa291250a9611313d387552b1c50b170131f9cb81a7f8f624f813bf1eee53c491e6b36709
8663ff549872ce798fa030e29523545a70e9c60ac0d2a3f55bf6a7a844fb5979da8c33df06dda
2b7e1ced2aa0c3361a6f9be655e9e852491c0c468bf9df785913cf37aa25eb16164b87099b00
706f9ecd92cb2d21052fa09d83da18f17aa282b7a6a5bbc5d6dbbb175a53df2e2e296c5667e7
eb40ea8a5d690af708b73bff4ebd2ab22f3b7c05097f0f056db524334a2adfbd63c81c83e0167
484958c98cc29d5f2ef69002652e40da1aeda5f8a7e11716e842522c6d8df42a7373bc4a19d2
1dbf5626

Sig3 (sig1 * sig2 à)

0x371caea0e651f5debe6fcc753eb3c7f87045c3caef40fc7fe7ea5da21e7c6874b95f725237796250770
60e7c1e10c749eda7ac14c998a76da66f8d74e813c5827483fe08ea474b198953aae94efd9d89d0d647
8653ab4e23e5313cbec593da3b305b8a26df1b14c90c8e0bc039f31621736a6aacdc7ec19db850954ac
27df4f465cbeabae0b99939fa361b6e7aeb70e18f0cd39dfb44ca17138fd34592880665fd7be72c82ee2
fd06fed5a19a77204c0bd5860af9e55a4ea67aa9e6c29588163dead8e01294c5b9344fc79f65b85d5e8f
b3184fae3292b4be1f74fe9b13c84d6ff68319efcc876c455dea3f62ef32f5310678db5b2e24436e7c037
ec6688791a61f3aecc0f98e243879440746bdf7c2446afdf98060e0e5b7d2979fc82b0c450f1d8fe1bd5
6
61969623805015702f9054e90151a073f7e6031a300eaa1b975aa5d80b43da96fe593386b244b9d1a6
397a58ea33e1613979b87de3951da94d845e2bd758d97bc1e16cb480725940f60c6baab8e8d485c0d
d275ba1572b64a69e6f99c425707fbb070a044b2d63dd3ec038f1f6acef536560eea22166b65d71f07fe
60f3fff4e494695c5d2eea344bddba353e4029c50d57775c294d38717bab621520d4d3cc22370928ed1
06493fa07ce9fcf2da10f4c35b3c99337e1315eee75df0107fb6bc51f7a8a9ae07db141aac6a50c7aabfc
45e29084820f1e7780fa5b79cc6f5547b3c3f6278d04f1e5afb108beea4838eac79dd1b771ebb715191
3fd13256d588caebb96ceb34579df61ddcf3911ec19511fc9c7794092c423d14fbd5e527f30aab50cf9a
92d157afc025fedae8e89c808c7d44e4b91d66d44fa51d524cb7c678a1ee536aa69ba92869f31ce7e5e
40308c174e509ac165f993d3b078579a38fe394f35654f1340f4a0c7faa589d523348e0b6dfcfbb8948bf
8c1355e4ca5108249cef4316cd616a680ab90d1b55a613d720d097ab5adb42d58c1937ba59417e7b1d
d1bc69039c6e103a8433afb6e7b1e09accd3fa5d3a157df3652886eeee27462935df004b35f921525b5f
d0abc803e3d6f81ac8f326e9f6165e70

1(d) Explain how Alice can modify the digital signature scheme to prevent the forgery attack in (c) without changing the keys from (a). Show the new signature of the same message msg from (a) signed by her private key (d, N) using the modified signature scheme.

To prevent the forgery attack, Alice can modify her digital signature scheme by using a hash function. In this task Alice will have to generate a key hash function for the given message, we used SHA256 key hash function. The hashed message would now be signed by Alice using her same secret key to create her new unforgeable signature.

1. Importing the hash library: port hashlib
2. h_sha256=hashlib.sha256(msg.str())
3. hashed_msg=int(h_sha256.hexdigest(),16)
4. print(hex(hashed_msg))
5. hashed_sig=power_mod(hashed_msg,d,N)
6. signed_hashed_message=(hex(hashed_sig))
7. signed_hashed_message= "0x"+signed_hashed_message
8. print(signed_hashed_message)

sig=
'43926d86701391467452eb77437dbc0b24cbe2b3b57fbb3abdcef0fb930418ad10295fff49368f564fec5
28814474843e5c5c8296a70403725be85afb6bc4f19560262a08abf9c5afd03b99ded9dd0c708ba6557
8874cf87c3448a441e299b931e5355248c82f7a0ce87afdd8683f0b7245385592fb98175ea42054be64b
49d3d4379e661c57e5cdd218fb67927a58dc94a569392d893d07dedf8c6334c30a7089f0c94f529c7ae
be99fe77402389b526a15aec8f9deaa29c08203cf3ad186b9d2a2907a4db0ec5d93fe1ad5ed8e58d501
6a94471e91308d93cd5fc77c4e4653ec907895b8864ea9499693bad2fba14fbeee7425022a702b128d3
55ad51904f51cbc45757ddcba78c126eaf84b79bdd17305e4bbfa1c69561ef2baf59e1f9fd4cf16033239f

cd3f0e11fbfa73841b1503f3efc9261a343dab2753b3b5cc0fac6d76731c988b020251856eb2d05c97da
2cb3a4f2818f9b5406319eb9e60d6f9f342d4908e0e757675330c7fa5410540abfac3e534b00c3fa004ba
a49d7760d9b4'

import hashlib sage:
N=0xe55c85be1e8f31b8cfa79da46e313545fe58d51308f427be1798373cce2304c0cee692f4
ab78387dc5d3161b5a1f33df90858c5c0a8fe906579257043a527f33e37b3466b7929be81abe
c6e9979215abf92d71032caf5fffe4a5f1c176172d8fb62da7beecc255e45b75a44e30ebbeb91
ecb97de7dc51a0c1d19f1cb0e5658b4a66cd4500252dc8f50076c357f5dece3f94ef1133cd2c5
92a5c9eb22a2e818f95252f0917caf47737807ece3a0f508f1af03b8eabd2f3d6cc881b27627e
3cb5eda7862c25213592ebf1f8470dff22d7603d299ee69628101c75133d65618692aad5f3b2f
fb3a22e1084a900cb0543107b02f8062737181eab4870cf25f0ed473cf4095530702314dd0a8
cace3a6fd0169f2dfea254d3ab152381c3ae535f780a1b532fe040eae7ba864bf28543a6dec71
1e62878ec4471341c8ee00824e9cae7627c29de36f3678cbfe046dce37bd6c7639c51f9387e1
b756bda7622efb9ee49fb258266b19fb359ef3f959ffabb0ebf3747bb923cf69899bfdcaba18bd
4dbb7

sage: N

sage:
d=0x541af9701e04a45700ce962015c835a0d503fe1e5cca2b48a99e47a32473f2ea40f48c2e
ec31c98555657255d5565bcf3f4fb98886d6febc34a0950817dae88a3e808f569b3a47b1751d
4013a861095166ae2322e6dfe8740d844c8284ab3b29d7c4261efcf2c64c56bd6ce2bf4db342
6ee879683cf669f6c7351c55398cb03a8e4c9a0e3ccbe5d527a3912a8cea045414b7bdef2ffe9
a348c56dec274ba676e05a224553543910fe6940169f73be36bbca1c0cd53525f53e4b2aa9e
69423ef077b2d1bfe8d45927a677f74418240b95ed5c698e62fb429ece5fabbdcff8f64c480bff4
6bb6a448ae350739795abd156a5814378248b7100bfbaa07b039bc105a32a6fe74e0768857
7edecd515bd452a41cbfc017b9d26e76a5bec2ce433714a02f0f2c3784b65738adc849c3c31f
8a731132e4bd8c2b2c0b33de87403c2b7ff12ab3d9582453844b4ff03142f899b256e407c330
1adc46794d14bd668beac877e9cb5aa0602c447b75d3424d3a71495ed55a86fb1b01b5fbae2
a766f6172301

sage: d

sage:
msg=0x6b948843b86adb04a834cba6a76d5753da8ffbdcd01782a49d395f52f4c37a0cc39ee
b41646ebc2b2003bacb203328e210604f248e02fa95aa6eec50751abe267f5c0b70f60901a4f
d338f61bb2000acb3f2cf80d602acf85c5ee2f015667e9520e2d5c1aa84dcc69c9358a376846d
2a0e9b52877fe17a76ce4bf6c46c7a46f61102d42869e0a594c4ad71a699a603654e4d6bdf83
fc09b9741b70e82013302517efceebc9be49a7bc86ab89653f3281ffcc20824970410461510f4
a9b538f8d5468b872cbef23a348b61576ae1f840138f14e7f8f13643aae1467cd534803555f8b
2facb34fae15d53dc8c954bc8af0561597bfbb5a82c3b08bc83d349962aaaa6e164a138045b9
6dd9730aa7e1bb440838c42296ff2bdf53ca69f09c7e74c5e855455ffb052399e82e7e182d8ef
a08c96bdd166a00381d3fc53bb2a3d46b0aa6e2af8a45cd00e8bffe34fb7bafd20dade1efece7
331b417136e2ed971c8f16e193948f3c6595e9f63a948610f1d3e2246e6603d0b039f9bdd50fc
50baadef

Solution:

Cyber Security Project

E016d9466db6ce33bd024073050c95b16f15419496b08c2da2174b0031dd01e946c554f5421f3818ff3
d17a7701214de7776cbd52e6d3dacbd10719d4811ef8ae5ae4d8316acbd8449b892cee259f26164769
7ed847e169ce24efec0a764057072f391a29556232d1f00561e9af16a8c797a4114a18611063206a016
b1a57a6967d0b60b7bc52395df288bf23f9c43eaefb7e20639b3dd33a074866291af52e9e69e2bc5ab4
ef453b642ad26552e1ae9bb0c2339e41f889558bda4f90fd02753531bb41fdb25f1b73e400fa9fe65fb3c
39634d64d3c665004b3a9a8730026cf337ec7d90f1b4ae67e68500c9ec5e8da07e57cfe0b9d552c85eb
25a5da52bf6105f57ea78c38a47d8b58fc19e0e4507eef3198aefec7005bb22f5570d1923e6b68812c13f
133cc275155d91691ecfd65634527e00b13fd03d1f3a988a9ddce81ad55798153022d29f8e58974df7b3
8a2654399516c5ef3884718c9f05481c5946afe67391ac8be0eae19b4dc9e2960e5c57eb4daa9996c8a
fcaa993ed092bf50d