

Forum:[Hack Forums](#)

Thread: [Lock/Unlock - Protect - Encrypt Your Files - Microsoft AES 256 File Encryption](#)

Hackers - July 20, 2024, 1:32 pm

Microsoft AES 256 File Encryption (MS-AES-256-FE.exe)

Encryption is crucial for ensuring data privacy and protecting sensitive information from unauthorized access. It plays a vital role in maintaining personal security by safeguarding digital communications and stored data. Robust encryption practices help prevent data breaches and cyberattacks, ensuring that personal and confidential information remains confidential. In an increasingly digital world, the importance of encryption for data privacy and personal security cannot be overstated.

[View on Github](#)

This simple program can take any type of file, encrypt it with AES-256 using Windows, and bcrypt functions. It is a simple CLI program that takes the parameters for your file, password, and lock/unlock commands.

This code share is meant to guide you so you can create your own encryption related programs.

Source Code Spoiler ([Click to View](#))

```
Code #ifndef MSAES256FE_SECURE_H
#define MSAES256FE_SECURE_H
```

```
#include <windows.h>
#include <bcrypt.h>
#include <iostream>
#include <fstream>
#include <vector>
#include <stdexcept>
#include <iomanip>
```

```
#pragma comment(lib, "bcrypt.lib")
```

```
#define MSAES256FE_SUCCESS(Status) ((NTSTATUS)(Status) >= 0)
```

```
class MSAES256_FileEncryption {
public:
    MSAES256_FileEncryption(const wchar_t* file, const std::wstring& key, const std::wstring&
operation) :
        file(file), key(key), operation(operation) {}
```

```
    void Execute() {
        try {
```

```

    if (operation == L"lock") {
        LockFile();
    }
    else if (operation == L"unlock") {
        UnlockFile();
    }
    else {
        std::wcerr << L"ERR: Invalid operation. Please use 'lock' or 'unlock'." << std::endl;
    }
}
catch (const std::exception& ex) {
    std::cerr << "ERR: An error occurred: " << ex.what() << std::endl;
}
}

```

private:

```

const wchar_t* file;
std::wstring key;
std::wstring operation;

```

```

void ShowError(const wchar_t* msg, NTSTATUS status) {
    std::wcerr << L"ERR: " << msg << L" (Code: 0x" << std::hex << status << L")" << std::endl;
}

```

```

void ValidateStatus(const wchar_t* msg, NTSTATUS status) {
    if (!MSAES256FE_SUCCESS(status)) {
        ShowError(msg, status);
        throw std::runtime_error("ERR: The operation has failed.");
    }
}

```

```

void AnnounceOperation(const wchar_t* operation) {
    std::wcout << L"MSG: Executing the following operation: " << operation << std::endl;
}

```

```

void DisplayHex(const void* data, size_t size) {
    const uint8_t* byteData = static_cast<const uint8_t*>(data);
    for (size_t i = 0; i < size; ++i) {
        std::wcout << std::hex << std::setw(2) << std::setfill(L'0') << static_cast<int>(byteData[i]) <<
L" ";
        if ((i + 1) % 16 == 0) {
            std::wcout << std::endl;
        }
    }
    if (size % 16 != 0) {
        std::wcout << std::endl;
    }
}

```

```

void GenerateRandomData(std::vector<BYTE>& buffer) {
    AnnounceOperation(L"Generating random data.");
    BCryptAlgHandle randAlgHandle;
    ValidateStatus(L"ERR: The function BCryptOpenAlgorithmProvider has failed.",

```

```

BCryptOpenAlgorithmProvider(&randAlgHandle, BCRYPT_RNG_ALGORITHM, NULL, 0));
    if (!buffer.empty()) {
        ValidateStatus(L"ERR: The function BCryptGenRandom has failed.",
BCryptGenRandom(randAlgHandle, buffer.data(), static_cast<ULONG>(buffer.size()), 0));
    }
    BCryptCloseAlgorithmProvider(randAlgHandle, 0);
    std::wcout << L"MSG: Random data has been generated:" << std::endl;
    DisplayHex(buffer.data(), buffer.size());
}

std::vector<BYTE> CreateKey() {
    AnnounceOperation(L"Creating the encryption key.");
    BCRYPT_ALG_HANDLE algHandle;
    ValidateStatus(L"ERR: The function BCryptOpenAlgorithmProvider has failed.",
BCryptOpenAlgorithmProvider(&algHandle, BCRYPT_SHA256_ALGORITHM, NULL, 0));
    BCRYPT_HASH_HANDLE hashHandle;
    ValidateStatus(L"ERR: The function BCryptCreateHash has failed.",
BCryptCreateHash(algHandle, &hashHandle, NULL, 0, NULL, 0, 0));
    ValidateStatus(L"ERR: The function BCryptHashData has failed.",
BCryptHashData(hashHandle, reinterpret_cast<PUCHAR>(const_cast<wchar_t*>(key.c_str())),
static_cast<ULONG>(key.size() * sizeof(wchar_t)), 0));
    std::vector<BYTE> keyBuffer(32);
    ValidateStatus(L"ERR: The function BCryptFinishHash has failed.",
BCryptFinishHash(hashHandle, keyBuffer.data(), static_cast<ULONG>(keyBuffer.size()), 0));
    BCryptDestroyHash(hashHandle);
    BCryptCloseAlgorithmProvider(algHandle, 0);
    std::wcout << L"MSG: The encryption key is:" << std::endl;
    DisplayHex(keyBuffer.data(), keyBuffer.size());
    return keyBuffer;
}

std::vector<BYTE> ReadFileContent() {
    std::ifstream fileStream(file, std::ios::binary);
    if (!fileStream.is_open()) {
        throw std::runtime_error("ERR: Unable to open the file for reading.");
    }
    std::vector<BYTE> content((std::istreambuf_iterator<char>(fileStream)),
std::istreambuf_iterator<char>());
    return content;
}

void WriteFileContent(const std::vector<BYTE>& content, const wchar_t* outputFileName) {
    std::ofstream fileStream(outputFileName, std::ios::binary);
    if (!fileStream.is_open()) {
        throw std::runtime_error("ERR: Unable to open the file for writing.");
    }
    fileStream.write(reinterpret_cast<const char*>(content.data()), content.size());
}

void LockFile() {
    AnnounceOperation(L"Locking the file.");
    std::vector<BYTE> fileContent = ReadFileContent();
    std::vector<BYTE> key = CreateKey();

```

```

std::vector<BYTE> iv(16);
GenerateRandomData(iv);
std::vector<BYTE> originalIV = iv;
BCRYPT_ALG_HANDLE encAlgHandle;
ValidateStatus(L"ERR: The function BCryptOpenAlgorithmProvider has failed.",
BCryptOpenAlgorithmProvider(&encAlgHandle, BCRYPT_AES_ALGORITHM, NULL, 0));
ValidateStatus(L"ERR: The function BCryptSetProperty has failed.",
BCryptSetProperty(encAlgHandle, BCRYPT_CHAINING_MODE,
(PBYTE)BCRYPT_CHAIN_MODE_CBC, sizeof(BCRYPT_CHAIN_MODE_CBC), 0));
BCRYPT_KEY_HANDLE keyHandle;
ValidateStatus(L"ERR: The function BCryptGenerateSymmetricKey has failed.",
BCryptGenerateSymmetricKey(encAlgHandle, &keyHandle, NULL, 0, key.data(),
static_cast<ULONG>(key.size()), 0));
DWORD cbCipherText = 0;
ValidateStatus(L"ERR: The function BCryptEncrypt (determine size) has failed.",
BCryptEncrypt(
    keyHandle,
    fileContent.data(),
    static_cast<ULONG>(fileContent.size()),
    NULL,
    iv.data(),
    static_cast<ULONG>(iv.size()),
    NULL,
    0,
    &cbCipherText,
    BCRYPT_BLOCK_PADDING
));
std::vector<BYTE> cipherText(cbCipherText);
ValidateStatus(L"ERR: The function BCryptEncrypt has failed.", BCryptEncrypt(
    keyHandle,
    fileContent.data(),
    static_cast<ULONG>(fileContent.size()),
    NULL,
    iv.data(),
    static_cast<ULONG>(iv.size()),
    cipherText.data(),
    cbCipherText,
    &cbCipherText,
    BCRYPT_BLOCK_PADDING
));
BCryptDestroyKey(keyHandle);
BCryptCloseAlgorithmProvider(encAlgHandle, 0);
std::vector<BYTE> encryptedContent;
encryptedContent.insert(encryptedContent.end(), originalIV.begin(), originalIV.end());
encryptedContent.insert(encryptedContent.end(), cipherText.begin(), cipherText.end());
std::wstring outputFileName = std::wstring(file) + L".locked";
WriteFileContent(encryptedContent, outputFileName.c_str());
std::wcout << L"MSG: The file has been locked successfully." << std::endl;
}

```

```

void UnlockFile() {
    AnnounceOperation(L"Unlocking the file.");
    std::vector<BYTE> encryptedContent = ReadFileContent();
}

```

```

    if (encryptedContent.size() < 16) {
        throw std::runtime_error("ERR: The file format is invalid.");
    }
    std::vector<BYTE> iv(encryptedContent.begin(), encryptedContent.begin() + 16);
    std::wcout << L"MSG: The initialization vector (IV) is:" << std::endl;
    DisplayHex(iv.data(), iv.size());
    std::vector<BYTE> cipherText(encryptedContent.begin() + 16, encryptedContent.end());
    std::vector<BYTE> key = CreateKey();
    BCrypt_ALG_HANDLE decAlgHandle;
    ValidateStatus(L"ERR: The function BCryptOpenAlgorithmProvider has failed.",
    BCryptOpenAlgorithmProvider(&decAlgHandle, BCRYPT_AES_ALGORITHM, NULL, 0));
    ValidateStatus(L"ERR: The function BCryptSetProperty has failed.",
    BCryptSetProperty(decAlgHandle, BCRYPT_CHAINING_MODE,
    (PBYTE)BCRYPT_CHAIN_MODE_CBC, sizeof(BCRYPT_CHAIN_MODE_CBC), 0));
    BCrypt_KEY_HANDLE keyHandle;
    ValidateStatus(L"ERR: The function BCryptGenerateSymmetricKey has failed.",
    BCryptGenerateSymmetricKey(decAlgHandle, &keyHandle, NULL, 0, key.data(),
    static_cast<ULONG>(key.size()), 0));
    std::vector<BYTE> decryptedText(cipherText.size());
    DWORD cbData = 0;
    NTSTATUS status = BCryptDecrypt(
        keyHandle,
        cipherText.data(),
        static_cast<ULONG>(cipherText.size()),
        NULL,
        iv.data(),
        static_cast<ULONG>(iv.size()),
        decryptedText.data(),
        static_cast<ULONG>(decryptedText.size()),
        &cbData,
        BCRYPT_BLOCK_PADDING
    );
    if (MSAES256FE_SUCCESS(status)) {
        BCryptDestroyKey(keyHandle);
        BCryptCloseAlgorithmProvider(decAlgHandle, 0);
        decryptedText.resize(cbData);
        std::wstring outputFileName = std::wstring(file) + L".unlocked";
        WriteFileContent(decryptedText, outputFileName.c_str());
        std::wcout << L"MSG: The file has been unlocked successfully." << std::endl;
        return;
    }
    BCryptDestroyKey(keyHandle);
    BCryptCloseAlgorithmProvider(decAlgHandle, 0);
    ShowError(L"ERR: The function BCryptDecrypt has failed.", status);
    throw std::runtime_error("ERR: Unlocking the file has failed.");
}
};

int wmain(int argc, wchar_t* argv[]) {
    try {
        if (argc != 4) {
            std::wcerr << L"ERR: Usage: " << argv[0] << L" <filename> <password> <lock/unlock>" <<
            std::endl;

```

```

        return 1;
    }
    MSAES256_FileEncryption MSAES256_FileEncryption(argv[1], argv[2], argv[3]);
    MSAES256_FileEncryption.Execute();
}
catch (const std::exception& ex) {
    std::cerr << "ERR: An error occurred: " << ex.what() << std::endl;
    return 1;
}

return 0;
}
#endif // MSAES256FE_SECURE_H

```

Usage:

Code MS-AES-256-FE.exe <filename> <password> <lock/unlock>

Microsoft-AES-256-File-Encryption-x86.exe <filename> <password> <lock/unlock>

Microsoft-AES-256-File-Encryption.exe <filename> <password> <lock/unlock>

Encrypt / Lock a file:

Code MS-AES-256-FE.exe ./test.txt hello-world lockThis will create ./test.txt.locked

Decrypt / Unlock a file:

Code MS-AES-256-FE.exe ./test.txt.locked hello-world unlockThis will create
./test.txt.locked.unlocked

Downloads

Spoiler ([Click to View](#))

Quote:x64 (tested Win 11)

<https://github.com/LNodesL/files/blob/main/MS-AES-256-FE/dist/Microsoft-AES-256-File-Encryption.exe>

<https://ipfs.filebase.io/ipfs/QmStXBDA8aKx1rU37teoKjCxb3MNQE5aCuz4Xm2cqouNHh>

<https://ipfs.io/ipfs/QmStXBDA8aKx1rU37teoKjCxb3MNQE5aCuz4Xm2cqouNHh>

x86 (tested Win 11)

<https://github.com/LNodesL/files/blob/main/MS-AES-256-FE/dist/Microsoft-AES-256-File-Encryption-x86.exe>

<https://ipfs.filebase.io/ipfs/QmfWqGfQCKcFYbVBWhFR1tjtVewjYwnkvgm22nBPb4j8qE>

<https://ipfs.io/ipfs/QmfWqGfQCKcFYbVBWhFR1tjtVewjYwnkvgm22nBPb4j8qE>

Inspired by AESCrypt :

Quote:AESEncrypt implementation of Microsoft Cryptography API, encrypt/decrypt with AES-256 from a passphrase. <https://github.com/hackerhouse-opensource/AESEncrypt>

AI Generated Description of Source Code:

Spoiler ([Click to View](#))

MSAES256_FileEncryption Guide

This guide explains the functionality and usage of the MSAES256_FileEncryption class for encrypting and decrypting files using the AES-256 algorithm.

Overview

This class provides functionality for encrypting and decrypting files using the AES-256 encryption algorithm provided by the Windows CNG (Cryptography Next Generation) API. The program is designed to be run as a console application, taking three command-line arguments: the filename, the encryption/decryption password, and the operation (either "lock" or "unlock").

File Structure and Key Components

Header Guards and Includes:

The file begins with header guards to prevent multiple inclusions. It includes necessary headers for Windows API, CNG, and standard C++ libraries for I/O operations, exception handling, and data manipulation.

Preprocessor Directives:

Code `#pragma comment(lib, "bcrypt.lib")` ensures that the bcrypt library is linked.

Macros:

Code `#define MSAES256FE_SUCCESS(Status) ((NTSTATUS)(Status) >= 0)` is used to check the success status of CNG function calls.

Class Definition:

Code class MSAES256_FileEncryption encapsulates the functionality for file encryption and decryption.

Public Interface:

The constructor initializes the file path, encryption key, and operation type.

Code void Execute() handles the execution of the appropriate operation (encryption or decryption).

Private Methods: Code void ShowError(const wchar_t* msg, NTSTATUS status): Displays error messages.

Code void ValidateStatus(const wchar_t* msg, NTSTATUS status): Validates the status of operations and throws exceptions if they fail.

Code void AnnounceOperation(const wchar_t* operation): Announces the operation being performed.

Code void DisplayHex(const void* data, size_t size): Displays data in hexadecimal format.

Code void GenerateRandomData(std::vector<BYTE>& buffer): Generates random data for the initialization vector (IV).

Code std::vector<BYTE> CreateKey(): Creates the encryption key from the provided password using SHA-256 hashing.

Code std::vector<BYTE> ReadFileContent(): Reads the content of the specified file.

Code void WriteFileContent(const std::vector<BYTE>& content, const wchar_t* outputFileName): Writes data to a specified file.

Code void LockFile(): Encrypts the file content and writes the encrypted data along with the IV to a new file.

Code void UnlockFile(): Decrypts the file content and writes the decrypted data to a new file.

Main Function:

Code int wmain(int argc, wchar_t* argv[]): Entry point of the application. It validates the command-line arguments and invokes the MSAES256_FileEncryption class to perform the specified

operation.

Example Usage

To use this program, compile it as a console application. The program expects three command-line arguments: the file to be encrypted or decrypted, the password, and the operation type ("lock" for encryption and "unlock" for decryption). For example:

```
Code MSAES256_FileEncryption.exe example.txt mypassword lock
MSAES256_FileEncryption.exe example.txt.locked mypassword unlock
```

Notes

Ensure that the bcrypt library is available and properly linked.

The program is designed to run on Windows and utilizes the Windows CNG API for cryptographic operations.

The application handles errors gracefully, displaying informative messages when operations fail.

By following these guidelines, you should be able to compile and run this program in a 100% static configuration for both x64 and x32 platforms in Visual Studio.

If you like this small, single-file solution to protect your files by locking & unlocking / encrypting with a passphrase, using MS AES-256 encryption then please let me know if you have any questions or comments in this thread.

Dystortion - July 20, 2024, 1:59 pm

Great post mate. I've never delved much in cryptography so I'll be sure to add this to my notes. Thank you

Tsar - July 20, 2024, 4:53 pm

Fantastic nodes, great share and very helpful/useful. Appreciate it brother

Hackers - July 21, 2024, 4:37 am

(Jul 20, 2024 01:59 PM)Dystortion Wrote: [\[CITE\]](#) Great post mate. I've never delved much in cryptography so I'll be sure to add this to my notes. Thank you

(Jul 20, 2024 04:53 PM)Rapid Wrote: [\[CITE\]](#) Fantastic nodes, great share and very helpful/useful. Appreciate it brother

Thanks guys, just wanted to do something - had been a bit since I posted any C++ code

It's so easy to protect a file too, but many people save passwords and keys in plaintext. My hope is that someone can use this for their own security. The other thing is that it's not relying on a lesser-known AES library like my previous file encryption codebase I shared on HF.

secret_code - July 23, 2024, 5:32 am

(Jul 21, 2024 04:37 AM)nodes Wrote: [\[CITE\]](#) Thanks guys, just wanted to do something - had been a bit since I posted any C++ code

It's so easy to protect a file too, but many people save passwords and keys in plaintext. My hope is that someone can use this for their own security. The other thing is that it's not relying on a lesser-known AES library like my previous file encryption codebase I shared on HF.

nice one, i've been following you so far(not stalking, just following your content) and it looks very genuine to me, code looks ok, documentation looks ok, i also noticed something that i also do, im referring to use gpt for documentation, i do it sometimes not always, maybe when i have no time or i didn't write any comment at all and the code is large then i would use gpt for the documentation cuz im lazy, but this is the only thing i can do with gpt, for coding snippets i actually NEVER saw a good gpt poc :).

about the code, i would separate lock and unlock into multiple units, this is just my personal opinion, im not holding any fact just opinions, the reason is because sometimes you only need 1 or the other, and if for example this gets injected by the mean of steganography byte poisoning into a .png(.jpeg or .jpg might not work as intended because they're lossy and for stego poisoning you need lossless compression), the code needs to be small, that is the case where you might prefer to have only one module for unit, but i guess that wasn't the purpose anyway, im just making a concept, this concept doesn't mean that x is better then y but only a concept reflecting a point of view of the context based on events, i do also agree that sometimes having all in one unit is good, i would like to tell you that i appreciate your huge contribution on the forum, i see you in many places all over and everything you share seems genuine... !

Edited: i edit this because i just saw your source code, with all respect, i would like to give you some insights on where you can improve or correct code, im just trying to help not criticizing the code:

1) when initializing variables either mutable or constant, use direct initialization instead of copy initialization, the reason is because copy initialization requires more operations then direct initialization, at high level this isn't obvious because they seem to be the same.

2) not proper const usage, use const if you can, const data goes in ".rodata" registers, means it is faster to read, being allocated on the stack, stack is faster then heap but i guess u already know. when you allocate const data, you're referring to ".rodata" register sectors which are faster bcz they get allocated on faster cache line like L1 caches, which is the fastest cache line but smaller, while read-write data goes on ".rdata" which is slower cuz it gets allocated on slower caches like L2, L3...

3) you're missing some good attributes, there are attributes u can use like `__attribute__((ATTRIBUTE_NAME))` or if you prefer more modern and newer syntax you can go with `[[ATTRIBUTE_NAME]]` variant, both are the same but `__attribute__((x))` has more options available, the only weak point is compiler support, both clang and gcc have different options available, so you need to be careful what you're with `__attribute__((x))`, use `[[ATTR_NAME]]` instead, look at the following example:

Code // good code but not so performant...

```
int16_t x(0);
if(x > 0) {
    ... something
}
```

// same code but more performant

```
if(x > 0) [[likely]] {
    ... something
}
```

// OR

```
if(x < 0) [[unlikely]] { // unlikely because x is signed but we know that is highly improbable that will be
    < 0
    ... something
}
```

// this is called compiler optimization, and the compiler will thank you for this ;)

another example will be with function inline optimization:

```
Code // weaker syntax
static bool FrequentCheck() {
return true;
}
// let's say FrequentCheck is a small function performing easy tasks that wont throw any error...
__attribute__((hot)) [[nodiscard]] static const bool FrequentCheck() noexcept {
return true;
}
// if the function is not called so often...
__attribute__((gnu::cold)) [[nodiscard]] static const bool FrequentCheck() noexcept {
return true;
}
```

there are a dozen of possible attributes you can use to improve the code... obviously i know that the compiler does improvements all over the code, but does not always make the best decisions because it doesn't know exactly where and how the code behaves in certain ways, also.... it will be harder to read and maintain but this is the only way you ensure best performance for your code... especially for cryptographic operations that require quite more computations... but don't worry, i actually encounter very often c/c++ programmers who know the existence of inline optimization attributes, what i see is just like 1/10 actually do use them, and if they do, they do it wrong way, i've seen things like `__attribute__((hot))` and `[[gnu::cold]]` together.. which doesn't make any sense because one is saying "high frequency execution" while the other "low frequency execution", if you want to use them, make sure they're correct :), thank you for your understanding, im just another crazy lazy motherfucker :)

Hackers - July 23, 2024, 5:42 am

(Jul 23, 2024 05:32 AM)secret_code Wrote: [\[CITE\]](#) nice one, i've been following you so far(not stalking, just following your content) and it looks very genuine to me, code looks ok, documentation looks ok, i also noticed something that i also do, im referring to use gpt for documentation, i do it sometimes not always, maybe when i have no time or i didn't write any comment at all and the code is large then i would use gpt for the documentation cuz im lazy, but this is the only thing i can do with gpt, for coding snippets i actually NEVER saw a good gpt poc :).

about the code, i would separate lock and unlock into multiple units, this is just my personal opinion, im not holding any fact just opinions, the reason is because sometimes you only need 1 or the other, and if for example this gets injected by the mean of steganography byte poisoning into a .png(.jpeg or .jpg might not work as intended because they're lossy and for stego poisoning you need lossless compression), the code needs to be small, that is the case where you might prefer to have only one module for unit, but i guess that wasn't the purpose anyway, im just making a concept, this concept doesn't mean that x is better then y but only a concept reflecting a point of view of the context based on events, i do also agree that sometimes having all in one unit is good, i would like to tell you that i appreciate your huge contribution on the forum, i see you in many places all over and everything you share seems genuine... !

Thanks for your comment!

I did use GPT to refactor some of this from that AESCrypt library. You are right that the compression & exact methods could be separated out. I treated it as more like on/off or encrypt/decrypt and kept it all in one.

For me, the thing I got out of this was using bcrypt lib instead of others like I have in past:

I had used a custom, but not well known AES library in past for SFED (Single File Encryption & Decryption): <https://github.com/LNodesL/SFED-AES-256>

I then used the same AES lib for NCrypt: <https://github.com/LNodesL/NCrypt>
I used XOR for Sonofa - <https://github.com/LNodesL/Sonofa-Byte-Crypting-Program>
XOR in Espana - <https://github.com/LNodesL/EspanaKrypt>
I have used a ton of other algos in other langs, but hadn't used bcrypt in C++ yet.

I like to pick mini projects to try new things, and sometimes I'll release on HF for the sole sake of sharing because otherwise it'll end up in an archive folder on my PC, never to be opened again.
I appreciate your kind words!! I'm glad people find my content useful. I'm currently juggling my research into the Sumerian tablets with learning C++, and working on some other projects too.

secret_code - July 23, 2024, 6:26 am
(Jul 23, 2024 05:42 AM)nodes Wrote: [\[CITE\]](#) Thanks for your comment!
I did use GPT to refactor some of this from that AESCrypt library. You are right that the compression & exact methods could be separated out. I treated it as more like on/off or encrypt/decrypt and kept it all in one.

For me, the thing I got out of this was using bcrypt lib instead of others like I have in past:

I had used a custom, but not well known AES library in past for SFED (Single File Encryption & Decryption): <https://github.com/LNodesL/SFED-AES-256>
I then used the same AES lib for NCrypt: <https://github.com/LNodesL/NCrypt>
I used XOR for Sonofa - <https://github.com/LNodesL/Sonofa-Byte-Crypting-Program>
XOR in Espana - <https://github.com/LNodesL/EspanaKrypt>
I have used a ton of other algos in other langs, but hadn't used bcrypt in C++ yet.

I like to pick mini projects to try new things, and sometimes I'll release on HF for the sole sake of sharing because otherwise it'll end up in an archive folder on my PC, never to be opened again.
I appreciate your kind words!! I'm glad people find my content useful. I'm currently juggling my research into the Sumerian tablets with learning C++, and working on some other projects too.
no problem, im trying to give some of the contribution, that's all, exchanging opinions is always a smart thing to do, who don't share opinions will never make progress in my point of view, so is important, i edited the previous comment as i was reading the source code i spotted different areas with possible space for improvement, again, just optimizations not corrections... you're doing good actually...

Hackers - August 9, 2024, 6:40 am
(Jul 23, 2024 06:26 AM)secret_code Wrote: [\[CITE\]](#) no problem, im trying to give some of the contribution, that's all, exchanging opinions is always a smart thing to do, who don't share opinions will never make progress in my point of view, so is important, i edited the previous comment as i was reading the source code i spotted different areas with possible space for improvement, again, just optimizations not corrections... you're doing good actually...
agreed, always good to share knowledge and discuss
Shame you are no longer with us and it looks like you requested to be banned permanently. :(

Stanley - October 28, 2025, 6:26 pm
I'm Stanley HF's helper bot. The OP (original poster) of this thread wants everyone to see this page and has elected to bump it to the top of the showthread listing. Since HF doesn't want a ton of LQP (Low Quality Posts) this is the best method of bumping your content. There is a small ?yte fee for each bump. Any OP can bump any of their threads on HF from the linked icon on the top of their first post.

If you haven't already read the first post of this thread we encourage you to do so.

Thanks,
Stanley

Hackers - *March 16, 2026, 8:48 am*

Happy to help explain anything if someone has a question!