ESE532 Project P1 Report

Ritika Gupta, Taylor Nelms, and Nishanth Shyamkumar

October 30, 2019

1. Our group makeup is Ritika Gupta, Taylor Nelms, and Nishanth Shyamkumar.

2. (a) We end up with $64ns$ to process each $64b$ word of input, which comes out to 76.8 (so, 76) cycles for a 1.2GHz processor.

   (b) By similar logic as the last question, with a 200MHz clock, we end up with 12.8 (so, 12) cycles to process all of the input.

3. (a) (i) **Content-Defined Chunking**:

```
skip input to minChunkSize - windowSize
buffer = input[minChunkSize - windowSize : minChunkSize]
curHash = 0
for byte in buffer:
    curHash += hash(byte)
if curHash == 0:
    markChunkBreak()
else:
    while (curHash != 0 and (notAtMaxChunkSize())):
        curHash -= hash(buffer[0])
        moveBufferWindow()
        readNextByte()
        curHash += hash(buffer[windowSize - 1])
    markChunkBreak()
```

   (ii) **SHA-256**:

```
h[0:7] = initializeHashValues()
k[0:63] = initializeRoundConstants()
padInitialMessage()#pads to a 512-bit boundary
for chunk512bitSection in chunk:
    w[0:15] = chunk512bitSection

    #Extend the first 16 words into the remaining 48 words w[16..63] of the message s
    for i from 16 to 63
        s0 := (w[i-15] rightrotate  7) xor (w[i-15] rightrotate 18) xor (w[i-15] righ
3)
        s1 := (w[i- 2] rightrotate 17) xor (w[i- 2] rightrotate 19) xor (w[i- 2] righ
        w[i] := w[i-16] + s0 + w[i-7] + s1

    a:h = h[0:7]
    #Compression function main loop:
    for i from 0 to 63
        S1 := (e rightrotate 6) xor (e rightrotate 11) xor (e rightrotate 25)
        ch := (e and f) xor ((not e) and g)
        temp1 := h + S1 + ch + k[i] + w[i]
        S0 := (a rightrotate 2) xor (a rightrotate 13) xor (a rightrotate 22)
        maj := (a and b) xor (a and c) xor (b and c)
        temp2 := S0 + maj

        h := g
```

```
        g := f
        f := e
        e := d + temp1
        d := c
        c := b
        b := a
        a := temp1 + temp2

    h[0:7] += [a:h]

digest = h0 append h1 append h2 append h3 append h4 append h5 append h6 append h7
```

Credit: Wikipedia

(iii) **Chunk Matching**:

```
if shaResult in chunkDictionary:
    send(shaResult)
else:
    send(LZW(rawChunk))
```

(iv) **LZW Encoding**:

Credit: Wikipedia

(b)

(c)

(d)

(e)

4. (a)

(b)

(c)

(d)

(e)

5. (a)

(b)

(c)

(d)

(e)

(f)