**CSE 3140 Lab 3: Cryptography, Malware and Ransomware**

Tom McCarthy (tkm20002)

Section Z81

VM IP: 172.16.49.96

1. Name of matching file: flintlock.exe
2. Name of matching file: unbestowed.exe
3. Name of correctly signed file: spreadeagleism.exe

**Experiment:**

Using the python file Q3Experiment.py (submitted with this lab report), I tested how the length of the key affects time required to sign and verify signatures. The script generates keys of length 1024, 2048, 4096, 8192 and 16384 bits. For each key, it signs a hash and verifies it before outputting the total time taken for each key. The results are below:

A screenshot of a computer code

Description automatically generated

Clearly, the time it takes increases substantially for keys of increasing length. The value of hashing is apparent; if long contents were not hashed first, then the length of the content and corresponding key of the same length would cause the system to take a very long time to sign and verify the contents. Hashing shortens the content to sign, making the process more efficient.

1. Decrypted file contents: binaurally87&
2. Decrypted file contents: councilman78@
3. Approval Code: RHHMY6

For my ransomware system, I used a shared key, k, which is 16 random bytes. This is a standard length and it is sufficiently long to assume brute-force methods will not crack the key. I encrypted and decrypted the key using [RSA PKCS#1 v1.5](https://pycryptodome.readthedocs.io/en/latest/src/public_key/rsa.html) because it is an established and widely used system. I used modulus of length 2048 bits, which is a widely accepted standard for sufficient length.