CHAPTER 18

One-Way Hash Functions

Background

- A one-way hash function H(M), operates on an arbitrary-length pre-image message M and returns a fixed-length hash value h.

  h = H(M), where h is of length m.

- Many functions can take an arbitrary length input and return an output of fixed length, but one-way hash function have additional characteristics that make them one-way.

1. Given M, it is easy to compute h.
2. Given h, it is hard to compute M such that H(M)=h.
3. Given h, it is hard to find another message M`, such that H(M) = H(M`)

Length of One-Way Hash Functions

- Hash functions of a 64-bits are just too small to survive a birthday attack. Most practical hash function produce 128-bit hashes.
- This forces anyone attempting the birthday attack to hash 2^{64} random documents to find two that hash to the same value, not enough for lasting security.

- NIST, in its Secure Hash Standard(SHS) uses a 160-bit hash value.
- This makes the birthday attack even harder, required 2^{80} random hashes.
- The following method has been proposed to generate a longer hash value than a given hash function produces.

1. Generate a hash value of a message (using any one-way hash function).
2. Prepend the hash value to the message.
3. Generate the hash value of the concatenation of the message and the hash value.
4. Create a larger hash value consisting of the hash value generated in step (1) concatenated with the hash value generated at step(3).
5. Repeat steps (1) through (3) as many times as you wish, concatenating as you go.

- Although this method has never been proved to be either secure or insecure, various people have some serious reservations about it.

MD4

- MD4 is a one-way hash function designed by Ron Rivest.
- MD stands for Message Digest, the algorithm produces a 128-bit hash or message digest of the input message.

- Design goals of MD4 (outlines by Rivest):

1. Security: It is computationally infeasible to find two messages that hashed to the same value.
2. Direct Security: MD4's security is not based on any assumption, like the difficulty of factoring.

3. Speed: Suitable for high speed software implementation.
4. Simplicity and Compactness: MD4 is as simple as possible without large data structures or a complicated program.
5. Favor Little Endian Architecture: MD4 is optimized for microprocessor architecture specially for Intel Microprocessor.

MD5

- MD5 is an improved version of MD4 and more complex than MD4.
- It is similar in design and also produces a 128-bit hash.

Description of MD5:

- As an initial processing, the algorithm produce a block multiple of 512, of the input message.
- Each 512-bit is then divided into sixteen 32-bit sub-blocks.
- The output of the algorithm is a set of four 32-bit blocks, which concatenate to form a single 128-bit hash value.

- First the message is padded so that its length is just 64 bits short of being a multiple of 512.
- This padding is a single 1-bit added to the end of the message, followed by as many zeroes as are required.

MD5 Continue…

- Then a 64-bit representation of the message's length(before padding were added) is appended to the result.
- This ensures that different messages will not look the same after padding.
- MD5 has 4 rounds of 16 operations each.

- Each operation performs a nonlinear function on three of a, b, c and d. Then it adds the result to the fourth variable, a sub-block of the text and a constant. Then it rotates that result to the right a variable number of bits and adds the result to one of a, b, c, or d. Finally the result replaces one of a, b, c or d.

MD5 Continue…

If M_j represents the jth sub-block of the message (from 0 to 15), and <<<s represents a left circular shift of s bits, the four operations are:

(The FF(), GG(), HH() and II() operations corresponds to 1st, 2nd ,3rd, and 4th rounds respectively)

FF(a, b, c, d, M_j, s, t_i) denotes a = b + ((a + F(b, c, d) + M_j + t_i)<<<s)
GG(a, b, c, d, M_j, s, t_i) denotes a = b + ((a + G(b, c, d) + M_j + t_i)<<<s)
HH(a, b, c, d, M_j, s, t_i) denotes a = b + ((a + H(b, c, d) + M_j + t_i)<<<s)
II(a, b, c, d, M_j, s, t_i) denotes a = b + ((a + I(b, c, d) + M_j + t_i)<<<s)

The four nonlinear functions associated with each operations are as follows:

$$F(X, Y, Z) = (X \wedge Y) \vee (\neg X) \wedge Z)$$
$$G(X, Y, Z) = (X \wedge Z) \vee (Y \wedge (\neg X))$$
$$H(X, Y, Z) = X \oplus Y \oplus Z$$
$$I(X, Y, Z) = Y \oplus (X \vee (\neg Z)$$

MD5 Continue…

- Four 32-bit variables are initialized:

  A = 0x01234567

  B = 0x891bcdef

  C = 0xfedcba98

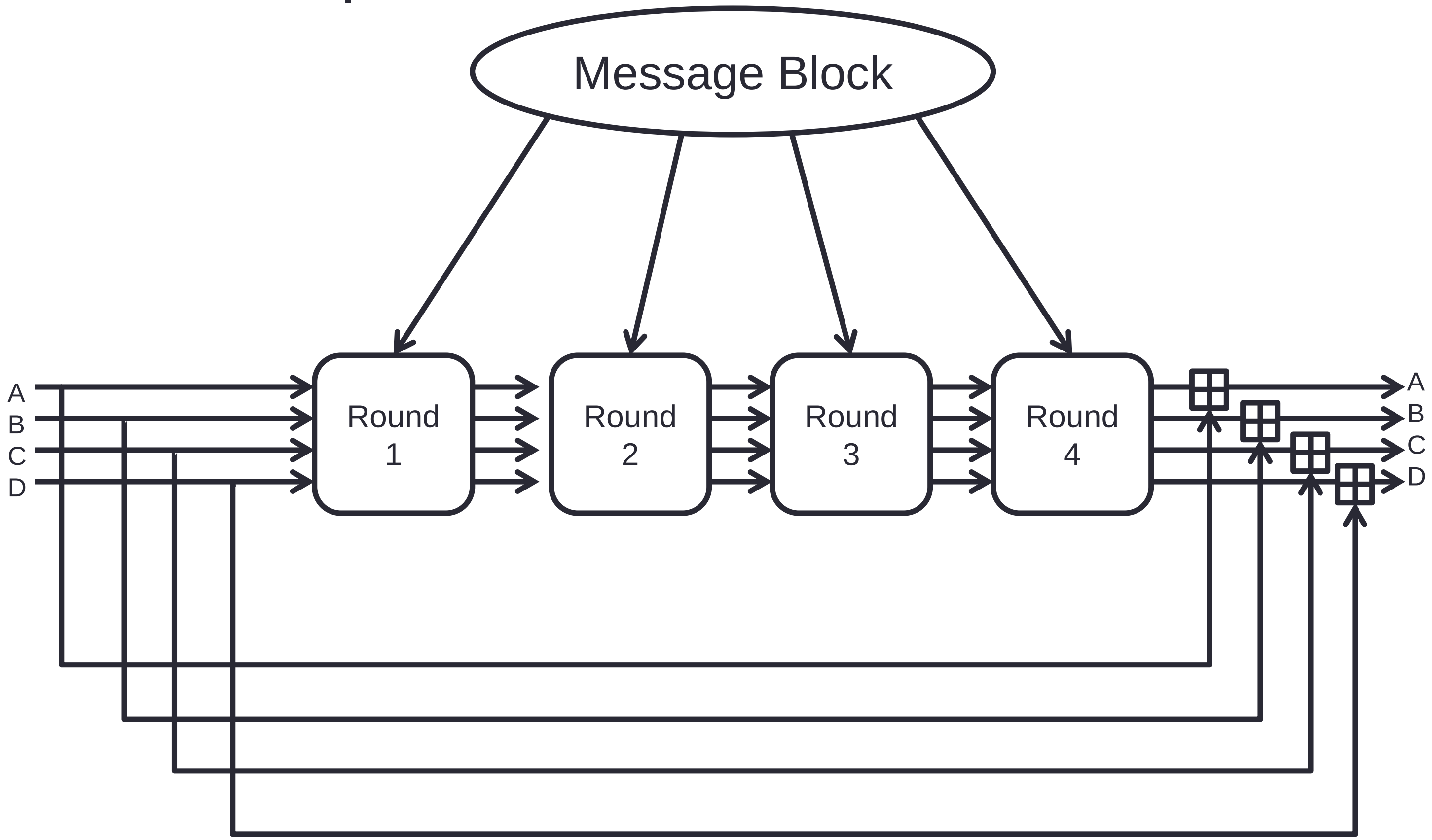
  D = 0x76543210

- These are called chaining variables. Now the main loop begins. The loop continues for as many 512-bit blocks as are in the message.

- The four variables are copied into different variables: a gets A, b gets B, c gets C, and d gets D.

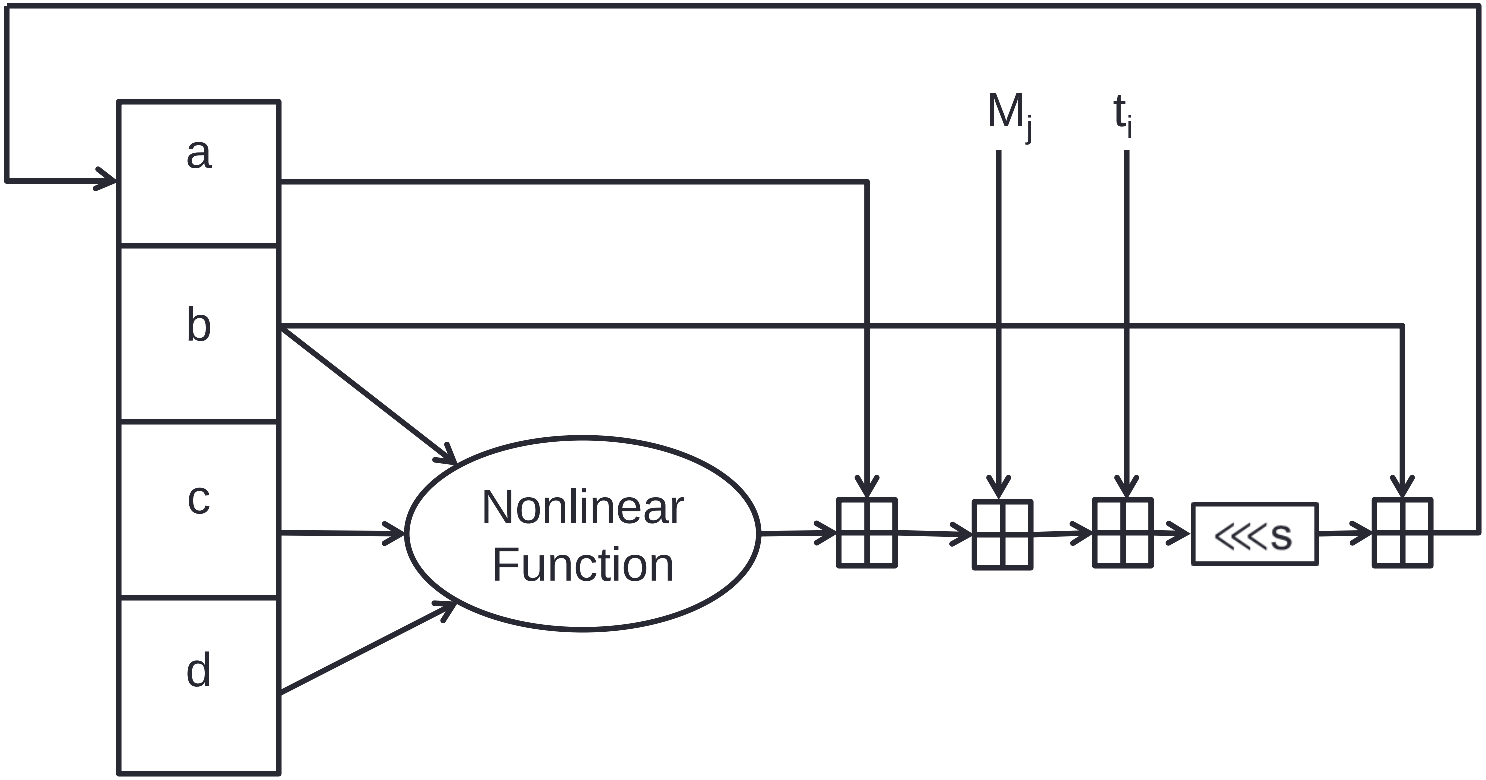
MD5 Continue…

MD5 main loop:



MD5 Continue…

- One MD5 operation:



MD5 Continues…

- The four rounds(64 steps) looks like:

Round 1:

FF(a, b, c, d, M_0, 7, 0xd76aa478)
FF(d, a, b, c, M_1, 12, 0xe8c7b756)
FF(c, d, a, b, M_2, 17, 0x242070db)
FF(b, c, d, a, M_3, 22, 0xclbdceee)
FF(a, b, c, d, M_4, 7, 0xf57c0faf)
FF(d, a, b, c, M_5, 12, 0x4787c62a)
FF(c, d, a, b, M_6, 17, 0xa8304613)
FF(b, c, d, a, M_7, 22, 0xfd469501)
FF(a, b, c, d, M_8, 7, 0x698098d8)
FF(d, a, b, c, M_9, 12, 0x8b44f7af)
FF(c, d, a, b, M_{10}, 17, 0xffff5bb1)
FF(b, c, d, a, M_{11}, 22, 0x895cd7be)
FF(a, b, c, d, M_{12}, 7, 0x6b901122)
FF(d, a, b, c, M_{13}, 12, 0xfd987193)
FF(c, d, a, b, M_{14}, 17, 0xa679438e)
FF(b, c, d, a, M_{15}, 22, 0x49b40821)

MD5 Continue…

Round 2:

- GG(a, b, c, d, M_1, 5, 0xf61e2562)
- GG(d, a, b, c, M_6, 9, 0xc040b340)
- GG(c, d, a, b, M_{11}, 14, 0x265e5a51)
- GG(b, c, d, a, M_0, 20, 0xe9b6c7aa)
- GG(a, b, c, d, M_5, 5, 0xd62f105d)
- GG(d, a, b, c, M_{10}, 9, 0x02441453)
- GG(c, d, a, b, M_{15}, 14, 0xd8a1e681)
- GG(b, c, d, a, M_4, 20, 0xe7d3fbc8)
- GG(a, b, c, d, M_9, 5, 0x21e1cde6)
- GG(d, a, b, c, M_{14}, 9, 0xc33707d6)
- GG(c, d, a, b, M_3, 14, 0xf4d50d87)
- GG(b, c, d, a, M_8, 20, 0x455a14ed)
- GG(a, b, c, d, M_{13}, 5, 0xa9e3e905)
- GG(d, a, b, c, M_2, 9, 0xfcefa3f8)
- GG(c, d, a, b, M_7, 14, 0x676f02d9)
- GG(b, c, d, a, M_{12}, 20, 0x8d3a4c8a)

MD5 Continue…

Round 3:

- HH(a, b, c, d, M_5, 4, 0xfffa3942)
- HH(d, a, b, c, M_8, 11, 0x8771f681)
- HH(c, d, a, b, M_{11}, 16, 0x6d9d6122)
- HH(b, c, d, a, M_{14}, 23, 0xfde5380c)
- HH(a, b, c, d, M_1, 4, 0xa4beea44)
- HH(d, a, b, c, M_4, 11, 0x4bdecfa9)
- HH(c, d, a, b, M_7, 16, 0xf6bb4b60)
- HH(b, c, d, a, M_{10}, 23, 0xbebfbc70)
- HH(a, b, c, d, M_{13}, 4, 0x289b7ec6)
- HH(d, a, b, c, M_0, 11, 0xeaa127fa)
- HH(c, d, a, b, M_3, 16, 0xd4ef3085)
- HH(b, c, d, a, M_6, 23, 0x04881d05)
- HH(a, b, c, d, M_9, 4, 0xd9d4d039)
- HH(d, a, b, c, M_{12},11, 0xe6db99e5)
- HH(c, d, a, b, M_{15}, 16, 0x1fa27cf8)
- HH(b, c, d, a, M_2, 23, 0xc4ac5665)

MD5 Continue…

Round 4:

II(a, b, c, d, M_0, 6, 0xf4292244)
II(d, a, b, c, M_7, 10, 0x432aff97)
II(c, d, a, b, M_{14}, 15, 0xab9423a7)
II(b, c, d, a, M_5, 21, 0xfc93a039)
II(a, b, c, d, M_{12}, 6, 0x655b59c3)
II(d, a, b, c, M_3, 10, 0x8f0ccc92)
II(c, d, a, b, M_{10}, 15, 0xffeff47d)
II(b, c, d, a, M_1, 21, 0x85845dd1)
II(a, b, c, d, M_8, 6, 0x6fa87e4f)
II(d, a, b, c, M_{15}, 10, 0xfe2ce6e0)
II(c, d, a, b, M_6, 15, 0xa3014314)
II(b, c, d, a, M_{13}, 21, 0x4e0811a1)
II(a, b, c, d, M_4, 6, 0xf7537e82)
II(d, a, b, c, M_{11},10, 0xbd3af235)
II(c, d, a, b, M_2, 15, 0x2ad7d2bb)
II(b, c, d, a, M_9, 21, 0xeb86d391)

MD5 Continue…

- Those t_i were chosen as follows:

- In step i, t_i is the integer part of 2^{32}*abs(sin(i)), where i is in radians.

- After all of this a, b, c and d are added to A, B, C and D respectively and the algorithm continues with the next block of data.

- The final output is the concatenation of A, B, C and D.

Secure Hash Algorithm(SHA)

- NIST along with the NSA, designed the Secure Hash Algorithm (SHA) for use with the Digital Signature Standard.
- When a message of any length<264 bits is input, the SHA produces a 160-bit output called a message digest.
- The MD is then input to the DSA, which computes the signature for the message.
- Signing the MD rather than the message often improves the efficiency of the process, because the MD is usually smaller than the message.
- The same MD should be obtained by the verifier of the signature when the received version of the message is used as input to SHA.

SHA Continue…

- Description of SHA: SHA produces a 160-bit hash, where as MD5 produces 128-bit hash.

- The message is padded to make it a multiple of 512 bits long. Same as MD5: First append a one, then as many zeros as necessary to make it 64-bits short of a multiple of 512, and finally a 64-bit representation of the length of the message before padding.

- Five 32-bit variables are initialized as follows:

  A = 0x67452301

  B = 0xefcdab89

  C = 0x98badcfe

  D =0x10325476

  E = 0xc3d2e1f0

SHA Continue…

- The main loop of the algorithm then begins.
- It processes the message 512 bits at a time and continues for as many 512-bit blocks as are in the message.

- First the five variables are copied into different variables: a gets A, b gets B, c gets C, d gets D, and e gets E.

- The main loop has four rounds of 20 operations each (MD5 has four rounds of 16 operations each).

- Each operation performs a nonlinear function on three of a, b, c, d and e, and then does shifting and adding similar to MD5.

SHA Continue…

- SHA's set of nonlinear function is:

  $f_t(X, Y, Z) = (X \wedge Y) \vee ((\neg X) \wedge Z)$, for t=0 to 19

  $f_t(X, Y, Z) = X \oplus Y \oplus Z$, for t=20 to 39

  $f_t(X, Y, Z) = (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z)$, for t=40 to 59

  $f_t(X, Y, Z) = X \oplus Y \oplus Z$, for t=60 to 79

- Four constants are used in the algorithm:

  K_t = 0x5a827999 for t = 0 to 19

  K_t = 0x6ed9eba1 for t = 20 to 39

  K_t = 0x8f1bbcdc for t = 40 to 59

  K_t = 0xca62c1d6 for t = 60 to 79

  These constants came from: 0x5a827999=$2^{1/2}/4$, 0x6ed9eba1=$3^{1/2}/4$, 0x8f1bbcdc=$5^{1/2}/4$ and 0xca62c1d6=$10^{1/2}/4$; all times 2^{32}.

SHA Continue…

- The message block is transformed from sixteen 32-bit words (M_0 to M_{15}) to eighty 32-bit words (W_0 to W_{79}) using the following algorithm:

  $W_t = M_t$, for t=0 to 15

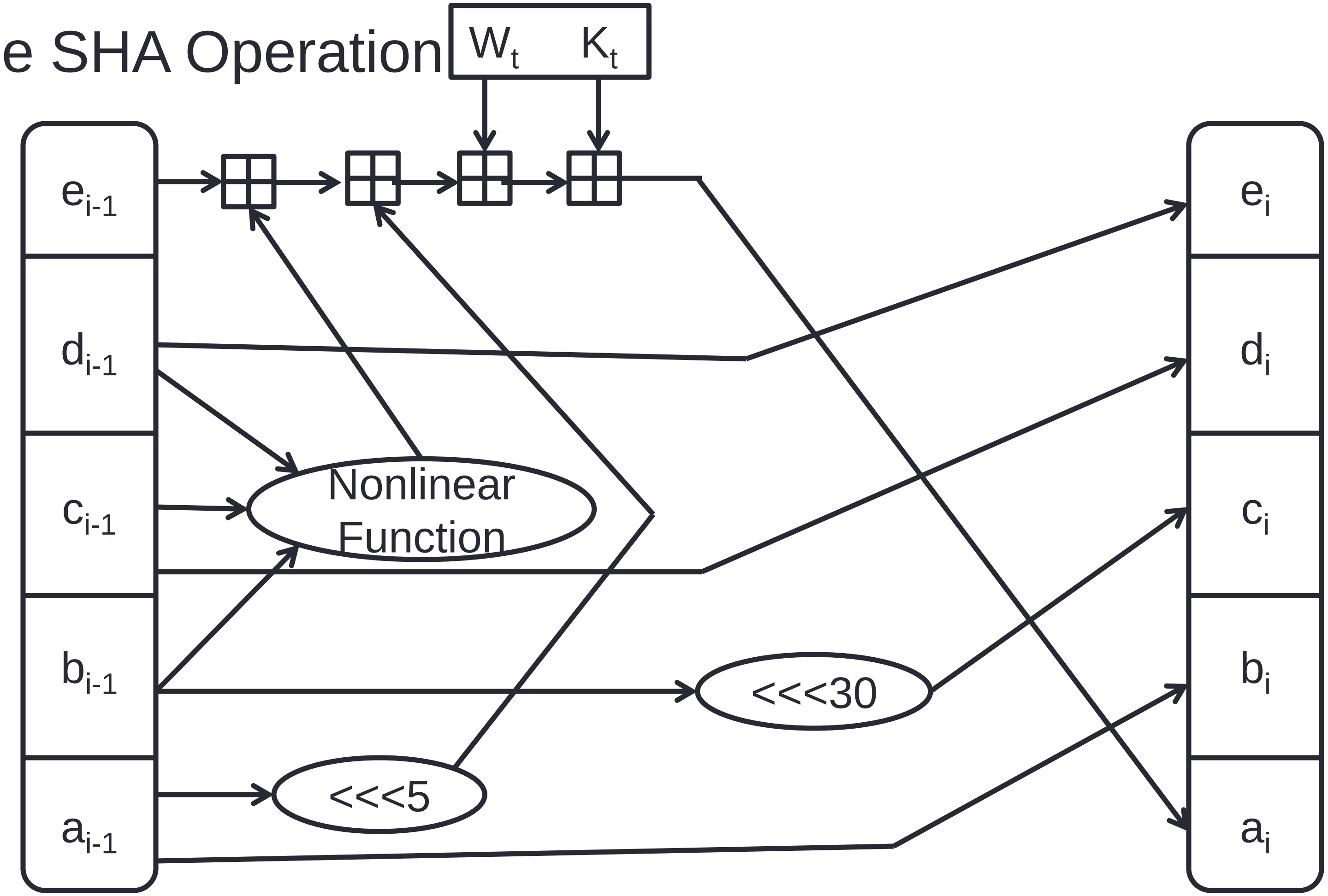
  $W_t = (W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) \lll 1$, for t=16 to 79.

- If t is the operation number (from 0 to 79), W_t represents the t^{th} sub-block of the expanded message, and <<<s represents a left circular shift of s bits, then the main loop look like:

```
For t=0 to 79
    TEMP = (a<<<5) + f_t(b, c, d) + e + W_t + K_t
    e = d
    d = c
    c = b<<<30
    b = a
    a = TEMP
```

SHA One Operation

One SHA Operation



SHA Continue…

- After all of this, a, b, c, d and e are added to A, B, C, D and E respectively and the algorithm continues with the next block (next 512 bits if any) of data.

- The final output is the concatenation of A, B, C, D and E.