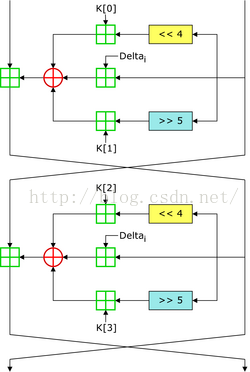
[TEA、XTEA、XXTEA加密解密算法](http://blog.csdn.net/gsls200808/article/details/48243019)

在密码学中，微型加密算法（Tiny Encryption Algorithm，TEA）是一种易于描述和执行的块密码，通常只需要很少的代码就可实现。其设计者是剑桥大学计算机实验室的大卫·惠勒与罗杰·尼达姆。这项技术最初于1994年提交给鲁汶的快速软件加密的研讨会上，并在该研讨会上演讲中首次发表。

在给出的代码中：加密使用的数据为2个32位无符号整数，密钥为4个32位无符号整数即密钥长度为128位

加密过程：



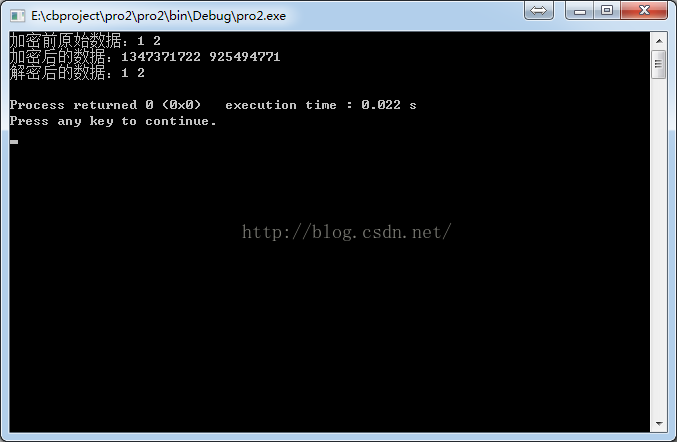
算法实现：

示例代码：  
C语言代码（需支持C99）

**[cpp]** [view plain](http://blog.csdn.net/gsls200808/article/details/48243019) [copy](http://blog.csdn.net/gsls200808/article/details/48243019)

1. #include <stdio.h>
2. #include <stdint.h>
4. //加密函数
5. **void** encrypt (uint32\_t\* v, uint32\_t\* k) {
6. uint32\_t v0=v[0], v1=v[1], sum=0, i;           /\* set up \*/
7. uint32\_t delta=0x9e3779b9;                     /\* a key schedule constant \*/
8. uint32\_t k0=k[0], k1=k[1], k2=k[2], k3=k[3];   /\* cache key \*/
9. **for** (i=0; i < 32; i++) {                       /\* basic cycle start \*/
10. sum += delta;
11. v0 += ((v1<<4) + k0) ^ (v1 + sum) ^ ((v1>>5) + k1);
12. v1 += ((v0<<4) + k2) ^ (v0 + sum) ^ ((v0>>5) + k3);
13. }                                              /\* end cycle \*/
14. v[0]=v0; v[1]=v1;
15. }
16. //解密函数
17. **void** decrypt (uint32\_t\* v, uint32\_t\* k) {
18. uint32\_t v0=v[0], v1=v[1], sum=0xC6EF3720, i;  /\* set up \*/
19. uint32\_t delta=0x9e3779b9;                     /\* a key schedule constant \*/
20. uint32\_t k0=k[0], k1=k[1], k2=k[2], k3=k[3];   /\* cache key \*/
21. **for** (i=0; i<32; i++) {                         /\* basic cycle start \*/
22. v1 -= ((v0<<4) + k2) ^ (v0 + sum) ^ ((v0>>5) + k3);
23. v0 -= ((v1<<4) + k0) ^ (v1 + sum) ^ ((v1>>5) + k1);
24. sum -= delta;
25. }                                              /\* end cycle \*/
26. v[0]=v0; v[1]=v1;
27. }
29. **int** main()
30. {
31. uint32\_t v[2]={1,2},k[4]={2,2,3,4};
32. // v为要加密的数据是两个32位无符号整数
33. // k为加密解密密钥，为4个32位无符号整数，即密钥长度为128位
34. printf("加密前原始数据：%u %u\n",v[0],v[1]);
35. encrypt(v, k);
36. printf("加密后的数据：%u %u\n",v[0],v[1]);
37. decrypt(v, k);
38. printf("解密后的数据：%u %u\n",v[0],v[1]);
39. **return** 0;
40. }

执行结果：

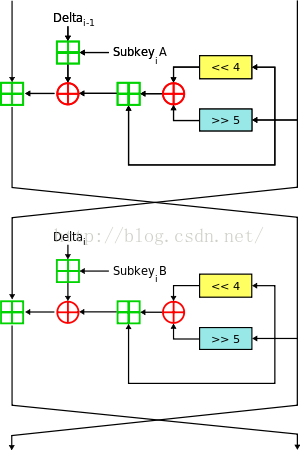
http://write.blog.csdn.net/postedithttp://write.blog.csdn.net/postedit

**[plain]** [view plain](http://blog.csdn.net/gsls200808/article/details/48243019) [copy](http://blog.csdn.net/gsls200808/article/details/48243019)

1. 加密前原始数据：1 2
2. 加密后的数据：1347371722 925494771
3. 解密后的数据：1 2
5. Process returned 0 (0x0)   execution time : 0.020 s
6. Press any key to continue.

XTEA是TEA的升级版，增加了更多的密钥表，移位和异或操作等等，设计者是Roger Needham, David Wheeler

加密过程：

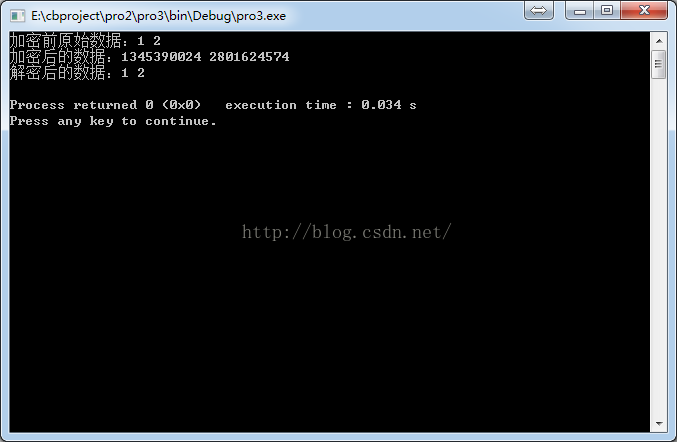


算法实现：

示例代码：

**[cpp]** [view plain](http://blog.csdn.net/gsls200808/article/details/48243019) [copy](http://blog.csdn.net/gsls200808/article/details/48243019)

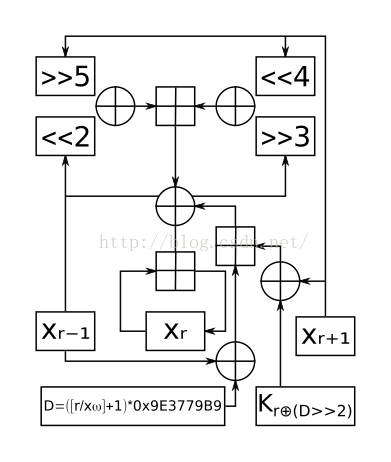
1. #include <stdio.h>
2. #include <stdint.h>
4. /\* take 64 bits of data in v[0] and v[1] and 128 bits of key[0] - key[3] \*/
6. **void** encipher(unsigned **int** num\_rounds, uint32\_t v[2], uint32\_t **const** key[4]) {
7. unsigned **int** i;
8. uint32\_t v0=v[0], v1=v[1], sum=0, delta=0x9E3779B9;
9. **for** (i=0; i < num\_rounds; i++) {
10. v0 += (((v1 << 4) ^ (v1 >> 5)) + v1) ^ (sum + key[sum & 3]);
11. sum += delta;
12. v1 += (((v0 << 4) ^ (v0 >> 5)) + v0) ^ (sum + key[(sum>>11) & 3]);
13. }
14. v[0]=v0; v[1]=v1;
15. }
17. **void** decipher(unsigned **int** num\_rounds, uint32\_t v[2], uint32\_t **const** key[4]) {
18. unsigned **int** i;
19. uint32\_t v0=v[0], v1=v[1], delta=0x9E3779B9, sum=delta\*num\_rounds;
20. **for** (i=0; i < num\_rounds; i++) {
21. v1 -= (((v0 << 4) ^ (v0 >> 5)) + v0) ^ (sum + key[(sum>>11) & 3]);
22. sum -= delta;
23. v0 -= (((v1 << 4) ^ (v1 >> 5)) + v1) ^ (sum + key[sum & 3]);
24. }
25. v[0]=v0; v[1]=v1;
26. }
28. **int** main()
29. {
30. uint32\_t v[2]={1,2};
31. uint32\_t **const** k[4]={2,2,3,4};
32. unsigned **int** r=32;//num\_rounds建议取值为32
33. // v为要加密的数据是两个32位无符号整数
34. // k为加密解密密钥，为4个32位无符号整数，即密钥长度为128位
35. printf("加密前原始数据：%u %u\n",v[0],v[1]);
36. encipher(r, v, k);
37. printf("加密后的数据：%u %u\n",v[0],v[1]);
38. decipher(r, v, k);
39. printf("解密后的数据：%u %u\n",v[0],v[1]);
40. **return** 0;
41. }



**[plain]** [view plain](http://blog.csdn.net/gsls200808/article/details/48243019) [copy](http://blog.csdn.net/gsls200808/article/details/48243019)

1. 加密前原始数据：1 2
2. 加密后的数据：1345390024 2801624574
3. 解密后的数据：1 2
5. Process returned 0 (0x0)   execution time : 0.034 s
6. Press any key to continue.

XXTEA，又称Corrected Block TEA，是XTEA的升级版，设计者是Roger Needham, David Wheeler  
加密过程：



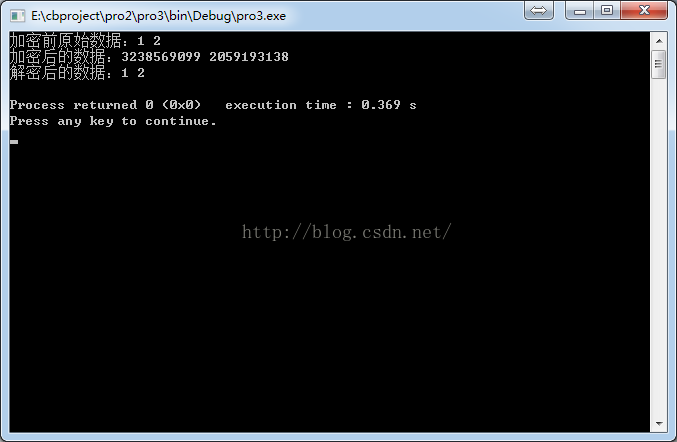
算法实现：

示例代码：

**[cpp]** [view plain](http://blog.csdn.net/gsls200808/article/details/48243019) [copy](http://blog.csdn.net/gsls200808/article/details/48243019)

1. #include <stdio.h>
2. #include <stdint.h>
3. #define DELTA 0x9e3779b9
4. #define MX (((z>>5^y<<2) + (y>>3^z<<4)) ^ ((sum^y) + (key[(p&3)^e] ^ z)))
6. **void** btea(uint32\_t \*v, **int** n, uint32\_t **const** key[4])
7. {
8. uint32\_t y, z, sum;
9. unsigned p, rounds, e;
10. **if** (n > 1)            /\* Coding Part \*/
11. {
12. rounds = 6 + 52/n;
13. sum = 0;
14. z = v[n-1];
15. **do**
16. {
17. sum += DELTA;
18. e = (sum >> 2) & 3;
19. **for** (p=0; p<n-1; p++)
20. {
21. y = v[p+1];
22. z = v[p] += MX;
23. }
24. y = v[0];
25. z = v[n-1] += MX;
26. }
27. **while** (--rounds);
28. }
29. **else** **if** (n < -1)      /\* Decoding Part \*/
30. {
31. n = -n;
32. rounds = 6 + 52/n;
33. sum = rounds\*DELTA;
34. y = v[0];
35. **do**
36. {
37. e = (sum >> 2) & 3;
38. **for** (p=n-1; p>0; p--)
39. {
40. z = v[p-1];
41. y = v[p] -= MX;
42. }
43. z = v[n-1];
44. y = v[0] -= MX;
45. sum -= DELTA;
46. }
47. **while** (--rounds);
48. }
49. }

52. **int** main()
53. {
54. uint32\_t v[2]= {1,2};
55. uint32\_t **const** k[4]= {2,2,3,4};
56. **int** n= 2; //n的绝对值表示v的长度，取正表示加密，取负表示解密
57. // v为要加密的数据是两个32位无符号整数
58. // k为加密解密密钥，为4个32位无符号整数，即密钥长度为128位
59. printf("加密前原始数据：%u %u\n",v[0],v[1]);
60. btea(v, n, k);
61. printf("加密后的数据：%u %u\n",v[0],v[1]);
62. btea(v, -n, k);
63. printf("解密后的数据：%u %u\n",v[0],v[1]);
64. **return** 0;
65. }



**[plain]** [view plain](http://blog.csdn.net/gsls200808/article/details/48243019) [copy](http://blog.csdn.net/gsls200808/article/details/48243019)

1. 加密前原始数据：1 2
2. 加密后的数据：3238569099 2059193138
3. 解密后的数据：1 2
5. Process returned 0 (0x0)   execution time : 0.369 s
6. Press any key to continue.