

<Final project>

Bea Jae Kyeong, Esmanur Karaca

Index

- The main purpose of the project
- Regulations
- Implementations of the code
- The following result
- Application of corrections from errors by cppcheck
- The reduction of the cyclomatic complexity by metriculator

The main purpose

The main purpose of our project is to apply of SEI CERT Coding Standards to check if our program is well-coded. This is to make our program's quality higher. If the code of the program doesn't meet the standard of SEI CERT C++ Coding Standard, we modified the code.

We used Cppcheck tool designed for C/C++ for this practice. Its main goal is to detect the kinds of errors that a compiler usually can't detect.

Regulations

We chose 10 regulations from the standards, which are :

1. [FIO51-CPP. Close files when they are no longer needed](#)
2. [FIO38-C. Do not copy a FILE object](#)
3. [FIO45-C. Avoid TOCTOU race conditions while accessing files](#)
4. [FIO46-C. Do not access a closed file](#)
5. [DCL51-CPP. Do not declare or define a reserved identifier](#)
6. [FLP30-C. Do not use floating-point variables as loop counters](#)
7. [STR34-C. Cast characters to unsigned char before converting to larger integer sizes](#)
8. [ARR30-C. Do not form or use out-of-bounds pointers or array subscripts](#)

9. [DCL40-C. Do not create incompatible declarations of the same function or object](#)
10. [ERR33-C. Detect and handle standard library errors](#)

FIO is from the Rule 07. Input Output (FIO), DCL from Declarations and Initialization (DCL) Rule 01, MSC from Rule 49. Miscellaneous (MSC), STR from Rule 05. Characters and Strings (STR), CTR from Rule 04. Containers (CTR), DCL from Rule 01. Declarations and Initialization (DCL), ARR from Rule 06. Arrays (ARR) and ERR from Rule 08. Exceptions and Error Handling (ERR). The link shows by clicking regulations above.

Implementations of the code

1. FIO51-CPP. Close files when they are no longer needed

We should use `std::fstream::close()` to close the file when it is no needed. When it is not closed before `std::terminate()` is called it can make many risks. It may allow an attacker to exhaust system resources and can increase the risk that data written into in-memory file buffers will not be flushed due to abnormal termination.

```
8  fp1 = fopen("plain.bin", "rb");
9  if (!fp) {
10     perror("key.txt");
11     exit(1);
12 }
13 if (!fp1) {
14     perror("plain.txt");
15     exit(1);
16 }
17 // Printing plain text array
18 while (fread((void*)buf, 1, sizeof(buf), fp1)) {
19     cout << "PLAIN : ";
20     for (i = 0; i < 16; i++) {
21         plain[i] = buf[i];
22         cout << plain[i] << " ";
23     }
24     cout << endl;
25 }
26 // Printing key array
27 while (fread((void*)buf, 1, sizeof(buf), fp)) {
28     cout << "KEY : ";
29     for (i = 0; i < 16; i++) {
30         key[i] = buf[i];
31         cout << hex << key[i] << " ";
32     }
33     cout << endl;
34 }
35 fclose(fp);
36 fclose(fp1);
37
```

We can see that the program is already closed by calling `fclose()` after the file is opened. So the program meets this standard.

2. FIO38-C. Do not copy a FILE object

We should use a copy of the pointer instead of using a copy of the FILE object. Using a copy of a FILE object in place of the original may result in a crash, which can be used in a denial-of-service attack.

```
1 int main(void)
2 {
3     int i = 0;
4     int key[32];        /// Array storing key
5     int plain[20];      /// Array storing plain text
6     int decrypted[20];  /// Array storing decrypted text
7     unsigned char buf[16];
8     unsigned char buf1[16];
9     /// Declaring file pointer
10    FILE *fp;
11    FILE *fp1;
12    FILE *fp2;
13    FILE *fp3;
14
15    cout << hex;
16
17    cout << "RC : ";
18    for (int i = 0; i < 10; i++)
19        cout << RCon(i) << " ";
20    cout << endl;
21
22    /// Reading binary file
23    fp = fopen("key.bin", "rb");
24
```

We can see that the original FILE object is not copied when it is used.

3. FIO45-C. Avoid TOCTOU race conditions while accessing files

A TOCTOU (time-of-check, time-of-use) race condition is possible when two or more concurrent processes are operating on a shared file system. TOCTOU conditions can be exploited when a program performs two or more file operations on the same file name or path name. TOCTOU race conditions can result in privilege escalation.

The solution is to use the x mode of fopen(). This mode causes fopen() to fail if the file exists. The x mode provides exclusive access to the file only if the host environment provides this support.

```
/// Writing cipher text and decrypted text in binary file
fp2 = fopen("cipher.bin", "wb");
fp3 = fopen("decrypted.bin", "wb");
if (!fp2) {
    perror("cipher.bin");
    exit(1);
}
if (!fp3) {
    perror("decrypted.bin");
    exit(1);
}
/// Writing cipher file
for (i = 0; i < 16; i++) {
    buf1[i] = cipher[i];
}
fwrite(&buf1, 1, sizeof(buf1), fp2);
```

Our program doesn't matched the solution, so we modified "wb" to "wx" in fopen().

```
/// Writing cipher text and decrypted text in binary file
fp2 = fopen("cipher.bin", "wx");
fp3 = fopen("decrypted.bin", "wx");
if (!fp2) {
    perror("cipher.bin");
    exit(1);
}
if (!fp3) {
    perror("decrypted.bin");
    exit(1);
}
/// Writing cipher file
for (i = 0; i < 16; i++) {
    buf1[i] = cipher[i];
}
fwrite(&buf1, 1, sizeof(buf1), fp2);
```

4. FIO46-C. Do not access a closed file

We shouldn't use stdout after the file is already closed. Using the value of a pointer to a FILE object after the associated file is closed is undefined behavior.

```
    }  
    fwrite(&buf1, 1, sizeof(buf1), fp3);  
  
    fclose(fp2);  
    fclose(fp3);  
  
    fputs("stdout successfully closed.", stderr);  
  
    system("pause");  
    return 0;  
}
```

We applied the code, instead of using printf() after the file is closed, we used fputs() to meet the standard.

5. DCL51-CPP. Do not declare or define a reserved identifier

We shouldn't declare or define a reserved identifier. No other identifiers are reserved. Declaring or defining an identifier in a context in which it is reserved results in undefined behavior.

In a user-defined literal, literal suffix identifiers are required to start with an underscore. Literal suffixes without the underscore prefix are reserved for future library implementations.

```
*/
int main(void)
{
    int i = 0;
    int key[32];        /// Array storing key
    int plain[20];      /// Array storing plain text
    int decrypted[20];  /// Array storing decrypted text
    unsigned char buf[16];
    unsigned char buf1[16];
    int operator"" _one;
    /// Declaring file pointer
    FILE *fp;
    FILE *fp1;
    FILE *fp2;
    FILE *fp3;

    cout << hex;

    cout << "RC : ";
```

We didn't had the code in our program, so we wrote it. A user-defined literal `operator"" _one` is declared. Literal suffix identifiers are required to start with an underscore for future library implementations.

```
int i = 0;
int key[32];        /// Array storing key
int plain[20];      /// Array storing plain text
int decrypted[20];  /// Array storing decrypted text
unsigned char buf[16];
unsigned char buf1[16];
int operator"" _one;
/// Declaring file pointer
FILE *fp;
FILE *fp1;
FILE *fp2;
```

In order to meet the standard, user-defined literal is named `operator"" _one`, which is not a reserved identifier.

6. FLP30-C. Do not use floating-point variables as loop counters

Floating-point numbers are subject to representational limitations just as integers are, and binary floating-point numbers cannot represent all real numbers exactly, even if they can be represented in a small number of decimal digits.


```

72 int InvSbox(int n) {
73     int i, j, k, m;
74
75     for (i = 1; i < 256; i++) {
76         m = i;
77         for (j = 7; j >= 0; j--) {
78             b[j] = m / (1 << j);    /// Making 1 to 255 into binary and store b[0] to b[7]
79             m %= (1 << j);
80         }
81         for (k = 0; k < 8; k++) {
82             b1[k] = b[(k + 2) % 8] ^ b[(k + 5) % 8] ^ b[(k + 7) % 8] ^ c1[k];    /// XOR after
83             m = m + b1[k] * (1 << k);    /// Restore from binary
84         }
85         invsbox[i] = InvGF[(255 - GF[m])];    /// Inverse
86     }
87     sbox[0] = 0xf3;    /// 0 doesn't have inverse
88     return invsbox[n];
89 }

```

In our code, we are not using the float as a counter of the loop, instead we are using integer to prevent the value having float points. So when we shift j times from "1", we want the value to be 0, not having a float point. By doing this, the float is not needed as a counter.

7. ARR30-C. Do not form or use out-of-bounds pointers or array subscripts

Writing to out-of-range pointers or array subscripts can result in a buffer overflow and the execution of arbitrary code with the permissions of the vulnerable process. Reading from out-of-range array subscripts can result in unintended information disclosure.

```

int Sbox(int num) {
    int m, w;
    int a = 1;
    for (int e = 0; e < 256; e++) { /// Fun
        InvGF[e] = a;
        w = a & 0x80;
        a <<= 1;
        if (w == 0x80)
            a ^= 0x69;
        a ^= InvGF[e];
        GF[InvGF[e]] = e;
    }
    InvGF[255] = 0;

    for (int i = 1; i < 256; i++) {

```

In our code, Sbox, InvSbox, and Rcon has index which is an integer. We can

meet this standard by using an unsigned type to avoid having to check for negative values while still rejecting out-of-bounds positive values of index.

```
int Sbox(size_t num) {
    int m, w;
    int a = 1;
    for (int e = 0; e < 256; e++) { /// Function G
        InvGF[e] = a;
        w = a & 0x80;
        a <<= 1;
        if (w == 0x80)
            a ^= 0x69;
        a ^= InvGF[e];
        GF[InvGF[e]] = e;
    }
    InvGF[255] = 0;

    for (int i = 1; i < 256; i++) {
        m = InvGF[(255 - GF[i])]; /// Invers
        for (int j = 7; j >= 0; j--) { /// Genera
            b[j] = m / (1 << j);
            m %= (1 << j);
        }
    }
}
```

We can use size_t instead of integer for preventing errors.

8. STR34-C. Cast characters to unsigned char before converting to larger integer sizes

Signed character data must be converted to unsigned char before being assigned or converted to a larger signed type. Conversion of character data resulting in a value in excess of UCHAR_MAX is an often-missed error that can result in a disturbingly broad range of potentially severe vulnerabilities.

```
void SubBytes()
{
    cout << "SB: ";
    for (int i = 0; i < 4; i++)
    {
        for (int j = 0; j < 4; j++)
        {
            state[j][i] = Sbox(state[j][i]);
            cout << state[j][i] << " ";
        }
    }
    cout << endl;
}
```

In our code, state array is an integer value, and Sbox is an unsigned char array. We defined variables as an unsigned char before assigning it to an integer, so we meet the standard.

9. DCL40-C. Do not create incompatible declarations of the same function or object

Two or more incompatible declarations of the same function or object must not appear in the same program because they result in undefined behavior. We should use same declaration of the variable which has the same name.

```
/* @brief Substitute each Byte value of state array by Sbox
 */
void SubBytes()
{
    cout << "SB: ";
    for (int i = 0; i < 4; i++)
    {
        for (int j = 0; j < 4; j++)
        {
            state[j][i] = Sbox(state[j][i]);    /// assigned Sbox as an unsigned char before converted into state a
            cout << state[j][i] << " ";
        }
    }
    cout << endl;
}
/**
```

In our code, we are using same declaration of same named value, for example int for every i, j, and size_t for every num variables. So we are meeting the standard.

10. ERR33-C. Detect and handle standard library errors

We should check for every returned value. Failing to detect error conditions can lead to unpredictable results, including abnormal program termination and denial-of-service attacks or could allow an attacker to run arbitrary code.

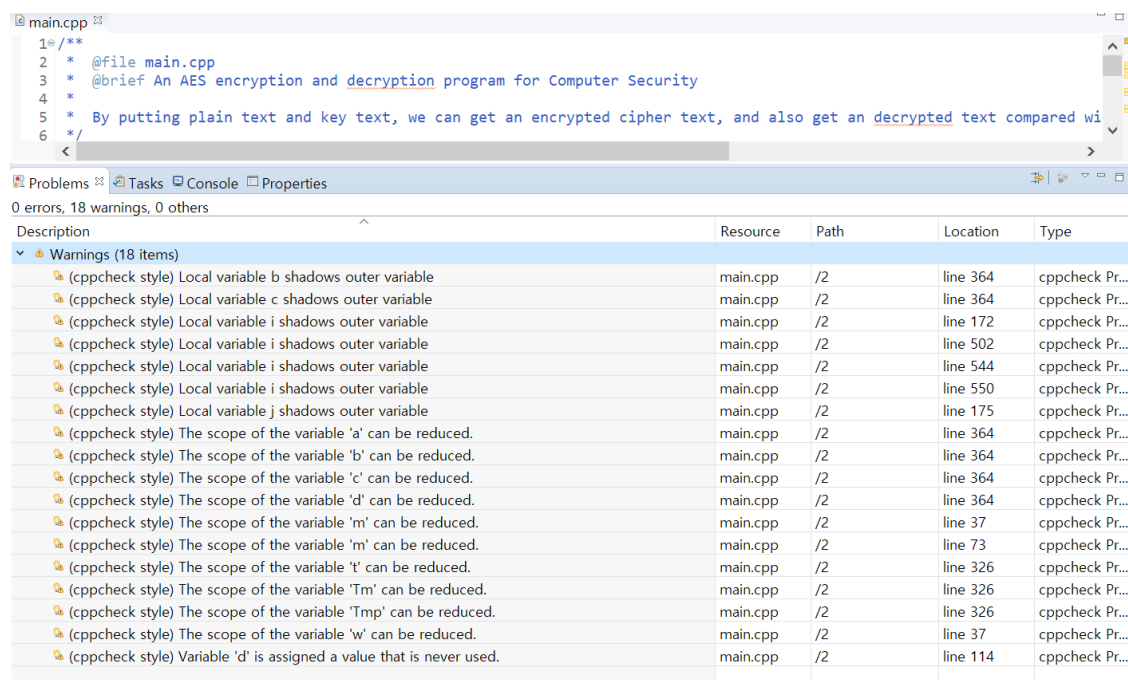
```
/// Reading binary file
fp = fopen("key.bin", "rb");
fp1 = fopen("plain.bin", "rb");
if (!fp) {
    perror("key.txt");
    exit(1);
}
if (!fp1) {
    perror("plain.txt");
    exit(1);
}
/// Printing plain text array
while (fread((void*)buf, 1, sizeof(buf), fp1)) {
    if(!fread((void*)buf, 1, sizeof(buf), fp1)){
        break;    /// Indicating error
    }
}
```

According to the C Standard, the `fopen()` function returns a `NULL` value to indicate that an error occurred. This compliant solution tests for this condition before reading from a file to eliminate the chance of operating on the wrong portion of the file if `fopen()` fails.

```
/// Reading binary file
fp = fopen("key.bin", "rb");
fp1 = fopen("plain.bin", "rb");
if (fp==NULL) {    /// fopen returns NULL when error occurs
    perror("key.txt");
    exit(1);
}
if (fp1==NULL) {    /// fopen returns NULL when error occurs
    perror("plain.txt");
    exit(1);
}
/// Printing plain text array
while (fread((void*)buf, 1, sizeof(buf), fp1)) {
```

The code is modified because `fopen` could return `NULL`, not `FALSE`. By changing the code, we can handle the error exactly.

Application of corrections from errors by cppcheck



We have 18 warnings detected by cppcheck.

1. Local variable b, c shadows outer variable

```
361 void InvMixColumns()  
362 {  
363     unsigned char a, b, c, d;  
364     for (int i = 0; i < 4; i++)  
365     {  
366         a = state[0][i];  
367         b = state[1][i];  
368         c = state[2][i];  
369         d = state[3][i];  
370  
371         /// Operating multiply with fixed polynomial expression by Multiply and XOR operation  
372         state[0][i] = Multiply(a, 0x0e) ^ Multiply(b, 0x0b) ^ Multiply(c, 0x0d) ^ Multiply(d, 0x09);  
373         state[1][i] = Multiply(a, 0x09) ^ Multiply(b, 0x0e) ^ Multiply(c, 0x0b) ^ Multiply(d, 0x0d);  
374         state[2][i] = Multiply(a, 0x0d) ^ Multiply(b, 0x09) ^ Multiply(c, 0x0e) ^ Multiply(d, 0x0b);  
375         state[3][i] = Multiply(a, 0x0b) ^ Multiply(b, 0x0d) ^ Multiply(c, 0x09) ^ Multiply(d, 0x0e);  
376     }
```

Unsigned char b, c shadows outer variable. So we changed the code by putting definitions inside 'for' loop. And we changed b and c by b_ and c_

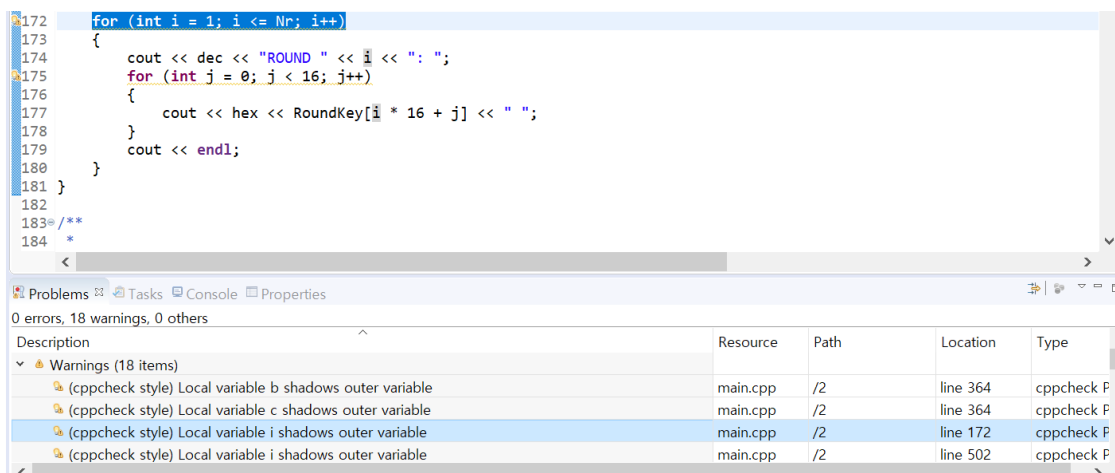
due to arrays defined ahead by b and c. When the variables aren't used outside the loop, it's better to define it inside the loop.

```

361 void InvMixColumns()
362 {
363     for (int i = 0; i < 4; i++)
364     {
365         unsigned char a, b_, c_, d;    /// By defining a,b,c,d here, we can stop them from shadowing outer variables
366         a = state[0][i];
367         b_ = state[1][i];
368         c_ = state[2][i];
369         d = state[3][i];
370
371         /// Operating multiply with fixed polynomial expression by Multiply and XOR operation
372         state[0][i] = Multiply(a, 0x0e) ^ Multiply(b_, 0x0b) ^ Multiply(c_, 0x0d) ^ Multiply(d, 0x09);
373         state[1][i] = Multiply(a, 0x09) ^ Multiply(b_, 0x0e) ^ Multiply(c_, 0x0b) ^ Multiply(d, 0x0d);
374         state[2][i] = Multiply(a, 0x0d) ^ Multiply(b_, 0x09) ^ Multiply(c_, 0x0e) ^ Multiply(d, 0x0b);
375         state[3][i] = Multiply(a, 0x0b) ^ Multiply(b_, 0x0d) ^ Multiply(c_, 0x09) ^ Multiply(d, 0x0e);
376     }

```

2. Local variable i, j shadows outer variable



Local variable i and j are already defined before, so we don't have to define them as integer again.

```

j
for (i = 1; i <= Nr; i++)    /// Local variable i shadowed outer variable, so we changed it by erasing int
{
    cout << dec << "ROUND " << i << ": ";
    for (j = 0; j < 16; j++)
    {
        cout << hex << RoundKey[i * 16 + j] << " ";
    }
    cout << endl;
}

```

We solved the warning by removing int ahead of i and j.

3. The scope of the variable Tmp, Tm, t can be reduced

```

325=void MixColumns()
326 {
327     unsigned char Tmp, Tm, t;
328
329     for (int i = 0; i < 4; i++)
330     {
331         t = state[0][i];
332
333         Tmp = state[0][i] ^ state[1][i] ^ state[2][i] ^ state[3][i];
334         // execute matrix multiply operation and store in Tm
335         Tm = state[0][i] ^ state[1][i];
336         Tm = xtime(Tm);
337         state[0][i] ^= Tm ^ Tmp;
338         Tm = state[1][i] ^ state[2][i];
339         Tm = xtime(Tm);
340         state[1][i] ^= Tm ^ Tmp;

```

Problems Tasks Console Properties

0 errors, 5 warnings, 0 others

Description	Resource	Path	Location
Warnings (5 items)			
(cppcheck style) Local variable a shadows outer variable	main.cpp	/2	li
(cppcheck style) Local variable d shadows outer variable	main.cpp	/2	li
(cppcheck style) The scope of the variable 't' can be reduced.	main.cpp	/2	li
(cppcheck style) The scope of the variable 'Tm' can be reduced.	main.cpp	/2	li
(cppcheck style) The scope of the variable 'Tmp' can be reduced.	main.cpp	/2	li

We can solve it by putting the definitions of variable Tmp, Tm and t inside the 'for' loop.

```

325=void MixColumns()
326 {
327     for (int i = 0; i < 4; i++)
328     {
329         unsigned char Tmp, Tm, t; // putting inside the loop solves warnings
330         t = state[0][i];
331
332         Tmp = state[0][i] ^ state[1][i] ^ state[2][i] ^ state[3][i];
333         // execute matrix multiply operation and store in Tm
334         Tm = state[0][i] ^ state[1][i];
335         Tm = xtime(Tm);
336         state[0][i] ^= Tm ^ Tmp;
337         Tm = state[1][i] ^ state[2][i];
338         Tm = xtime(Tm);
339         state[1][i] ^= Tm ^ Tmp;
340

```

Problems Tasks Console Properties

0 errors, 2 warnings, 0 others

Description	Resource	Path	Location
Warnings (2 items)			
(cppcheck style) Local variable a shadows outer variable	main.cpp	/2	line 39
(cppcheck style) Local variable d shadows outer variable	main.cpp	/2	line 11

4. Local variable m, w shadows outer variable

```

71 int InvSbox(size_t n) {           /// preventing index out-of-bound
72     int i, j, k;
73
74     for (i = 1; i < 256; i++) {
75         int m = i;
76         for (j = 7; j >= 0; j--) {
77             b[j] = m / (1 << j);    /// Making 1 to 255 into binary and store b[0] to b[7]
78             m %= (1 << j);
79         }
80         for (k = 0; k < 8; k++) {
81             b1[k] = b[(k + 2) % 8] ^ b[(k + 5) % 8] ^ b[(k + 7) % 8] ^ c1[k];    /// XOR after matrix op
82             m = m + b1[k] * (1 << k);    /// Restore from binary
83         }
84         invsbox[i] = InvGF((255 - GF[m])1);    /// Inverse

```

Problems Tasks Console Properties

0 errors, 1 warning, 0 others

Description	Resource	Path	Loc
Warnings (1 item)			
(cppcheck style) Local variable m shadows outer variable	main.cpp	/2	line

Local variable m and w shadows outer variable, so we can define it outside of function to make it global variable. And we can remove int ahead of m and w.

```

14 #define xtime(x) ((x << 1) ^ ((x >> 7) & 1) * 0x169)    /// Declaring xtime
15 #define Multiply(x,y) (((y & 1) * x) ^ ((y >> 1 & 1) * xtime(x)) ^ ((y >> 2 & 1) * xtime(xtime(x))) ^ ((y >> 3 & 1) * xtime(xtime(xtime(x))))
16 /// Declaring the Multiply for Mix column
17 #define Nb 4    /// Length of the Block
18 #define Nk 4    /// Length of the key
19 #define Nr 10    /// Number of Round
20 int state[4][4] = { 0, };    /// Array storing instant state
21 int RoundKey[240] = { 0, };    /// Array storing Round Key
22 int c[8] = { 1,0,1,0,1,0,0,0 };    /// Array used for Sbox XOR operation
23 int c1[8] = { 1,1,1,0,0,0,1,1 };    /// Array used for Inverse Sbox XOR operation
24 int b[8] = { 0, };
25 int b1[8] = { 0, };
26 int m, w;    /// By defining m and w here, we can stop m, w from shadowing outer variable
27 unsigned char sbox[256];
28 unsigned char invsbox[256];    /// Inverse of the Sbox
29 unsigned char GF[256] = { 0, };    /// Array of Galua Field(2^8) for calculation inverse element
30 unsigned char InvGF[256];    /// Array of Inverse Galua Field(2^8) for calculation inverse element

```

By defining m and w here, we can stop m, w from shadowing outer variable.

5. Variable d is assigned a value that is never used.


```

112=void KeyExpansion(int *key)
113 {
114     int i, j, d = 0;
115     unsigned char temp[4], k;
116     cout << "ROUND 0: ";
117     for (i = 0; i < Nk; i++)    /// Storing key value in Round key
118     {
119         RoundKey[4 * i] = key[4 * i];
120         RoundKey[4 * i + 1] = key[4 * i + 1];
121         RoundKey[4 * i + 2] = key[4 * i + 2];
122         RoundKey[4 * i + 3] = key[4 * i + 3];
123     }
124     for (k = 0; k < 16; k++) {
125         cout << RoundKey[k] << " ";
126     }
127     cout << endl;
128     while (i < (Nh * (Nr + 1)))
129

```

Problems Tasks Console Properties

1 errors, 1 warning, 8 others

Description	Resource	Path
invalid operands of types 'int' and 'int [8]' to binary 'operator*'	main.cpp	/2
make: *** [subdir.mk:20: main.o] Error 1	2	
Warnings (1 item)		
(cppcheck style) Variable 'd' is assigned a value that is never used.	main.cpp	/2

Variable d is assigned with 0, and it is told that the value is never used. So we solved the problem by erasing '=0'.

```

112=void KeyExpansion(int *key)
113 {
114     int i, j, d;    /// solved by cppcheck erasing definition by d
115     unsigned char temp[4], k;
116     cout << "ROUND 0: ";
117     for (i = 0; i < Nk; i++)    /// Storing key value in Round key
118     {
119         RoundKey[4 * i] = key[4 * i];
120         RoundKey[4 * i + 1] = key[4 * i + 1];
121         RoundKey[4 * i + 2] = key[4 * i + 2];
122         RoundKey[4 * i + 3] = key[4 * i + 3];
123     }
124     for (k = 0; k < 16; k++) {
125         cout << RoundKey[k] << " ";
126     }
127     cout << endl;

```

6. Invalid arguments

```

532     int cipher[128] = { 0, };    /// Array storing cipher text
533     cout << endl << endl << "<-----ENCRYPTION----->" << endl << endl << "KEY EXPANSION" << endl;
534     KeyExpansion(*key);
535     Cipher(plain, cipher);    /// Encryption
536     cout << endl << "CIPHER : ";
537     for (i = 0; i < 16; i++)
538         cout << cipher[i] << " ";
539     cout << endl << endl << "<-----DECRYPTION----->" << endl << endl;
540
541     InvCipher(cipher, decrypted);    /// Decryption
542     cout << "DECRYPTED : ";
543     for (i = 0; i < 16; i++)
544         cout << decrypted[i] << " ";
545     cout << endl;

```

Problems Tasks Console Properties

error, 0 warnings, 0 others

Description	Resource	Path	Location
Errors (1 item)			
Invalid arguments	main.cpp	/2	line 534

This issue is not existing in our project, because it is already solved. But we can see that parameter key needs to remove * ahead.

```

531
532 int cipher[128] = { 0, }; // Array storing cipher text
533 cout << endl << endl << "<-----ENCRYPTION----->" << endl << endl << "KEY EXPANSION" << endl;
534 KeyExpansion(key);
535 Cipher(plain, cipher); // Encryption
536 cout << endl << "CIPHER : ";
537 for (i = 0; i < 16; i++)
538     cout << cipher[i] << " ";
539 cout << endl << endl << "<-----DECRYPTION----->" << endl << endl;
540
541 InvCipher(cipher, decrypted); // Decryption
542 cout << "DECRYPTED : ";
543 for (i = 0; i < 16; i++)
544     cout << decrypted[i] << " ";
545 cout << endl;

```

7. Array accessed out of bounds

```

47 }
48 InvGF[256] = 0;
49
50 for (int i = 1; i < 256; i++) {
51     m = InvGF[(255 - GF[i])]; // Inverse element
52     for (int j = 7; j >= 0; j--) {
53         b[j] = m / (1 << j); // Generating inverse element into binary b[0] to b[7]
54         m %= (1 << j);
55     }
56     for (int k = 0; k < 8; k++) {
57         b1[k] = b[k] ^ b[(k + 4) % 8] ^ b[(k + 5) % 8] ^ b[(k + 6) % 8] ^ b[(k + 7) % 8] ^ c[k];
58         m += b1[k] * (1 << k); // Restore from binary
59     }
60 }

```

Problems Tasks Console Properties

error, 0 warnings, 0 others

Description	Resource	Path	Location
(cppcheck error) Array 'InvGF[256]' accessed at index 256, which is out of bounds.	main.cpp	/2	line

Cppcheck finds error when index is out of bounds. Index of array 'InvGF[256]' should be lower than 256.

```

47 }
48 InvGF[255] = 0;
49
50 for (int i = 1; i < 256; i++) {
51     m = InvGF[(255 - GF[i])]; // Inverse element
52     for (int j = 7; j >= 0; j--) {
53         b[j] = m / (1 << j); // Generating inverse element into binary b[0] to b[7]
54         m %= (1 << j);
55     }
56     for (int k = 0; k < 8; k++) {
57         b1[k] = b[k] ^ b[(k + 4) % 8] ^ b[(k + 5) % 8] ^ b[(k + 6) % 8] ^ b[(k + 7) % 8] ^ c[k]; // 000 0000
58         m += b1[k] * (1 << k); // Restore from binary
59     }
60 }

```

Problems Tasks Console Properties

0 items

Description	Resource	Path	Location	Type
-------------	----------	------	----------	------

8. Symbol 'i' could not be resolved

```

422
423     /// Storing final encrypted state array in cipher array
424     for (i = 0; i < 4; i++)
425     {
426         for (int j = 0; j < Nb; j++)
427         {
428             cipher[4 * i + j] = state[j][i];
429         }
430     }
431 }
432 /**
433  *
434  * @param [in] cipher The array which stores ciphered text

```

Problems Tasks Console Properties

0 errors, 0 warnings, 0 others

Description	Resource	Path
Errors (5 items)		
Symbol 'i' could not be resolved	main.cpp	/2
Symbol 'i' could not be resolved	main.cpp	/2
Symbol 'i' could not be resolved	main.cpp	/2
Symbol 'i' could not be resolved	main.cpp	/2
Symbol 'i' could not be resolved	main.cpp	/2

Cppcheck checks if the symbol is not resolved. It has to be defined whether if it is a variable. So we already have int ahead of variable i.

```

416         cout << endl;
417     }
418     cout << "Round 10" << endl;
419     SubBytes();
420     ShiftRows();
421     AddRoundKey(Nr);    /// XORing last key and current block
422
423     /// Storing final encrypted state array in cipher array
424     for (int i = 0; i < 4; i++)
425     {
426         for (int j = 0; j < Nb; j++)
427         {
428             cipher[4 * i + j] = state[j][i];
429         }
430     }
431 }
432 /**
433  *
434  * @param [in] cipher The array which stores ciphered text

```

9. Consecutive break are unnecessary

```

500 fp = fopen("key.bin", "rb");
501 fp1 = fopen("plain.bin", "rb");
502 if (fp == NULL) { // fopen returns NULL when error occurs
503     perror("key.txt");
504     exit(1);
505     break;
506 }
507 if (fp1 == NULL) { // fopen returns NULL when error occurs
508     perror("plain.txt");
509     exit(1);
510 }
511 // Printing plain text array
512 while (fread((void*)buf, 1, sizeof(buf), fp1)) {
513     cout << "PLAIN : ";
514     for (i = 0; i < 16; i++) {
515         plain[i] = buf[i];
516         cout << plain[i] << " ";
517     }
518     cout << endl;

```

Problems Tasks Console Properties

0 errors, 1 warning, 0 others

Description	Resource	Path	Location	Type
Warnings (1 item)				
(cppcheck style) Consecutive return, break, continue, goto or throw statements are unnecessary.	main.cpp	/2	line 505	cppcl

Cppcheck checks whether if consecutive statements are unnecessary to be used. We already don't have any unnecessary statements.

```

498 // Reading binary file
499 fp = fopen("key.bin", "rb");
500 fp1 = fopen("plain.bin", "rb");
501 if (fp == NULL) { // fopen returns NULL when error occurs
502     perror("key.txt");
503     exit(1);
504 }
505 if (fp1 == NULL) { // fopen returns NULL when error occurs
506     perror("plain.txt");
507     exit(1);
508 }
509 // Printing plain text array
510 while (fread((void*)buf, 1, sizeof(buf), fp1)) {
511     cout << "PLAIN : ";
512     for (i = 0; i < 16; i++) {
513         plain[i] = buf[i];
514     }

```

10. Unmatched ')'

```

510 // Printing plain text array
511 while (fread((void*)buf, 1, sizeof(buf), fp1)) {
512     cout << "PLAIN : ";
513     for (i = 0; i < 16; i++) {
514         plain[i] = buf[i];
515         cout << plain[i] << " ";
516     }
517     cout << endl;
518 }
519 // Printing key array
520 while (fread((void*)buf, 1, sizeof(buf), fp)) {
521     cout << "KEY : ";
522     for (i = 0; i < 16; i++) {

```

Problems Tasks Console Properties

1 error, 0 warnings, 0 others

Description	Resource	Path
Errors (1 item)		
(cppcheck error) Unmatched ')'. Configuration: "	main.cpp	/2

Cppcheck checks if apostrophe is closed. We already closed well to define fread().

```
main.cpp
503 perror("key.txt");
504 exit(1);
505 }
506 if (fp1 == NULL) { /// fopen returns NULL when error occurs
507     perror("plain.txt");
508     exit(1);
509 }
510 /// Printing plain text array
511 while (fread((void*)buf, 1, sizeof(buf), fp1)) {
512     cout << "PLAIN : ";
513     for (i = 0; i < 16; i++) {
514         plain[i] = buf[i];
515         cout << plain[i] << " ";
516     }
517     cout << endl;
518 }
519 /// Printing key array
520 while (fread((void*)buf, 1, sizeof(buf), fp)) {
521     cout << "KEY : ";
522     for (i = 0; i < 16; i++) {
```

Problems Tasks Console Properties

0 items

Description	Resource	Path	Location	Type
-------------	----------	------	----------	------

The reduction of the cyclomatic complexity by metriculator

We have 14 functions in our project. And when we sorted it by McCabe, we can see that two functions : KeyExpansion() and main() V(G) are over 10. So we modified the functions to make them same or below than 10.

main.cpp

```

24 int b1[8] = { 0, };
25 int m, w;
26 unsigned char a, b_, c_, d;    /// By defining a,b,c,d here, we can stop them from shadowing outer
27 unsigned char sbbox[256];
28 unsigned char invsbbox[256];    /// Inverse of the Sbox
29 unsigned char GF[256] = { 0, };    /// Array of Galua Field(2^8) for calculation inverse element
30 unsigned char InvGF[256];    /// Array of Inverse Galua Field(2^8) for calculation inverse element
31 /**
32 *
33 * @param [in] num The index of Sbox array
34 * @return unsigned char The value of Sbox for calculation

```

metriculator

Scope (14 items)	LSLOC	EfferentC...	McCabe	NbParams	NbMem...
• KeyExpansion(i...	50	0	19	1	0
• main(void)	70	0	14	1	0
• Sbox(size_t nu...	22	0	6	1	0
• Cipher(int * pla...	22	0	6	2	0
• InvCipher(int * ...	22	0	6	2	0
• InvSbox(size_t n)	13	0	4	1	0
• MixColumns()	22	0	4	0	0
• InvMixColumns()	15	0	4	0	0
• RCon(size_t nu...	7	0	3	1	0
• AddRoundKey(...	7	0	3	1	0
• SubBytes()	7	0	3	0	0
• InvSubBytes()	7	0	3	0	0
• ShiftRows()	23	0	3	0	0
• InvShiftRows()	23	0	3	0	0

Main function : $V(G)=14$

```

475=/**
476 * @brief Opening cipher and decrypted binary file and writing the output
477 */
478
479 void Decrypt() {
480     int i;
481     unsigned char buf1[16];
482     int decrypted[20]; // Array storing decrypted text
483     int cipher[128] = { 0, }; // Array storing cipher text
484
485     FILE *fp2, *fp3;
486     // Writing cipher text and decrypted text in binary file
487     fp2 = fopen("cipher.bin", "wx");
488     fp3 = fopen("decrypted.bin", "wx");
489     if (!fp2) {
490         perror("cipher.bin");
491         exit(1);
492     }
493     if (!fp3) {
494         perror("decrypted.bin");
495         exit(1);
496     }
497     // Writing cipher file
498     for (i = 0; i < 16; i++) {
499         buf1[i] = cipher[i];
500     }
501     fwrite(&buf1, 1, sizeof(buf1), fp2);
502
503     // Writing decrypted file
504     for (i = 0; i < 16; i++) {

```

We worked on main function first. We made a new function called Decrypt(), which opens cipher and decrypted binary file and writes the output. By this modification, the $V(G)$ of main() became 10, and it meets our expectation.

Scope (15 items)	LSLOC	EfferentC...	McCabe	NbParams	NbMem...
• KeyExpansion(i...	50	0	19	1	0
• main(void)	49	0	10	1	0
• Sbox(size_t nu...	22	0	6	1	0
• Cipher(int * pla...	22	0	6	2	0
• InvCipher(int * ...	22	0	6	2	0
• Decrypt()	23	0	5	0	0
• InvSbox(size_t n)	13	0	4	1	0
• MixColumns()	22	0	4	0	0
• InvMixColumns()	15	0	4	0	0
• RCon(size_t nu...	7	0	3	1	0
• AddRoundKey(...	7	0	3	1	0
• SubBytes()	7	0	3	0	0
• InvSubBytes()	7	0	3	0	0
• ShiftRows()	23	0	3	0	0
• InvShiftRows()	23	0	3	0	0

KeyExpansion function : V(G)=19

```

149         temp[3] = Sbox(temp[3]);
150     }
151     temp[0] = temp[0] ^ RCon(i / Nk - 4); // We changed this line by adding -4 to prevent it from hardcoding
152 }
153 RoundKey[4 * i + 0] = RoundKey[(i - Nk) * 4 + 0] ^ temp[0];
154 RoundKey[4 * i + 1] = RoundKey[(i - Nk) * 4 + 1] ^ temp[1];
155 RoundKey[4 * i + 2] = RoundKey[(i - Nk) * 4 + 2] ^ temp[2];
156 RoundKey[4 * i + 3] = RoundKey[(i - Nk) * 4 + 3] ^ temp[3];
157 i++;
158 }
159 for (i = 1; i <= Nr; i++)
160 {
161     cout << dec << "ROUND " << i << ": ";
162     for (j = 0; j < 16; j++)
163     {
164         cout << hex << RoundKey[i * 16 + j] << " ";
165     }
166 }

```

Next function was KeyExpansion, which had higher complexity compared to the main function. The code had the part of hardcoding which made the cyclomatic complexity high. So we erased the part, and found the solution by putting minus 4 in the code "temp[0] = temp[0] ^ RCon(i / Nk);". After checking the code, it was successfully built and the complexity became 8.

Scope (15 items)	McCabe	EfferentC...	LSLOC	NbParams	NbMem...
main(void)	10	0	49	1	0
KeyExpansion(i...	8	0	38	1	0
Sbox(size_t nu...	6	0	22	1	0
Cipher(int * pla...	6	0	22	2	0
InvCipher(int * ...	6	0	22	2	0
Decrypt()	5	0	23	0	0
InvSbox(size_t n)	4	0	13	1	0
MixColumns()	4	0	22	0	0
InvMixColumns()	4	0	15	0	0
RCon(size_t nu...	3	0	7	1	0
AddRoundKey(...	3	0	7	1	0
SubBytes()	3	0	7	0	0
InvSubBytes()	3	0	7	0	0
ShiftRows()	3	0	23	0	0
InvShiftRows()	3	0	23	0	0

Application of cppcheck to the reviewed modules

```
*main.cpp
463  * @brief Opening cipher and decrypted binary file and writing the output
464  */
465
466 void Decrypt() {
467     int i;
468     unsigned char buf1[16];
469     int decrypted[20]; // Array storing decrypted text
470     int cipher[128] = { 0, }; // Array storing cipher text
471
472     FILE *fp2, *fp3;
473     // Writing cipher text and decrypted text in binary file
474     fp2 = fopen("cipher.bin", "wx");
475     fp3 = fopen("decrypted.bin", "wx");
476     if (!fp2) {
477         perror("cipher.bin");
478         exit(1);
479     }
480     if (!fp3) {
481         perror("decrypted.bin");
482         exit(1);
483     }
}
```

Problems 3 Tasks Console Properties metriculator

0 errors, 1 warning, 0 others

Description	Resource	Path
Warnings (1 item)		
(cppcheck style) Variable 'decrypted' is not assigned a value.	main.cpp	/2

The cppcheck detected warning from new function Decrypt, which is telling that Variable 'decrypted' is not assigned a value. So we defined it like Variable 'cipher', and the warning is removed.

```
465
466 void Decrypt() {
467     int i;
468     unsigned char buf1[16];
469     int decrypted[20] = { 0, }; // Array storing decrypted text
470     int cipher[128] = { 0, }; // Array storing cipher text
471
472     FILE *fp2, *fp3;
473     // Writing cipher text and decrypted text in binary file
474     fp2 = fopen("cipher.bin", "wx");
475     fp3 = fopen("decrypted.bin", "wx");
476     if (!fp2) {
477         perror("cipher.bin");
478         exit(1);
479     }
480     if (!fp3) {
481         perror("decrypted.bin");
482         exit(1);
483     }
}
```

Problems 3 Tasks Console Properties metriculator

0 items

Description	Resource
-------------	----------

Conclusion

The final modified code of our program works well.

```
C:\WINDOWS\system32\cmd.exe
SR: 2a 4c 47 3c c0 dd ab 99 6c f9 c5 2f aa 24 57 3
SB: 5 97 10 20 b3 f5 69 87 2f de 6a 4f 79 d2 6c d1
AR: f2 fc 46 60 eb 77 10 b7 68 1d 7b be b3 dd 87 ec
MC: ed 42 9 8e 67 88 ea 3e 5a 1b 40 b1 92 80 ac bb

Round 7
SR: ed 80 40 3e 67 42 ac b1 5a 88 9 bb 92 1b ea 8e
SB: 2d 7d e4 5f 75 ea 1a 33 c8 50 77 c6 a7 89 5c c9
AR: d0 c9 b9 d2 da 3 35 43 d7 11 1f 7 2a 45 a6 5
MC: 76 1c fb e3 75 75 ad 2 98 64 85 a7 a5 30 d8 81

Round 8
SR: 76 30 85 2 75 1c d8 a7 98 75 fb 81 a5 64 ad e3
SB: 48 81 2a 46 ee 1b c4 ff 12 ee cf 7a 37 26 7c cd
AR: 78 b9 cd 47 bc 46 b6 2 a2 46 88 cb a5 ab ee c0
MC: 54 d3 77 bb e5 5a 73 82 90 6e 9d c4 b5 ad be 86

Round 9
SR: 54 ad 9d 82 e5 d3 be c4 90 5a 77 86 b5 6e 73 bb
SB: 3 7c df 84 4 24 ed 9f 53 c8 2b c5 6 c7 96 c6
AR: 21 49 2a 8 66 41 78 63 b1 3d 1e 89 24 e2 43 7a
MC: 15 2a 8a ff e5 e 83 54 6d ba f4 38 45 a b5 5

Round 10
SR: 15 a f4 54 e5 2a b5 38 6d e 8a 5 45 ba 83 ff
SB: 0 1 2 3 4 5 6 7 8 9 a b c d e f
AR: 0 11 22 33 44 55 66 77 88 99 aa bb cc dd ee ff

DECRYPTED : 0 11 22 33 44 55 66 77 88 99 aa bb cc dd ee ff
cipher.bin: File exists
계속하려면 아무 키나 누르십시오 . . . █
```

We've met 10 standards of the rules, SEI CERT Coding Standards including C and C++.

The screenshot shows the Eclipse IDE interface. The main window displays the source code for `main.cpp`. The code implements a simple encryption and decryption process using a key expansion and a cipher function. The console at the bottom shows the build output, indicating that the program was successfully compiled with 0 errors and 0 warnings.

```

2/main.cpp - Eclipse IDE
ctor Navigate Search Project Run Window Help

[Icons] [Quick Access] [Java EE]

main.cpp
545     for (i = 0; i < 16; i++) {
546         key[i] = buf[i];
547         cout << hex << key[i] << " ";
548     }
549     cout << endl;
550 }
551 fclose(fp);
552 fclose(fp);
553
554 cout << endl << endl << "<-----ENCRYPTION----->" << endl << endl << "KEY EXPANSION" << endl;
555 KeyExpansion(key);
556 Cipher(plain, cipher);    /// Encryption
557 cout << endl << "CIPHER : ";
558 for (i = 0; i < 16; i++)
559     cout << cipher[i] << " ";
560 cout << endl << endl << "<-----DECRYPTION----->" << endl << endl;
561
562 InvCipher(cipher, decrypted);    /// Decryption
563 cout << "DECRYPTED : ";
564 for (i = 0; i < 16; i++)
565     cout << decrypted[i] << " ";
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1
```


function.

We've met 10 standards of SEI CERT Coding Standards by modifying our code, solved 10 issues from cppcheck by selecting some of the problems that cppcheck is able to detect, and finally reduced cyclomatic complexity of our modules each by same or lower than 10. Our program was finally improved by SEI CERT Coding Standards, Cppcheck, and Metriculator. The potential of error occurring got lowered by handling some exceptions and we checked errors and warnings by cppcheck to ensure the confidence of our program. We have finally checked our complexity by metriculator to make our program work efficiently. We made a change to our code by removing hard coded part and finding a new solution. We can say that our final program's software quality has improved to a better quality.