



Implement Secure Hashing Algorithm – 512 (SHA-512) as Functional Programming Paradigm

Read

Discuss

Courses

Practice

Given a string **S** of length **N**, the task is to find the SHA-512 Hash Value of the given string **S**.

Examples:

Input: *S = "GeeksforGeeks"*

Output:

*acc10c4e0b38617f59e88e49215e2e894afaee5ec948c2af6f44039f03c9fe47a9210e01d5cd926c142bd
c9179c2ad30f927a8faf69421ff60a5eaddcf8cb9c*

Input: *S = "hello world"*

Output:

*309ecc489c12d6eb4cc40f50c902f2b4d0ed77ee511a7c7a9bcd3ca86d4cd86f989dd35bc5ff499670da3
4255b45b0cfd830e81f605dcf7dc5542e93ae9cd76f*



Approach: Follow the steps below to solve the problem:

- Convert the given string into the binary form.
- Append '1' to the string and then '0' continuously until length of the string is $< (N\%(1024 - 128))$.
- Add the 128-bit binary representation of N in the string S .
- Find the number of chunks of the size of **1024** and store it in a variable, say **chunks** as $N/1024$.
- Divide the string S into **16** chunks of **64** characters.
- Extend the number of chunks to **80** by performing the following operations:
 - Iterate over the range [16, 80] and then find 4 values say **WordA**, **WordB**, **WordC**, **WordD** as:
 - $\text{WordA} = \text{rotate_right}(\text{Message}[g - 2], 19) \wedge \text{rotate_right}(\text{Message}[g - 2], 61) \wedge \text{shift_right}(\text{Message}[g - 2], 6)$.
 - $\text{WordB} = \text{Message}[g - 7]$.
 - $\text{WordC} = \text{rotate_right}(\text{Message}[g - 15], 1) \wedge \text{rotate_right}(\text{Message}[g - 15], 8) \wedge \text{shift_right}(\text{Message}[g - 15], 7)$.
 - $\text{WordD} = \text{Message}[g - 16]$.
 - Update the value of $\text{Message}[g]$ as $(\text{WordA} + \text{WordB} + \text{WordC} + \text{WordD})$.
- Initialize **8** variables say **A**, **B**, **C**, **D**, **E**, **F**, **G**, **H** of type 64-bit to store the final hash value of the given string S .
- Traverse the array $\text{Block}[]$ and perform the following steps:
 - Update the value of **A**, **B**, **C**, **D**, **E**, **F**, **G**, **H** using the Hash Function till 80 iterations by rotating one by one.
 - Now, update the value of **A**, **B**, **C**, **D**, **E**, **F**, **G**, **H** by the summation of previous values of **A**, **B**, **C**, **D**, **E**, **F**, **G**.

- After completing the above steps, print the [hexadecimal values](#) of A, B, C, D, E, F, G, H to get the Hash Value of the given string.

Below is the implementation of the above approach:

C++14

```
// C++ program for the above approach
#include <bits/stdc++.h>
using namespace std;
typedef unsigned long long int int64;

int64 Message[80];

// Stores the hexadecimal values for
// calculating hash values
const int64 Constants[80]
= { 0x428a2f98d728ae22, 0x7137449123ef65cd,
    0xb5c0fbcfec4d3b2f, 0xe9b5dba58189dbbc,
    0x3956c25bf348b538, 0x59f111f1b605d019,
    0x923f82a4af194f9b, 0xab1c5ed5da6d8118,
    0xd807aa98a3030242, 0x12835b0145706fbe,
    0x243185be4ee4b28c, 0x550c7dc3d5ffb4e2,
    0x72be5d74f27b896f, 0x80deb1fe3b1696b1,
    0x9bdc06a725c71235, 0xc19bf174cf692694,
    0xe49b69c19ef14ad2, 0xefbe4786384f25e3,
    0x0fc19dc68b8cd5b5, 0x240ca1cc77ac9c65,
    0x2de92c6f592b0275, 0x4a7484aa6ea6e483,
    0x5cb0a9dcbbd41fbd4, 0x76f988da831153b5,
    0x983e5152ee66dfab, 0xa831c66d2db43210,
    0xb00327c898fb213f, 0xbf597fc7beef0ee4,
    0xc6e00bf33da88fc2, 0xd5a79147930aa725,
    0x06ca6351e003826f, 0x142929670a0e6e70,
    0x27b70a8546d22ffc, 0x2e1b21385c26c926,
```

```
0xa2bfe8a14cf10364, 0xa81a664bbc423001,  
0xc24b8b70d0f89791, 0xc76c51a30654be30,  
0xd192e819d6ef5218, 0xd69906245565a910,  
0xf40e35855771202a, 0x106aa07032bbd1b8,  
0x19a4c116b8d2d0c8, 0x1e376c085141ab53,  
0x2748774cdf8eeb99, 0x34b0bcb5e19b48a8,  
0x391c0cb3c5c95a63, 0x4ed8aa4ae3418acb,  
0x5b9cca4f7763e373, 0x682e6ff3d6b2b8a3,  
0x748f82ee5defb2fc, 0x78a5636f43172f60,  
0x84c87814a1f0ab72, 0x8cc702081a6439ec,  
0x90beffffa23631e28, 0xa4506cebde82bde9,  
0xbef9a3f7b2c67915, 0xc67178f2e372532b,  
0xca273eceeaa26619c, 0xd186b8c721c0c207,  
0xeada7dd6cde0eb1e, 0xf57d4f7fee6ed178,  
0x06f067aa72176fba, 0x0a637dc5a2c898a6,  
0x113f9804bef90dae, 0x1b710b35131c471b,  
0x28db77f523047d84, 0x32caab7b40c72493,  
0x3c9ebe0a15c9bebc, 0x431d67c49c100d4c,  
0x4cc5d4becb3e42b6, 0x597f299cfc657e2a,  
0x5fcb6fab3ad6faec, 0x6c44198c4a475817 };
```

```
// Function to convert a binary string  
// to hexa-decimal value  
string gethex(string bin)  
{  
    if (bin == "0000")  
        return "0";  
    if (bin == "0001")  
        return "1";  
    if (bin == "0010")  
        return "2";  
    if (bin == "0011")  
        return "3";  
    if (bin == "0100")  
        return "4";  
    if (bin == "0101")  
        return "5";  
}
```

```

    if (bin == "0111")
        return "7";
    if (bin == "1000")
        return "8";
    if (bin == "1001")
        return "9";
    if (bin == "1010")
        return "a";
    if (bin == "1011")
        return "b";
    if (bin == "1100")
        return "c";
    if (bin == "1101")
        return "d";
    if (bin == "1110")
        return "e";
    if (bin == "1111")
        return "f";
}

// Function to convert a decimal value
// to hexa decimal value
string decimaltohex(int64 deci)
{
    // Stores the value as string
    string EQBIN = bitset<64>(deci).to_string();

    // Stores the equivalent hexa decimal
    string hexstring = "";
    string temp;

    // Traverse the string EQBIN
    for (unsigned int i = 0;
         i < EQBIN.length(); i += 4) {
        temp = EQBIN.substr(i, 4);
        hexstring += gethex(temp);
    }
}

```

```

    return hexstring;
}

// Function to convert a binary
// string to decimal value
int64 BintDec(string bin)
{
    int64 value = bitset<64>(bin)
                    .to_ullong();

    return value;
}

// Function to right rotate x by n bits
int64 rotate_right(int64 x, int n)
{
    return (x >> n) | (x << (64 - n));
}

// Function to right shift x by n bits
int64 shift_right(int64 x, int n)
{
    return (x >> n);
}

// Function to divide the string
// into chunks
void separator(string getBlock)
{
    // Stores the size of chunks
    int chunknum = 0;

    // Traverse the string S
    for (unsigned int i = 0;
         i < getBlock.length();
         i += 64, ++chunknum) {

```

```

        = BintDec(getBlock.substr(i, 64));
    }

    // Iterate over the range [16, 80]
    for (int g = 16; g < 80; ++g) {

        // Find the WordA
        int64 WordA = rotate_right(Message[g - 2], 19)
            ^ rotate_right(Message[g - 2], 61)
            ^ shift_right(Message[g - 2], 6);

        // Find the WordB
        int64 WordB = Message[g - 7];

        // Find the WordC
        int64 WordC = rotate_right(Message[g - 15], 1)
            ^ rotate_right(Message[g - 15], 8)
            ^ shift_right(Message[g - 15], 7);

        // Find the WordD
        int64 WordD = Message[g - 16];

        // Find the resultant code
        int64 T = WordA + WordB + WordC + WordD;

        // Return the resultant Hash Code
        Message[g] = T;
    }
}

// Function to find the major of a, b, c
int64 maj(int64 a, int64 b, int64 c)
{
    return (a & b) ^ (b & c) ^ (c & a);
}

// Function to find the ch value of a.

```

```

{
    return (e & f) ^ (~e & g);
}

// Function to find the Bitwise XOR with
// the right rotate over 14, 18, and 41
int64 sigmaE(int64 e)
{
    // Return the resultant value
    return rotate_right(e, 14)
        ^ rotate_right(e, 18)
        ^ rotate_right(e, 41);
}

// Function to find the Bitwise XOR with
// the right rotate over 28, 34, and 39
int64 sigmaA(int64 a)
{
    // Return the resultant value
    return rotate_right(a, 28)
        ^ rotate_right(a, 34)
        ^ rotate_right(a, 39);
}

// Function to generate the hash code
void Func(int64 a, int64 b, int64 c,
          int64& d, int64 e, int64 f,
          int64 g, int64& h, int K)
{
    // Find the Hash Code
    int64 T1 = h + Ch(e, f, g) + sigmaE(e) + Message[K]
        + Constants[K];
    int64 T2 = sigmaA(a) + maj(a, b, c);

    d = d + T1;
    h = T1 + T2;
}

```



```

// Function to convert the hash value
// of a given string
string SHA512(string myString)
{
    // Stores the 8 blocks of size 64
    int64 A = 0x6a09e667f3bcc908;
    int64 B = 0xbb67ae8584caa73b;
    int64 C = 0x3c6ef372fe94f82b;
    int64 D = 0xa54ff53a5f1d36f1;
    int64 E = 0x510e527fade682d1;
    int64 F = 0x9b05688c2b3e6c1f;
    int64 G = 0x1f83d9abfb41bd6b;
    int64 H = 0x5be0cd19137e2179;

    int64 AA, BB, CC, DD, EE, FF, GG, HH;

    stringstream fixedstream;

    // Traverse the string S
    for (int i = 0;
        i < myString.size(); ++i) {

        // Add the character to stream
        fixedstream << bitset<8>(myString[i]);
    }

    // Stores string of size 1024
    string s1024;

    // Stores the string in the
    // fixedstream
    s1024 = fixedstream.str();

    // Stores the length of string
    int orilen = s1024.length();
    int tobeadded;

```

```

// If 1024-128 is greater than modded
if (1024 - modded >= 128) {
    tobeadded = 1024 - modded;
}

// Else if 1024-128 is less than modded
else if (1024 - modded < 128) {
    tobeadded = 2048 - modded;
}

// Append 1 to string
s1024 += "1";

// Append tobeadded-129 zeros
// in the string
for (int y = 0; y < tobeadded - 129; y++) {
    s1024 += "0";
}

// Stores the binary representation
// of string length
string lengthbits
    = std::bitset<128>(orilen).to_string();

// Append the lengthbits to string
s1024 += lengthbits;

// Find the count of chunks of
// size 1024 each
int blocksnumber = s1024.length() / 1024;

// Stores the numbering of chunks
int chunknum = 0;

// Stores hash value of each blocks
string Blocks[blocksnumber];

```

```

for (int i = 0; i < s1024.length();
    i += 1024, ++chunknum) {
    Blocks[chunknum] = s1024.substr(i, 1024);
}

```

```

// Traverse the array Blocks[]
for (int letsgo = 0;
    letsgo < blocksnumber;
    ++letsgo) {

    // Divide the current string
    // into 80 blocks size 16 each
    separator(Blocks[letsgo]);

```

```

AA = A;
BB = B;
CC = C;
DD = D;
EE = E;
FF = F;
GG = G;
HH = H;

```

```

int count = 0;

```

```

// Find hash values
for (int i = 0; i < 10; i++) {

```

```

    // Find the Hash Values

```

```

    Func(A, B, C, D, E, F, G, H, count);
    count++;
    Func(H, A, B, C, D, E, F, G, count);
    count++;
    Func(G, H, A, B, C, D, E, F, count);
    count++;
    Func(F, G, H, A, B, C, D, E, count);

```

```

        count++;
        Func(D, E, F, G, H, A, B, C, count);
        count++;
        Func(C, D, E, F, G, H, A, B, count);
        count++;
        Func(B, C, D, E, F, G, H, A, count);
        count++;
    }

    // Update the value of A, B, C,
    // D, E, F, G, H

    A += AA;
    B += BB;
    C += CC;
    D += DD;
    E += EE;
    F += FF;
    G += GG;
    H += HH;
}

stringstream output;

// Print the hexadecimal value of
// strings as the resultant SHA-512
output << decimaltohex(A);
output << decimaltohex(B);
output << decimaltohex(C);
output << decimaltohex(D);
output << decimaltohex(E);
output << decimaltohex(F);
output << decimaltohex(G);
output << decimaltohex(H);

// Return the string
return output.str();

```

```
// Driver Code
int main()
{
    // Input
    string S = "GeeksForGeeks";

    // Function Call
    cout << S << ": " << SHA512(S);

    return 0;
}
```

Output:

GeeksForGeeks:

0acc10c4e0b38617f59e88e49215e2e894afaee5ec948c2af6f44039f03c9fe47a9210e01d5cd926c142bdc9179c
2ad30f927a8faf69421ff60a5eaddcf8cb9c

Time Complexity: $O(N)$








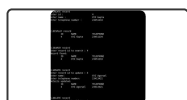
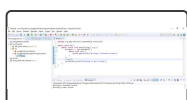
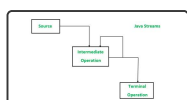
Auxiliary Space: $O(1)$

Feeling lost in the world of random DSA topics, wasting time without progress? It's time for a change! Join our DSA course, where we'll guide you on an exciting journey to master DSA efficiently and on schedule.

Ready to dive in? Explore our Free Demo Content and join our DSA course, trusted by over 100,000 geeks!

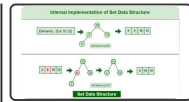
- [DSA in C++](#)
- [DSA in Java](#)

Similar Reads

 Bug in SHA-512 Hash Generation Java code	 Difference between Secure Socket Layer (SSL) and Secure Electronic Transaction (SET)
 SHA in Python	 SHA-384 Hash In Java
 SHA-224 Hash In Java	 SHA-256 Hash in Java
 SHA-1 Hash	 Implement Phone Directory using Hashing
 Functional Programming in Java with Examples	 Functional Programming in Java 8+ using the Stream API with Example

Related Tutorials

 Cryptography Tutorial	 Mathematical and Geometric Algorithms - Data Structure and Algorithm Tutorials
 Learn Data Structures with Javascript DSA	 Introduction to Max-Heap – Data Structure



Introduction to Set – Data Structure and Algorithm Tutorials

[Previous](#)

[Next](#)

Queries to find the count of characters preceding the given location

Minimum time required to complete all tasks without altering their order

Article Contributed By :



therain0605

therain0605

Vote for difficulty

Current difficulty : [Medium](#)

Easy

Normal

Medium

Hard

Expert

Improved By : [simranarora5sos](#), [anikakapoor](#), [sumitgumber28](#), [rkbhola5](#), [mitalibhola94](#)

Article Tags : [cryptography](#) , [Java-Functional Programming](#) , [Algorithms](#) , [Computer Networks](#) , [DSA](#) , [Hash](#)

Practice Tags : [Algorithms](#), [Hash](#)

[Improve Article](#)

[Report Issue](#)



A-143, 9th Floor, Sovereign Corporate
Tower, Sector-136, Noida, Uttar Pradesh -
201305

feedback@geeksforgeeks.org



Company

[About Us](#)

[Legal](#)

Explore

[Job-A-Thon Hiring](#)

[Challenge](#)

[Geekshow](#)

Languages

[Python](#)

[Java](#)

DSA Concepts

[Data Structures](#)

[Arrays](#)

DSA Roadmaps

[DSA for Beginners](#)

[Basic DSA Coding](#)

[Roadmap](#)

Web Development

[HTML](#)

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

In Media	Offline Classes (Delhi/NCR)	GoLang	Algorithms	DSA Roadmap by Sandeep Jain	Bootstrap
Contact Us	DSA in JAVA/C++	SQL	Searching	DSA with JavaScript	ReactJS
Advertise with us	Master System Design	R Language	Sorting	Top 100 DSA Interview Problems	AngularJS
GFG Corporate Solution	Master CP	Android Tutorial	Mathematical Dynamic Programming	All Cheat Sheets	NodeJS
Placement Training Program	GeeksforGeeks Videos				Express.js
Apply for Mentor					Lodash

Computer Science	Python	Data Science & ML	DevOps	Competitive Programming	System Design
GATE CS Notes	Python Programming Examples	Data Science With Python	Git	Top DSA for CP	What is System Design
Operating Systems	Django Tutorial	Data Science For Beginner	AWS	Top 50 Tree Problems	Monolithic and Distributed SD
Computer Network	Python Projects	Machine Learning Tutorial	Kubernetes	Top 50 Graph Problems	Scalability in SD
Database Management System	Python Tkinter	Maths For Machine Learning	Azure	Top 50 Array Problems	Databases in SD
Software Engineering	OpenCV Python Tutorial	Pandas Tutorial	GCP	Top 50 String Problems	High Level Design or HLD
Digital Logic Design	Python Interview Question	NumPy Tutorial		Top 50 DP Problems	Low Level Design or LLD
Engineering Maths		NLP Tutorial		Top 15 Websites for CP	Crack System Design Round
		Deep Learning Tutorial			System Design Interview Questions

Interview Corner	GfG School	Commerce	UPSC	SSC/ BANKING	Write & Earn
Company Wise	CBSE Notes for Class 8	Accountancy	Polity Notes	SSC CGL Syllabus	Write an Article

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#)

Preparation for SDE	CBSE Notes for Class	Economics	History Notes	SBI Clerk Syllabus	Pick Topics to Write
Experienced	10	Human Resource	Science and	IBPS PO Syllabus	Share your
Interviews	CBSE Notes for Class	Management (HRM)	Technology Notes	IBPS Clerk Syllabus	Experiences
Internship Interviews	11	Management	Economics Notes	Aptitude Questions	Internships
Competitive	CBSE Notes for Class	Income Tax	Important Topics in	SSC CGL Practice	
Programming	12	Finance	Ethics	Papers	
Aptitude Preparation	English Grammar	Statistics for	UPSC Previous Year		
		Economics	Papers		

@GeeksforGeeks, Sanchhaya Education Private Limited, All rights reserved