Algorithm Flow

The SHA-1 algorithm works by converting any length of input message into a fixed 160-bit (20-byte) hash value. In this implementation, the process starts by initializing five 32-bit hash buffers (H0 to H4) with predefined constants as per the SHA-1 standard. The input message is then padded according to the SHA-1 rules: a single `1` bit (0x80) is added, followed by enough `0` bits so that the total message length (in bits) is congruent to 448 modulo 512. Finally, the original message length (in bits) is appended as a 64-bit big-endian integer. This ensures that the padded message length is always a multiple of 512 bits.

Each 512-bit block is then processed through 80 rounds of bitwise operations involving logical functions, rotations, and additions with four different constants (K). The algorithm uses the `f(t, B, C, D)` function and circular left rotations (ROTL) to mix the data across the 80 rounds. After each block, the working variables (A–E) are added back to the hash state array H, gradually updating it with the processed message data.

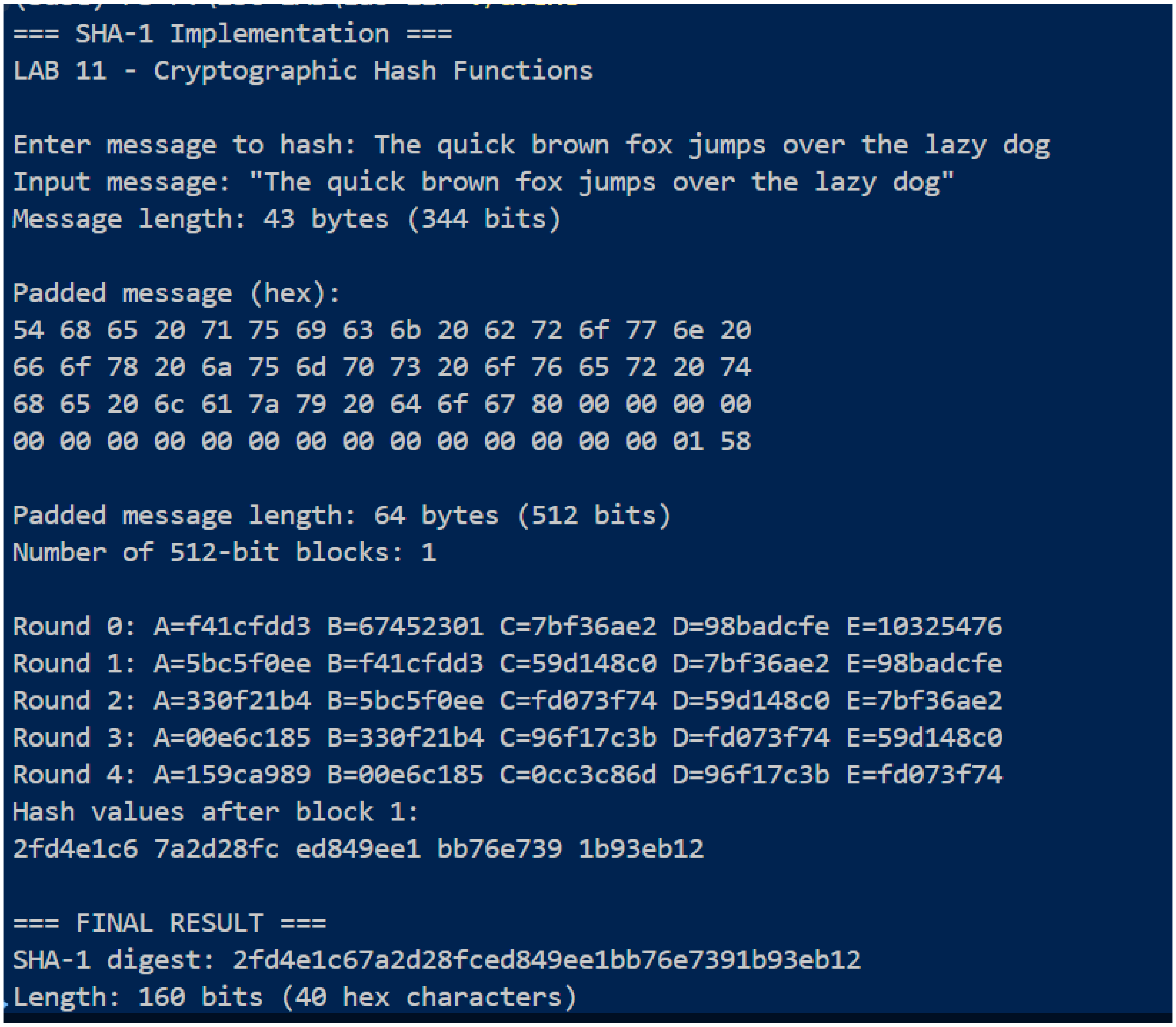
Observations from lab

While implementing this build setup, one key observation was that Windows environments handle build commands differently from Linux. Commands such as make and nproc that are common in Unix-based systems do not work directly in PowerShell. Instead, Visual Studio’s native tools and nmake must be used for successful compilation but as I already informed you it didn’t set up correctly.

Another observation was that intermediate variables (A, B, C, D, E) change rapidly over the 80 rounds, and printing a few rounds (like t < 5) helped confirm that the transformations were happening correctly. Testing showed that the initial constant values of H (from the SHA-1 standard) are crucial - changing even one constant completely alters the final digest.

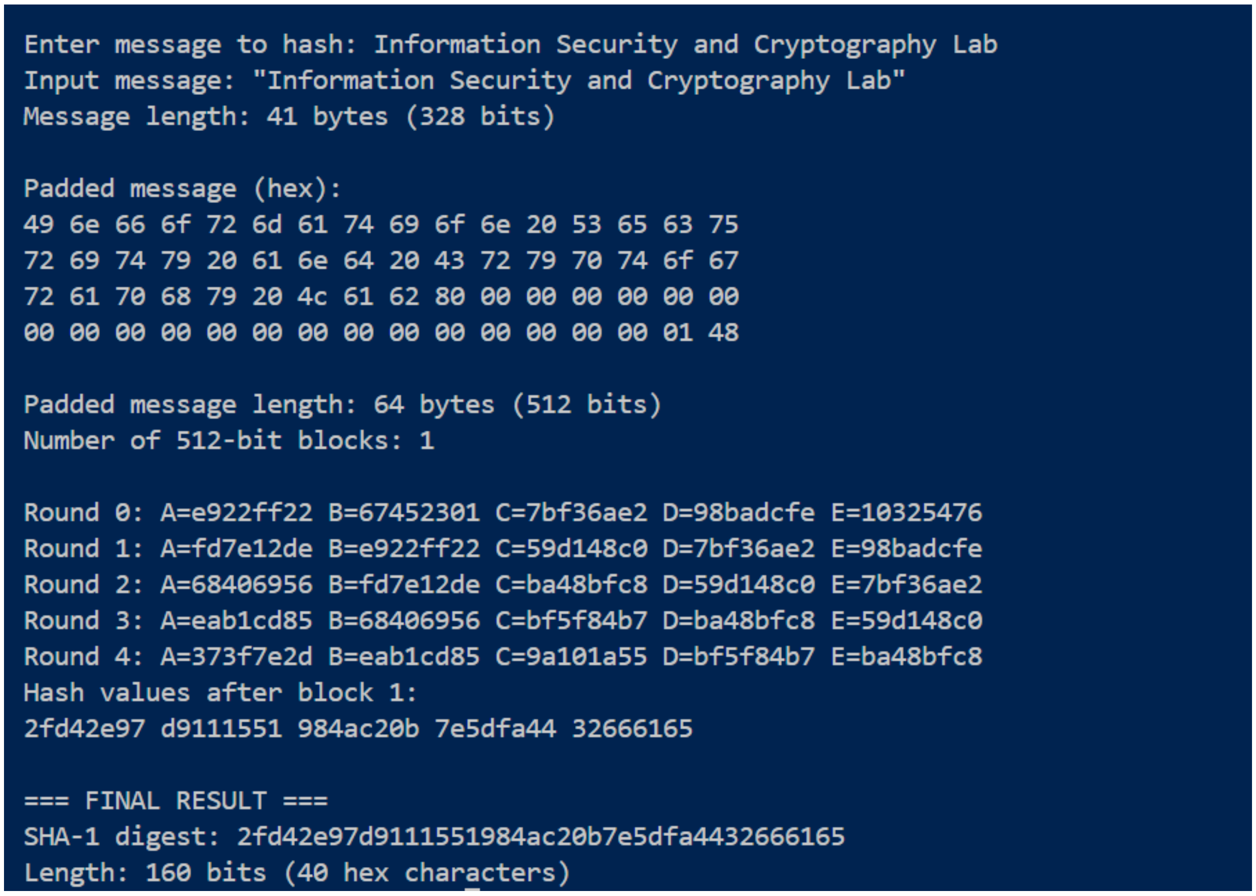
Outputs

1> “The quick brown fox jumps over the lazy dog”



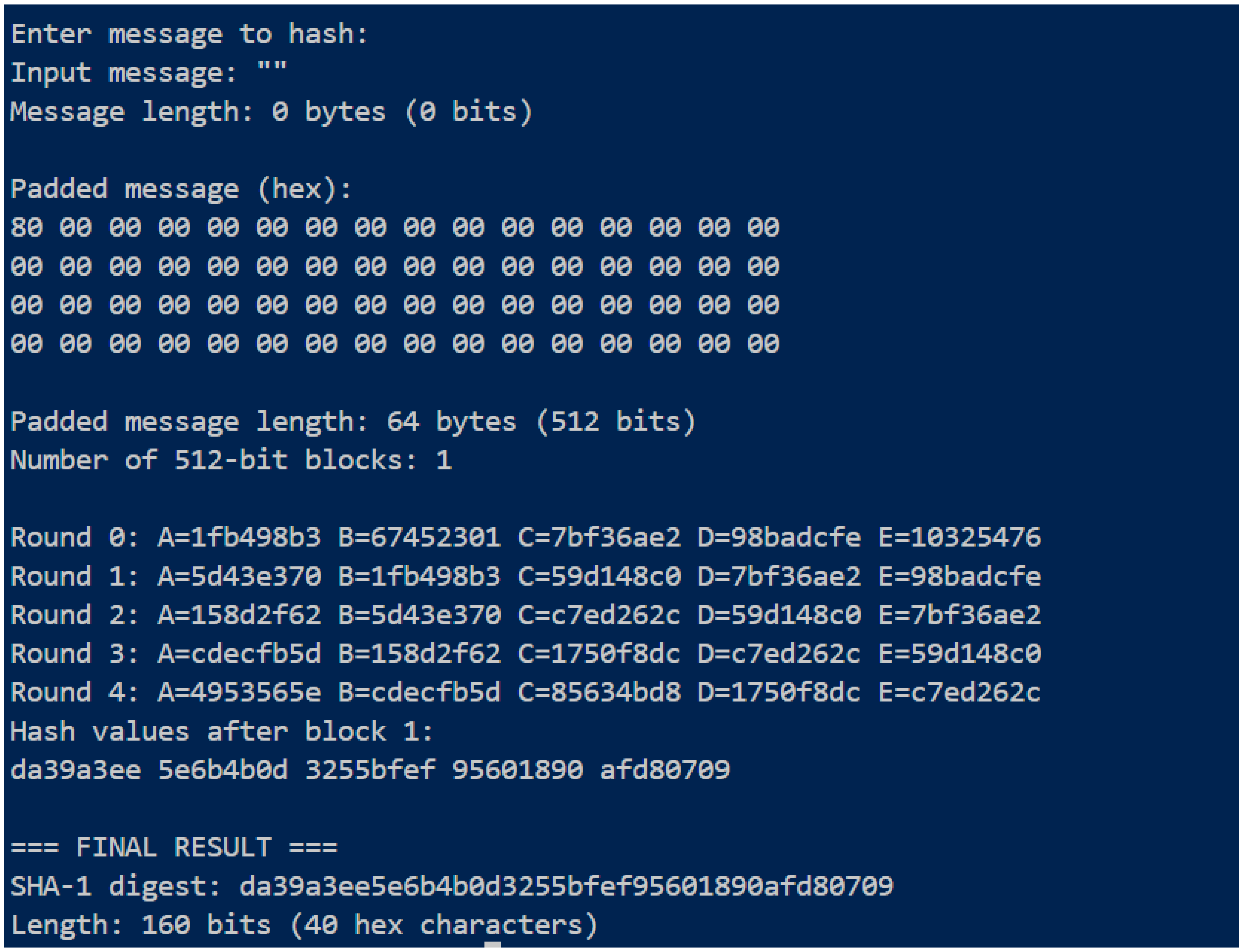
```
=== SHA-1 Implementation ===
LAB 11 - Cryptographic Hash Functions

Enter message to hash: The quick brown fox jumps over the lazy dog
Input message: "The quick brown fox jumps over the lazy dog"
Message length: 43 bytes (344 bits)

Padded message (hex):
54 68 65 20 71 75 69 63 6b 20 62 72 6f 77 6e 20
66 6f 78 20 6a 75 6d 70 73 20 6f 76 65 72 20 74
68 65 20 6c 61 7a 79 20 64 6f 67 80 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 58

Padded message length: 64 bytes (512 bits)
Number of 512-bit blocks: 1

Round 0: A=f41cfdd3 B=67452301 C=7bf36ae2 D=98badcfe E=10325476
Round 1: A=5bc5f0ee B=f41cfdd3 C=59d148c0 D=7bf36ae2 E=98badcfe
Round 2: A=330f21b4 B=5bc5f0ee C=fd073f74 D=59d148c0 E=7bf36ae2
Round 3: A=00e6c185 B=330f21b4 C=96f17c3b D=fd073f74 E=59d148c0
Round 4: A=159ca989 B=00e6c185 C=0cc3c86d D=96f17c3b E=fd073f74
Hash values after block 1:
2fd4e1c6 7a2d28fc ed849ee1 bb76e739 1b93eb12

=== FINAL RESULT ===
SHA-1 digest: 2fd4e1c67a2d28fced849ee1bb76e7391b93eb12
Length: 160 bits (40 hex characters)
```

2> “Information Security and Cryptography Lab”



```
Enter message to hash: Information Security and Cryptography Lab
Input message: "Information Security and Cryptography Lab"
Message length: 41 bytes (328 bits)

Padded message (hex):
49 6e 66 6f 72 6d 61 74 69 6f 6e 20 53 65 63 75
72 69 74 79 20 61 6e 64 20 43 72 79 70 74 6f 67
72 61 70 68 79 20 4c 61 62 80 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 48

Padded message length: 64 bytes (512 bits)
Number of 512-bit blocks: 1

Round 0: A=e922ff22 B=67452301 C=7bf36ae2 D=98badcfe E=10325476
Round 1: A=fd7e12de B=e922ff22 C=59d148c0 D=7bf36ae2 E=98badcfe
Round 2: A=68406956 B=fd7e12de C=ba48bfc8 D=59d148c0 E=7bf36ae2
Round 3: A=eab1cd85 B=68406956 C=bf5f84b7 D=ba48bfc8 E=59d148c0
Round 4: A=373f7e2d B=eab1cd85 C=9a101a55 D=bf5f84b7 E=ba48bfc8
Hash values after block 1:
2fd42e97 d9111551 984ac20b 7e5dfa44 32666165

=== FINAL RESULT ===
SHA-1 digest: 2fd42e97d9111551984ac20b7e5dfa4432666165
Length: 160 bits (40 hex characters)
```

3> “”



```
Enter message to hash:
Input message: ""
Message length: 0 bytes (0 bits)

Padded message (hex):
80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Padded message length: 64 bytes (512 bits)
Number of 512-bit blocks: 1

Round 0: A=1fb498b3 B=67452301 C=7bf36ae2 D=98badcfe E=10325476
Round 1: A=5d43e370 B=1fb498b3 C=59d148c0 D=7bf36ae2 E=98badcfe
Round 2: A=158d2f62 B=5d43e370 C=c7ed262c D=59d148c0 E=7bf36ae2
Round 3: A=cdecfb5d B=158d2f62 C=1750f8dc D=c7ed262c E=59d148c0
Round 4: A=4953565e B=cdecfb5d C=85634bd8 D=1750f8dc E=c7ed262c
Hash values after block 1:
da39a3ee 5e6b4b0d 3255bfef 95601890 afd80709

=== FINAL RESULT ===
SHA-1 digest: da39a3ee5e6b4b0d3255bfef95601890afd80709
Length: 160 bits (40 hex characters)
```