

C++ static code analysis: Cipher algorithms should be robust

5-6 minutes

[Strong cipher algorithms](#) are cryptographic systems resistant to cryptanalysis, they are not vulnerable to well-known attacks like brute force attacks for example.

A general recommendation is to only use cipher algorithms intensively tested and promoted by the cryptographic community.

More specifically for block cipher, it's not recommended to use algorithm with a block size inferior than 128 bits.

Noncompliant Code Example

[botan](#)

```
#include <botan/cipher_mode.h>
```

```
Botan::Cipher_Mode::create("Blowfish  
/CBC/PKCS7", Botan::ENCRYPTION);    //
```

Noncompliant: Blowfish use a 64-bit block size makes it vulnerable to birthday attacks

```
Botan::Cipher_Mode::create("DES/CBC  
/PKCS7", Botan::ENCRYPTION);        //
```

Noncompliant: DES works with 56-bit keys allow attacks via exhaustive search

```
Botan::Cipher_Mode::create("3DES/CBC  
/PKCS7", Botan::ENCRYPTION);        //
```

Noncompliant: Triple DES is vulnerable to meet-in-the-middle attack

```
Botan::Cipher_Mode::create("DESX/CBC  
/PKCS7", Botan::ENCRYPTION);        //
```

Noncompliant: Triple DES is vulnerable to meet-in-the-middle attack

```
Botan::Cipher_Mode::create("CAST-128/CBC  
/PKCS7", Botan::ENCRYPTION);    //
```

Noncompliant: 64-bit size block cipher
Botan::Cipher_Mode::create("GOST-
28147-89/CBC/PKCS7",
Botan::ENCRYPTION); // Noncompliant: 64-
bit size block cipher
Botan::Cipher_Mode::create("IDEA/CBC
/PKCS7", Botan::ENCRYPTION); //
Noncompliant: 64-bit size block cipher
Botan::Cipher_Mode::create("KASUMI/CBC
/PKCS7", Botan::ENCRYPTION); //
Noncompliant: 64-bit size block cipher
Botan::Cipher_Mode::create("MISTY1/CBC
/PKCS7", Botan::ENCRYPTION); //
Noncompliant: 64-bit size block cipher
Botan::Cipher_Mode::create("XTEA/CBC
/PKCS7", Botan::ENCRYPTION); //
Noncompliant: 64-bit size block cipher
Botan::Cipher_Mode::create("RC4",
Botan::ENCRYPTION); //
Noncompliant: has numerous design flaws
which make it hard to use correctly

crypto++

```
#include <cryptopp/arc4.h>
#include <cryptopp/blowfish.h>
#include <cryptopp/cast.h>
#include <cryptopp/des.h>
#include <cryptopp/gost.h>
#include <cryptopp/idea.h>
#include <cryptopp/rc2.h>
#include <cryptopp/tea.h>
```

CryptoPP::ARC4::Encryption(key,
sizeof(key)); // Noncompliant: RC4/ARC4 has
numerous design flaws which make it hard to
use correctly

CryptoPP::Blowfish::Encryption(key,
sizeof(key)); // Noncompliant: 64-bit size
block

CryptoPP::GOST::Encryption(key,
sizeof(key)); // Noncompliant: 64-bit size
block

CryptoPP::IDEA::Encryption(key, sizeof(key));
// Noncompliant: 64-bit size block

CryptoPP::XTEA::Encryption(key,

```
sizeof(key)); // Noncompliant: 64-bit size
block
CryptoPP::DES::Encryption(key, sizeof(key));
// Noncompliant: DES works with 56-bit keys
allow attacks via exhaustive search
CryptoPP::DES_EDE2::Encryption(key,
sizeof(key)); // Noncompliant: Triple DES is
vulnerable to meet-in-the-middle attack
CryptoPP::DES_EDE3::Encryption(key,
sizeof(key)); // Noncompliant: Triple DES is
vulnerable to meet-in-the-middle attack
CryptoPP::DES_XEX3::Encryption(key,
sizeof(key)); // Noncompliant: Triple DES is
vulnerable to meet-in-the-middle attack
CryptoPP::RC2::Encryption(key, sizeof(key));
// Noncompliant: RC2 is vulnerable to a
related-key attack
CryptoPP::RC2Encryption(key, sizeof(key)); //
Noncompliant; alternative
CryptoPP::RC2Decryption(key, sizeof(key)); //
Noncompliant; alternative
```

[OpenSSL](#)

```
#include <openssl/evp.h>
```

```
EVP_bf_cbc(); // Noncompliant: 64-bit size  
block
```

```
EVP_cast5_cbc(); // Noncompliant: 64-bit size  
block
```

```
EVP_des_cbc(); // Noncompliant: DES works  
with 56-bit keys allow attacks via exhaustive  
search
```

```
EVP_idea_cbc(); // Noncompliant: 64-bit size  
block
```

```
EVP_rc4(); // Noncompliant: has numerous  
design flaws which make it hard to use  
correctly
```

```
EVP_rc2_cbc(); // Noncompliant: RC2 is  
vulnerable to a related-key attack
```

Compliant Solution

[botan](#)

```
#include <botan/cipher_mode.h>
```

```
Botan::Cipher_Mode::create("AES-
```

```
256/GCM", Botan::ENCRYPTION); //
```

Compliant: AES is a good default choice for symmetric encryption

[crypto++](#)

```
#include <cryptopp/aes.h>
```

```
CryptoPP::AES::Encryption(key, sizeof(key));  
// Compliant: AES is a good default choice for  
symmetric encryption
```

[OpenSSL](#)

```
#include <openssl/evp.h>
```

```
EVP_aes_128_gcm() // Compliant: AES is a  
good default choice for symmetric encryption
```

See

- [OWASP Top 10 2021 Category A2](#) - Cryptographic Failures
- [OWASP Top 10 2017 Category A3](#) - Sensitive Data Exposure

- [MITRE, CWE-327](#) - Use of a Broken or Risky Cryptographic Algorithm
- [SANS Top 25](#) - Porous Defenses