Aim: For varying message sizes, test integrity of message using MD-5, SHA-1, and analyze the performance of the two protocols. Use crypt APIs

Theory:

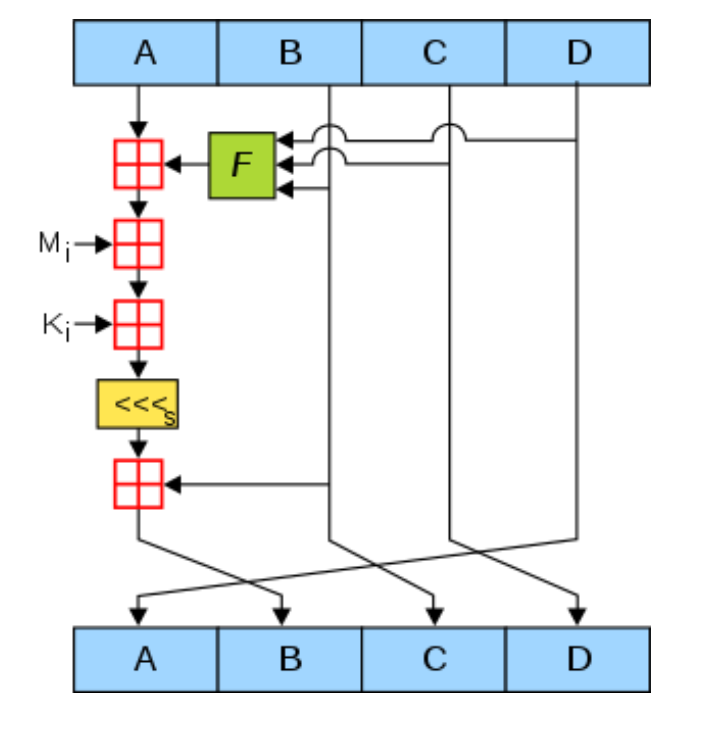
Explain MD5 in brief.

The MD5 hashing algorithm is a one-way cryptographic function that accepts a message of any length as input and returns as output a fixed-length digest value to be used for authenticating the original message.

The MD5 hash function was originally designed for use as a secure cryptographic hash algorithm for authenticating digital signatures. MD5 has been deprecated for uses other than as a non-cryptographic checksum to verify data integrity and detect unintentional data corruption.

Although originally designed as a cryptographic message authentication code algorithm for use on the internet, MD5 hashing is no longer considered reliable for use as a cryptographic checksum because researchers have demonstrated techniques capable of easily generating MD5 collisions on commercial off-the-shelf computers.

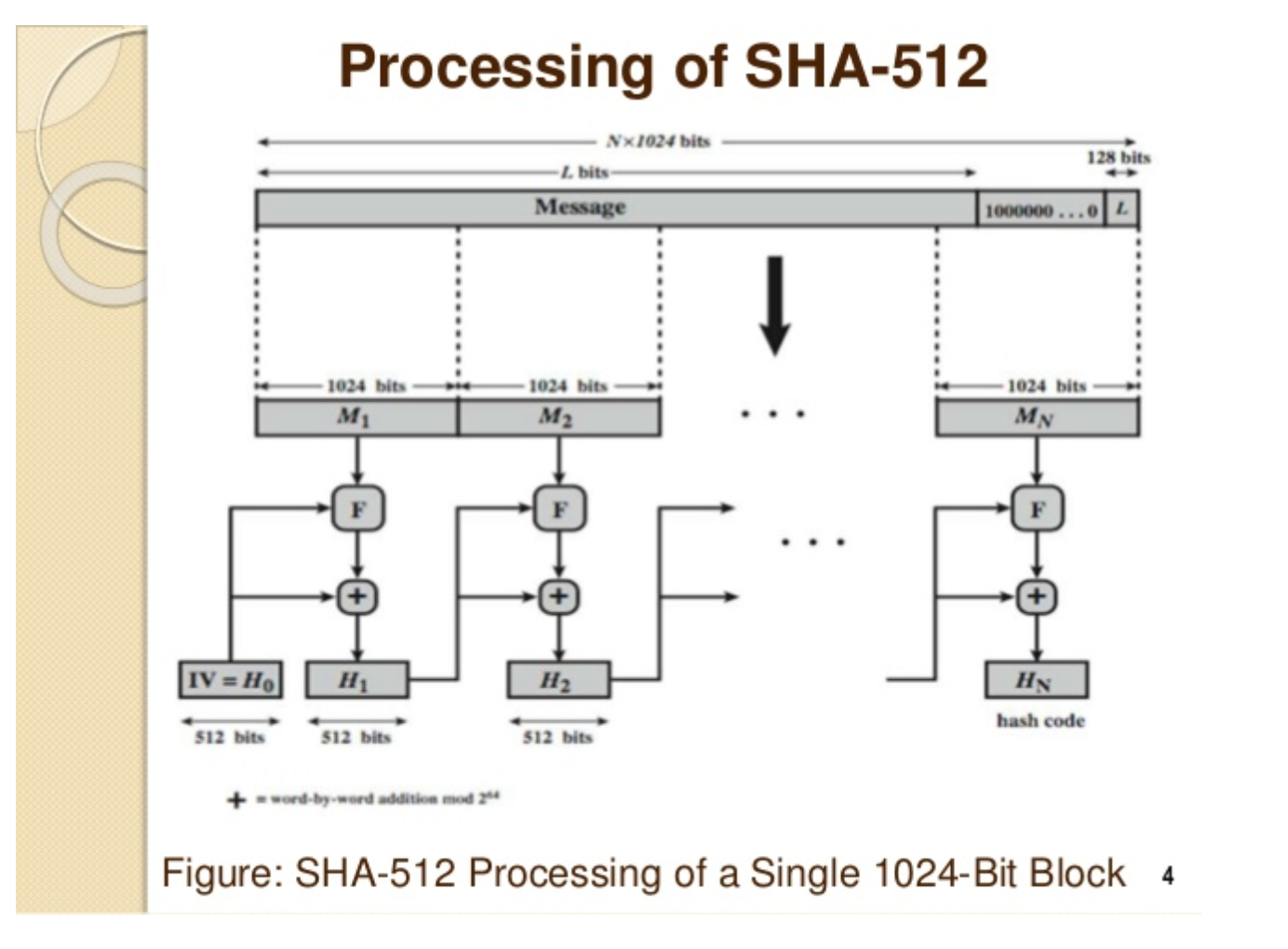
Block Diagram of MD5



Explain SHA 512 in brief

The SHA-512 compression function operates on a 1024-bit message block and a 512-bit intermediate hash value. It is essentially a 512-bit block cipher algorithm which encrypts the intermediate hash value using the message block as key. Hence there are two main components to describe: (1) the SHA-512 compression function, and (2) the SHA-512 message schedule

Block Diagram of SHA 512



Application ofMD5 and SHA-1

MD5 was developed to store one way hash of a password, and some file servers also provide pre-computed MD5 checksum of a file so that the user can compare the checksum of the downloaded file to it. Most Unix based Operating Systems include MD5 checksum utilities in their distribution packages.

SHA-1 (short for Secure Hash Algorithm 1) is one of several cryptographic hash functions. SHA-1 is most often used to verify that a file has been unaltered. This is done by producing a checksum before the file has been transmitted, and then again once it reaches its destination.

Program:

import hashlib import time

with open("npp.7.9.5.checksums.sha256","r") as fi: lines = fi.readlines()

each\_line=[]

for line in lines:

each\_line.append(line.strip().split(" "))

BUF\_SIZE = 3871000

md5 = hashlib.md5()

sha1 = hashlib.sha1()

sha256 = hashlib.sha256()

with open("npp.7.9.5.Installer.exe","rb") as fileref: while True:

data=fileref.read(BUF\_SIZE) if not data:

break

md5\_start = time.time()

md5.update(data)

md5\_end = time.time()

sha1\_start = time.time()

sha1.update(data)

sha1\_end = time.time()

sha256\_start = time.time()

sha256.update(data)

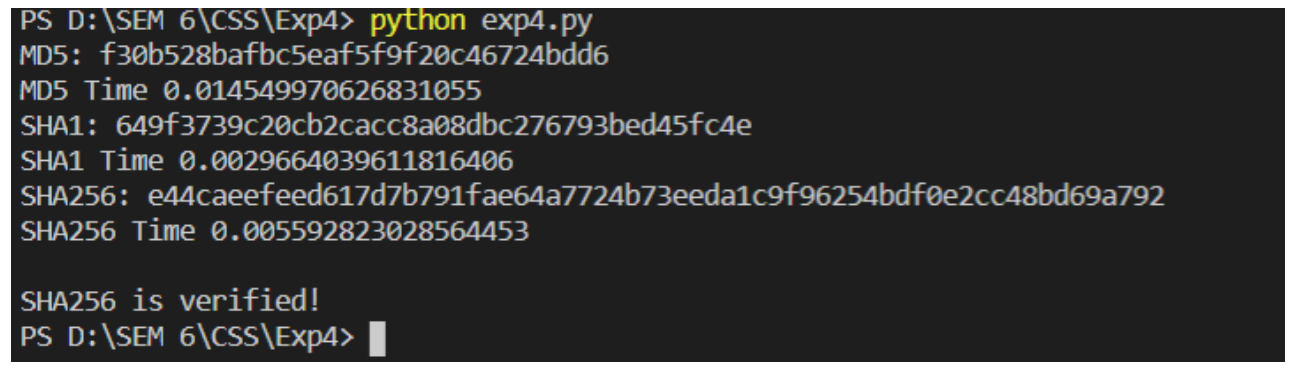
sha256\_end = time.time()

print("MD5: {0}".format(md5.hexdigest()))

print("MD5 Time", (md5\_end - md5\_start))

print("SHA1: {0}".format(sha1.hexdigest())) print("SHA1 Time", (sha1\_end - sha1\_start)) print("SHA256: {0}".format(sha256.hexdigest())) print("SHA256 Time", (sha256\_end - sha256\_start)) if each\_line[4][0] == str(sha256.hexdigest()):

print("\nSHA256 is verified! ")

Output:

Conclusion:

In this experiment we learnt about the hashing functions and using the different hashing api functions available. We were able to see the speed of the three hashing functions we used and were also able to verify the hashing values produced from the functions by comparing it to the officially verified value of the file.