**Implementation**

**Our next Goal was to start writing out our ideas on how we would be doing implementation for the project.**

**Step 1.Append padding bits**

**The input message is "padded" (extended) so that its length (in bits) equals to 448 mod 512. Padding is always performed, even if the length of the message is already 448 mod 512.**

**Padding is performed as follows: a single "1" bit is appended to the message, and then "0" bits are appended so that the length in bits of the padded message becomes congruent to 448 mod 512. At least one bit and at most 512 bits are appended.**

**Step 2. Append Length**

**Next we will had to decide how long Append length**

**A 64-bit representation of the length of the message is appended to the result of step1.**

**If the length of the message is greater than 2^64, only the low-order 64 bits will be used.**

**The resulting message (after padding with bits and with b) has a length that is an exact multiple of 512 bits. The input message will have a length that is an exact multiple of 16 (32-bit) words.**

**Step 3. Initialize MD buffer**

**Next we had to see how would we would want our encryption to display and what we wanted it to say with letters.**

**A four-word buffer (A, B, C, D) is used to compute the message digest. Each of A, B, C, D is a 32-bit register. These registers are initialized to the following values in hexadecimal, low-order bytes first):**

**word A: 01 23 45 67**

**word B: 89 ab cd ef**

**word C: fe dc ba 98**

**word D: 76 54 32 10**

**Step 4. Process message in 16-word blocks**

**We didn’t want our Process Message to be long so we decided to go simple and do 16- word block**

**Four functions will be defined such that each function takes an input of three 32-bit words and produces a 32-bit word output.**

**F (X, Y, Z) = XY or not (X) Z**

**G (X, Y, Z) = XZ or Y not (Z)**

**H (X, Y, Z) = X xor Y xor Z**

**I (X, Y, Z) = Y xor (X or not (Z))**

**Our First Round**

**Round 1.**

**[abcd k s i] denote the operation a = b + ((a + F (b, c, d) + X [k] + T [i]) <<< s).**

**Do the following 16 operations.**

**[ABCD 0 7 1] [DABC 1 12 2] [CDAB 2 17 3] [BCDA 3 22 4]**

**[ABCD 4 7 5] [DABC 5 12 6] [CDAB 6 17 7] [BCDA 7 22 8]**

**[ABCD 8 7 9] [DABC 9 12 10] [CDAB 10 17 11] [BCDA 11 22 12]**

**[ABCD 12 7 13] [DABC 13 12 14] [CDAB 14 17 15] [BCDA 15 22 16]**

**Summary**

* **Comparing to other digest algorithms, MD5 is simple to implement, and provides a "fingerprint" or message digest of a message of arbitrary length.**
* **It performs very fast on 32-bit machine.**
* **MD5 is being used heavily from large corporations, such as IBM, Cisco Systems, to individual programmers.**
* **MD5 is considered one of the most efficient algorithms currently available.**