FIT2093 Assignment 1 – Semester 1, 2019

Submission Guidelines

- **Deadline:** Assignment 1 is due on 31 May 2019 at 11:55pm (Melbourne, Australia time).
- **Required Files:** All the required files for the assignment are included in the ZIP archive file AssFiles.zip available on Moodle.
- **Student Groups:** Students should submit their report as a **group of 3 students** from their tutorial class. Each group should choose one student as the submitter who will submit the assignment on behalf of the group. Please form your assignment groups as soon as possible and email your tutor to let them know of your group members and the submitter group member. The names and student ids of all group members should also be clearly stated in the report.
NOTE 1: If you want to work in a group of 2 students you need to get approval from your tutor.
NOTE 2: If you are unable to find group members, please contact your tutor, who will assign you to a group.
- **Submission File Format:** Only PDF file format is accepted. On various text editor software you can use "Save as PDF" option or use free converters to convert your file to PDF. A handwritten submission is not acceptable.
Note: Do not submit a compressed version of the PDF file or a compression of multiple files. Such submissions may risk losing partial or complete assignment marks.
- **Submission Platform:** Electronic submission via Moodle.
- **Required Student Information:** Please include the name and student id of all students in the group within the main PDF file.
- **Filename Format:** `A1_TutorialDay_TutorialHour_GroupNumber.pdf`
- **Late Submission Policy:** Submit a special consideration form (available on moodle) to formally request a late submission.
- **Late Submission Penalty:** A late submitted assignment without prior permission will receive a late penalty of a 5% deduction per day (including Saturday and Sunday) or part thereof, after the due date and time.
- **Plagiarism:** It is an academic requirement that your submitted work be original. Zero marks will be awarded for the whole submission if there is any evidence of copying, collaboration, pasting from websites, or copying from textbooks.
Note: Plagiarism policy applies to all assessments.

Marks

- This assignment is worth **10%** of the total unit marks.
- The assignment is marked out of **100** nominal marks.
- For example if you obtain 60 marks for this assignment, it will contribute $\frac{60}{100} \times 10 = 6$ marks to your final unit grade.
- Answers to explanation questions will be graded based on the correctness and quality of the answer. Answers to computation questions will be graded based on the correctness of the method/code and the computation result.

Notes

- You can use the `sagemath` tool to perform any calculation necessary for this assignment. The `sagemath` web interface is available at: `https://sagecell.sagemath.org/`.
- **For each question**, you need to show **both** the results **and** your working process such as the source code or the commands you are using to solve the tasks. You will get **zero** marks for a specific question if only the result is provided. The format should be similar to the sample solutions of the lab contents on Moodle.
- Do not write the answers in scientific notation, you need to provide all of the digits for any requested parameter in **hexadecimal** format (similar to provided values). Note that numbers in this assignment written in hexadecimal format are written with a '0x' prefix (e.g. '0xa0b1c2d3'). This is a standard notation for indicating the hexadecimal format and many software packages expect hexadecimal numbers to be input in this way, e.g. this prefix allows SageMath to interpret the value in hex.
- Since the input values in the tasks are huge, newlines are used to fit the inputs within the page width. These newlines must be removed before pasting the values to the tools. You can use a text editor to remove the newlines.
- Try small examples (from lecture notes, lab contents, or other resources) to make sure you are using the proper format for the tools and the correct equation before trying the given values.

1 Task 1 (40 marks)

Alice is using the following "textbook" RSA-3072 digital signature scheme: $\mathsf{sig} = (\mathsf{msg})^d \bmod N$ for message authentication. Given the public modulus N, answer the following questions.

```
N=0xe55c85be1e8f31b8cfa79da46e313545fe58d51308f427be1798373cce2304c0cee692f4ab78387dc5
d3161b5a1f33df90858c5c0a8fe906579257043a527f33e37b3466b7929be81abec6e9979215abf92d71
032caf5fffe4a5f1c176172d8fb62da7beecc255e45b75a44e30ebbeb91ecb97de7dc51a0c1d19f1cb0e
5658b4a66cd4500252dc8f50076c357f5dece3f94ef1133cd2c592a5c9eb22a2e818f95252f0917caf47
737807ece3a0f508f1af03b8eabd2f3d6cc881b27627e3cb5eda7862c25213592ebf1f8470dff22d7603
d299ee69628101c75133d65618692aad5f3b2ffb3a22e1084a900cb0543107b02f8062737181eab4870c
f25f0ed473cf4095530702314dd0a8cace3a6fd0169f2dfea254d3ab152381c3ae535f780a1b532fe040
eae7ba864bf28543a6dec711e62878ec4471341c8ee00824e9cae7627c29de36f3678cbfe046dce37bd6
c7639c51f9387e1b756bda7622efb9ee49fb258266b19fb359ef3f959ffabb0ebf3747bb923cf69899bf
dcaba18bd4dbb7
```

(a) (10 marks) Given Alice's private key d and the message msg, compute Alice's signature sig.

```
d=0x541af9701e04a45700ce962015c835a0d503fe1e5cca2b48a99e47a32473f2ea40f48c2eec31c9
8555657255d5565bcf3f4fb98886d6febc34a0950817dae88a3e808f569b3a47b1751d4013a86109
5166ae2322e6dfe8740d844c8284ab3b29d7c4261efcf2c64c56bd6ce2bf4db3426ee879683cf669
f6c7351c55398cb03a8e4c9a0e3ccbe5d527a3912a8cea045414b7bdef2ffe9a348c56dec274ba67
6e05a224553543910fe6940169f73be36bbca1c0cd53525f53e4b2aa9e69423ef077b2d1bfe8d459
27a677f74418240b95ed5c698e62fb429ece5fabbdcff8f64c480bff46bb6a448ae350739795abd1
56a5814378248b7100bfbaa07b039bc105a32a6fe74e07688577edecd515bd452a41cbfc017b9d26
e76a5bec2ce433714a02f0f2c3784b65738adc849c3c31f8a731132e4bd8c2b2c0b33de87403c2b7
ff12ab3d9582453844b4ff03142f899b256e407c3301adc46794d14bd668beac877e9cb5aa0602c4
47b75d3424d3a71495ed55a86fb1b01b5fbae2a766f6172301

msg=0x6b948843b86adb04a834cba6a76d5753da8ffbdcd01782a49d395f52f4c37a0cc39eeb41646e
bc2b2003bacb203328e210604f248e02fa95aa6eec50751abe267f5c0b70f60901a4fd338f61bb20
00acb3f2cf80d602acf85c5ee2f015667e9520e2d5c1aa84dcc69c9358a376846d2a0e9b52877fe1
7a76ce4bf6c46c7a46f61102d42869e0a594c4ad71a699a603654e4d6bdf83fc09b9741b70e82013
```

```
302517efceebc9be49a7bc86ab89653f3281ffcc20824970410461510f4a9b538f8d5468b872cbef
23a348b61576ae1f840138f14e7f8f13643aae1467cd534803555f8b2facb34fae15d53dc8c954bc
8af0561597bfbb5a82c3b08bc83d349962aaaa6e164a138045b96dd9730aa7e1bb440838c42296ff
2bdf53ca69f09c7e74c5e855455ffb052399e82e7e182d8efa08c96bdd166a00381d3fc53bb2a3d4
6b0aa6e2af8a45cd00e8bffe34fb7bafd20dade1efece7331b417136e2ed971c8f16e193948f3c65
95e9f63a948610f1d3e2246e6603d0b039f9bdd50fc50baadef
```

(b) (10 marks) 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 $(\mathsf{msg1}, \mathsf{sig1})$ and $(\mathsf{msg2}, \mathsf{sig2})$ Bob received from Alice, show how Bob verifies the digital signatures. Do both messages pass the verification?

```
msg1=0x9cca26cb7a1713c2c95ee703089b84bd27311c5750c2a817586e7b1fed6e12a8051a4626b48
656a229eec292e30f0751d59ce1544300919801303cef1a08d08015ba5ea047e4348b340aca99b3f
cbcde9663b1a8d59eed23b6e1e834889729e53a691343f589babff5c8f8fe661bd0a643644dde1ba
9f37023bb01af97dd12bfbc5517a2a1a3e67108f0b287e7dbe9c9d81fdc5c1c7875c8577c6acd7ee
aee2a46a5ce3c2d5123b085cdb554c36fa3a2f756f3e4515a0b0bb5aa504ebcdf8d9e8837b2d3b8b
60eea5658be5dc9f3f9cddf1449b226d668591144a7bc4f17bf4e51a56ea29b0f6d7321054f208e9
65b022a0668e2563a058626ea7347d9fb776e5596198fc991b1dd450b0e621ab9b11e10a0ab5bdb9
f572f8f4b82f9edb5b17d154371118d3be51b28d08542c2d7ff5e99674a070d0c08fcb55f8cfdc8a
3739194b0d2ed99a34bbc70b0c43dc709be2bafce3255a6c747461b5bd24798160549d3ea9b37691
039d4d5482640bcf3297cce754358670761f22876c78a4ac69ead

sig1=0x2be8a53416b161b0b7e76e28046a7a8b923417536fc6b27f2cb1666daaf6ab0292b696d9946
134f3de1e4a9db1a0d47c5c888b0699bc29bf497dbcc49bd643bf151a06514d45f418aca2198a622
0d56970d7c15bc65caaac568148fa03f1d39d8cbf3b6e919e8ed3ff25655f15e18e0b89109f8f337
dcb80853dd22d73461dd3956e83e97debf6a5878f1219682cac4d2a71a2fc662743221d2faba4fe8
ab2f02055bbc895c38475c95590c9b08beeec49217bbc52a3771012022a9c537477892b1eeadf17e
f5f9471d1d40abd097ca9026fa321df01add8289bf611bd4c029ecab9c0539a22b9360f241950bcb
9b32dc339ca9c94054c0be4e1772629b7344a544c0eb566895d64ee773d220e417f9805bb3645751
5474146a22264c05c1a8f1a23c156c87111d4198dbee9e89fc14e059e0e9030c5d1ce2327ebbe990
99d006c99e5d0cfb9ea2cd94b944c8933277e53f63a0e0664c9a652711ac0438bab41c6107924afa
a77a5953c02e235a18f320822cdbb2621ea0912631d55aefd1a4d
```

```
msg2=0x31c276bb243008ad8ee81ee029e80aad7e9ff16ed54dafe20649756f6bae1b57fb095865c4a
902d55892d0d22e3ba2ca62d7bc3a069a18f9c8df8e5b09a640ab1dd35bac240e6fbec27d7089abd
cf943d3894cbe1e13a2db39dae0a1409259b76ead144ec7a8c308c4bd1ca1cc8133de63c46d80925
10ffa422bf8827d81e377bf70a07e6e82ed8b863cdc2aa6705731237f79f36aa6c35ca3f03542f0a
7d5c56e3711b96b20c7bbb0da837c6cc3abec24783d2b95de9e6bc052b81d21955912d40a18b03ba
9fcd206c37ccac389524b4ef4835822ea0cb3524ecd1a47ca2baa4bc66fd3dc4ba174aab59088019
f9932102709519f8146d3d5af858077f351c97d277aaacf9a832e0b4271475bf5fd29e380149e2bb
d443f0e0363f7e96d2a3f02e384a01caf6b11afde551696f411f26e603225fde420deec3f4f715ab
b5e445180d2717870b2285f761f0baa917751512442ddbfd05e529a15b649f6d45aa93fb31626ab9
a498d98612e225140c98551d0851057c332366e60f39234e0c711

sig2=0x79c8b72ca72f4c6363b3e29c1a6267ede2a1740dde90a071b8600f98f14eeee19b580213a87
2c00fb5146a851285945bda4728437622ab9e0f800881a7ebaf71cfcb558c8de0a150e1452443808
614f0e96dbbeedfabcdbf01e41b4b9601935bf9f12c5947d7a066c236d6843bcf05d136e1cd480ed
a39a40f3fa9e5a1b26033643859ad5b5bc91b185bd980d2efa223c5025c13389e542167999282c8c
b5aa180cfa89746f377bb3b2923bde3be1b6fa05980b6a80a9b52136dd3b933dcd54a095dacb8d1c
9fa7c8dbf96e421ae713440bba2f3c82c31c356a268d3623e7c1510a8a6ca506943f843682c73179
eaa35a9678ad599b2a41881a2b47234deb25640771b9ee8ca4f488b21d735bf3adc1ae37a786dbc0
622d7ba31a218d02567355af578b187eeef9de6b37feb408f3ac296d5410c9d4d2920f452e30cf21
```

```
5227075756ff2f4fb0a15741102c31c1e59662767d78691bf864f3fbda722da50b8e02e88f7b6029
eb85da47c44439b90e9cd4d027d171960b9438d9c9c73d211e555
```

(c) (10 marks) Assume the attacker Marvin intercepted the following two message/signature pairs $(\mathsf{msg1}, \mathsf{sig1})$ and $(\mathsf{msg2}, \mathsf{sig2})$ signed by Alice using the same private key (d, N) from (a). Explain how Marvin can perform an **efficient forgery** attack that, given $(\mathsf{msg1}, \mathsf{sig1})$, $(\mathsf{msg2}, \mathsf{sig2})$ and Alice's **public** key (N, e), computes a forgery message/signature pair $(\mathsf{msg3}, \mathsf{sig3})$, where $\mathsf{sig3}$ is a valid signature for the **new** message $\mathsf{msg3}$ (here, new means that $\mathsf{msg3}$ is not equal to either $\mathsf{msg1}$ or $\mathsf{msg2}$). 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.

```
msg1=0xa9edfdd28f7b79039fc041e8244d3d06bbac0e2cd108e1c5fd5691ae03545cf27a465bee121
6e6f97f5d2443767a32a6dcb46f012aa0ac19cf5e6b81097e2846cea4eab599620fc876ef6071d92
b67cec573d91366301a60f90efab964ccfbda9b5fc197ba86bdfb9e7a5380f7e28c90e7149a3d8ca
5d443e22df5e284cbb5700c2f89df1c6a7d21dca87a856523bf1a37e44e85e76a03be641ff9366b6
aa8552bee1763bca3cdba155899137fb8fae54fa4558a9cfcb5dfabcd1473068b93ca

sig1=0x75b4bfc2420836f896bde21d195204a875867d43765194babbf67a3b9803515c7c1779ecf9a
b20266071493a3b2e12272e7ae1a1f030055c51eb1076814a7bdde56b9381644e892a0a32d5b176a
f4428db1bddc53df9b28e0d7949c660559672fe497f051c978f407bbb961bfed8841e5bd46d52301
0355535d01246da0d821a9389537e1747f8c296dff83ba22bbd5993300c92846ea288aade9fb0591
c3bb3dd25372c15224a5ac3588734144190fce710a2e07493cc6bb0ad80f205667a4264ab0d1b139
b8ac8fc2f35a89ec2f9c6c159f683ef111796eb8d6b0d02d73f5cc890af958c9ea7fc9ada016bb53
c80191babb37facd0ead5093a969cc5947775f48b5b80797533d1cd40987eb537d63221c51dc8786
3d5ba6fe30fea2d1831fda5334e16b6e2c6db117d7eb5bcc918f6718213f6902fce55b498ca1f381
fba98f49c6858c65168de2416cb8408f6838d12e5c99c6d1e0aea0e6c3dcbda5075dea991ac42c75
9b835e9bd4ff303eab6702e798ed284ebeda6e50754828cad511b
```

```
msg2=0x4eaacda480337afeea561e82087deb98a9b16e84a7fdb9f6c586e37e04f74c42a891d2dbb39
7154a0e76b7df72c0975702965a1ca2a70bad04a7285b0a5618ef9bca9070f76c1930225f56f58d1
7b15b8954a35e08223f505c9c2e93cf8bf7f28c50647d385863dc36bd9e52556c896e89e9073015a
7c38a59f29141247014518444450a2e4c792b70e99bc730f82005154f7e6c79b4ef394aaa155c524c
18da44ea8615ca0cce07aa822c0ec6704902bef72db21f3302cfccf3ab79458c1f197

sig2=0x77dd1b15bcf1cf5dc65d64408a18771a7cfb2af1833de20d7802ddd85d2c09429f52bf9e1c2
700a39f22f2d98933e22fc57876f30a1d86696a6970fae76d89e5556046fe1f91a07015714fc3d30
2e326bbe364f5784ffb0a6c68f33a693a758e442e1e2410731d2f6e77776e0435f5d806c8b149aef
99fcc8f49620b487a6703178839d3658b4c5008e2db841383a9aceb0862aec852f6af50926f9045a
fff5e40ab52929993e3b79bca23777f06ea2914a284b90254798d131dfd0e8b97d0c94bfa589ed9a
75a4d4ea045fecf7325eb6e91ca6536113d6acbe6e83a1dfac9833edafb8d2d25439a8eb0ad6cec2
89d8628cc0733ca231409065c3215e3ad5e2f5ad61756e6d4bd5b14fffb78e9d1f470057061f0b9a
b988f3be141435a8a71f70f7084a20f6ce13ac2aceba4546afc6fd28b305dbde81c7b54be192c69f
118ce5d131b5f217e2ec804c1e11578cbff51270a4fd166322dd69aa48a18466f608c2621baf8280
02d55c8361cbb1f7b3279c986bdf1c3da2c12fe565e7c15d7250
```

(d) (10 marks) 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.

2 Task 2 (40 marks)

Use `gpg` and `openssl` to perform the following tasks.

(a) (10 marks) Alice wants to use the "Hybrid" encryption method combining RSA and AES-128 in **8-bit CFB** mode (CFB-8) to send an encrypted file to Bob (email address: `bob@fit2093.edu`). Given Bob's public key file `bob.pub` (this file is available on Moodle), Alice's AES-128 session key `K=0x6cfba139ea2c55e5ecff60429ce20ade`, CFB-8 IV value `IV=0x6b66a2ce1972853f84d1874736369036`, and the plaintext file `msg_qa.bin` (this file is available on Moodle), compute the two components of the hybrid encryption ciphertext in two files: (1) the RSA ciphertext component in file `c_rsa.bin` and (2) the AES ciphertext component in file `c_aes.bin`, both files written in **hexadecimal** format. Show your code / working process.

(b) (10 marks) Bob received the two ciphertext component files `c_rsa_qb.bin` and `c_aes_qb.bin` encrypted by the hybrid encryption scheme in (a) from Alice. (NOTE: these files are available on Moodle and are different than the ones you have computed in part (a) of this task.) Given Bob's private key file `bob.prv` (this file is available on Moodle) and `IV=0x5fe4bbaf52dfd660407a9a8e123901ea`, show Bob's decryption process and the decrypted plaintext.
Note: Bob's `gpg` passphrase is **fit2093**.
Hint: The symmetric session key `K` is encrypted as a **binary** file instead of a text file. Use a hex editor to get `K` in hexadecimal format.

(c) (10 marks) Due to network errors, instead of the files in part (b) of this task, Bob actually received the file `c_aes_qc.bin`, which contains an error in the **2nd** block (this file is available on Moodle). Compute how many blocks are corrupted by using a **mathematical formula**. Verify your result by decrypting `c_aes_qc.bin`. Show your verification process.

(d) (10 marks) The attacker Marvin intercepted two ciphertext files `c_aes1_qd.bin` and `c_aes2_qd.bin` sent by Alice. Marvin found out that due to a vulnerability in Alice's encryption software, Alice encrypted both files by using the same AES-128 key `K` and **same** `IV`, and he also managed to get the plaintext file `msg1_qd.bin` corresponding to ciphertext file `c_aes1_qd.bin` by hacking Alice's computer, though he didn't find the second plaintext file `msg2_qd.bin` corresponding to the ciphertext file `c_aes2_qd.bin`. Explain how Marvin can use his available information to find some information on the second plaintext file `msg2_qd.bin` corresonding to ciphertext `c_aes2_qd.bin`, even though Marvin does not know Alice's encryption key `K`. Show the content of the plaintext blocks of `msg2_qd.bin` that Marvin can compute.

3 Task 3 (20 marks)

After Alice did the mid-semester test, she has a new idea about how to mitigate the vulnerability of the PIN encrypted by the "schoolbook" RSA-3072 encryption ($c = (\mathsf{PIN})^e \bmod N$). Assume PIN is a positive integer smaller than 2^{16}. Instead of directly encrypting the PIN, she writes the PIN in binary format first, then she appends **2048** 1 bits at the **rightmost** side of the binary representation of PIN. For example, integer abcd in **hexadecimal** becomes an **integer** x:

$$x = 1010101111001101\ \underbrace{1\ldots1}_{\text{2048 1 bits}}$$

in **binary** format. She then computes $c' = x^e \bmod N$ as the ciphertext.

(a) (10 marks) Explain an **efficient** attack against the above mitigation technique. Given the public modulus N, Alice's public key e =0x10001 in **hexadecimal**, and the ciphertext c' encrypted by using the above mitigation technique, show how the attacker Marvin can recover the PIN **efficiently** without Alice's private key.

```
N=0xa05bb3783769b832a2d022646c48344948282cdcd42bff414ec90f23b7e3b1b817137664f40163
19586395741996245f9c2c66dee453352dc329fe54228beaa559a610114dbe902c32572e954660ad
bd06f8da8c770c33bb5ad15f506073ea0c50ff4e9906e16ee70d1311e0ad81896f4807282361f5b2
116488de06966b571cdb15da536226378bc1fba8a3476c5809b5a274a0117b5de3e52278d39fdfa6
2de29f338b0453ac3af61a30dcb2975949a3d0ec2d2b7f0d2c4d2e3ef6ddefa8caad21bc16972dce
cfcd5f9332373a759632f7f02c52dd424b83985eaa673ce67023366e85899729fc1d1fede02fa9c5
3aa01328c9108a3c5145f47ef988688f3076d49821314210d1f4db88fa836d41f3dc3960499eb46b
28261aaa1515e0fb6d7481ae051b607683cbfdc18d6b692f93d6facf4002d6fa835aac4d61911b66
859a81043763e1d0ef6e47f1a7a4c8d57993b0fb67b5758ed3aca9540d39e150935cdd0c320d166d
a65612ae78322f96853885e6a44add306a899fab2f87cf2a1d

c_prime=0x90fdea0c662ed2cef739c491c2f391d8cf636b80144c412580c02e3262e4fa10c6e101f4
c53d09619c7cb6fc9d8edfe2a676c1c128bd8e32528aff243101b9daf655bcd5460a9bf020ff4bef
0f61b94304b142b6b18830b8b4d5574e8b54903de67df71f39234fdf9f66723ab1bf426d1c0a95fa
bae8485e9edf7f4c868ca2816398b1f46ffda2a84b5d52ff36bad829ddc2e123f86cb266256824f0
47fb6f6a1c7593eaf4ae5c47c6f5e633370d832345fde53324d02687a9b21e60fcefb5e2e2eb1ace
969fe72afca67847acad093dec8976336ace5f135257f740f625851a3258854775a3f4f123eae1a6
253b2740de37d112bca596f36e4c0d4cfc50b05643b8ec0b52619ae7d0ae990e041ba01bc149ac4a
510c81e3aef3f4f2843a50f15c637e274e714c6a768e0c7d96e28a5365b64aee0315623794724573
648516ebc9b0f5135a180ac3141a98f2ef0f005f6980781036c9b1c7975774708d1929d1935ae782
de80722124220a9dd3fadc457d8bdb8be762b0158187ee619142637d
```

(b) (10 marks) An implementation of the RSA-3072 encryption software that Alice uses has the following vulnerability: the software neglects to clear the value of $\phi(N)$ from the memory after the key generation process. An attacker Marvin who manages to gain access to Alice's computer exploits this vulnerability by performing a memory dump of the machine after Alice completed her key generation, to get the value of Alice's $\phi(N)$. Explain how Marvin can **efficiently** factorise N by using this additional information. Show how to factorise the modulus N from (a) by using the following $\phi(N)$.

```
phi=0xa05bb3783769b832a2d022646c48344948282cdcd42bff414ec90f23b7e3b1b817137664f401
6319586395741996245f9c2c66dee453352dc329fe54228beaa559a610114dbe902c32572e954660
adbd06f8da8c770c33bb5ad15f506073ea0c50ff4e9906e16ee70d1311e0ad81896f4807282361f5
b2116488de06966b571cdb15da536226378bc1fba8a3476c5809b5a274a0117b5de3e52278d39fdf
a62de29f338b0453ac3af61a30dcb2975949a3d0ec2d2b7f0d2c4d2e3ef6ddefa8c915dbdc153c25
b847c313f96c30a78950106adcc70eef014d1340f26f0fd36a90d6a5e1c369a70658dfbb20feccf4
efd255d477924a95ae093387182cf946b4dae80d6b434fcb11f2a8e9265b23e7dd076733f268d8ca
cf0ba15aeee50b7fc577c7db1269f54436c8c9ade14d23f27097e128a4a312eb9c9e7cd9fc5c40ef
e18a6b3b56947761243265d6a3ccd7bc9027e6e4ecc267765ea293574502e955349b0d866ad5a16f
92bf6e96273d24dd25807554de152e65fa7273818bb3f013a73c
```