

Survey on Multi-language Design Smells

Thank you for agreeing to participate, it will take around 30 minutes to complete.

Study Policy:

- Participation in this study is completely voluntary. If you decide not to participate there will not be any negative consequences. If you decide to participate, you may stop participating at any time and withdraw entirely your participation or you may decide not to answer any specific question.
- Your identity and the data collected thanks to your participation will remain anonymous and will never be released to the public. Only anonymous data (aggregated or not) will be published in scientific articles, ensuring that the data cannot be linked back to a particular participant. The data will be kept by the principal investigator for five years before being destroyed.
- By submitting this survey, you are indicating that you have read the description of the study, are over the age of 18, and that you agree to the terms and consent as described in https://drive.google.com/file/d/1aZfHRCr0bEX0i33I_oQHIS9ui9h6rIC5/view?usp=sharing

If you have any questions, please contact us at mouna.abidi@polymtl.ca

Study Design: The purpose of this study is to investigate the prevalence of design smells related to multi-language systems. These systems are developed using more than one programming language. We aim to investigate the perceived prevalence and impact of the design smells detailed below. Our main goal is to improve the quality of those systems.

Definition of terminologies:

Not Handling Exceptions	The exceptions are not handled, developers generally rely on the exceptions provided by the other language
Assuming Safe Return Value	A value is returned to the other language without being checked. Thus, the interaction between both languages may not be correctly performed
Excessive Inter-language Communication	A wrong partitioning in both languages leads to many calls in a way or the other. It adds complexity takes more time to run and may indicate a bad separation of concerns
Too Much Clustering	The multi-language code is concentrated in a few classes, regardless of their concerns and responsibilities.
Too Much Scattering	Many classes are scarcely used in multi-language communication
Hard Coding Libraries	When different libraries are needed depending on the operating system, they are not loaded with conditions on the operating system, but for instance, with a try-catch mechanism, making it hard to know which library has really been loaded
Local References Abuse	The developer does not manage the memory in the native space properly and does not release local and global references
Memory Management Mismatch	Reference types passed from one language to another are not released in a language that does not handle the management of memory causing memory leaks
Not Caching Objects	A method is called to retrieve a field every time this field is needed, although the field's ID or value could have been cached.
Not Securing Libraries	The code loads a foreign library without any security check or restriction privilege
Not Using Relative Path	A library is loaded using only the name not the path. It cannot be accessed in the same way from everywhere
Excessive Objects	A whole object is passed as an argument, although only some of the fields were needed, and it would have been better for the system performance to pass only these fields
Unused Method Declaration	A method is declared in the host language but not implemented in the foreign language
Unused Method Implementation	A method is declared in the host language and implemented in the foreign language, but never called from the host language
Unused Parameters	Some arguments of a function are used neither in its body nor in the other language.

IEEE.)

- Expandability: The degree to which the design of a system can be extended.
- Simplicity: The degree to which the design of a system can be understood easily.
- Reusability: The degree to which a piece of design can be reused in another design.
- Learnability: The degree to which the code source of a system is easy to learn.
- Understandability: The degree to which the code source can be understood easily.
- Performance: The degree to which the code meets its requirements for timeliness.
- Modularity: The degree to which the implementation of the functions of a system is independent of one another.

Thank you.

Best regards,

*** 1. How often do you encounter the following design smells in your project(s)?**

Please check the definitions provided above before answering this questions

	1 Very Often	2 Often	3 Rarely	N/A
Not Handling Exceptions	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Assuming Safe Return Value	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Excessive Inter-language Communication	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Too Much Clustering	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Too Much Scattering	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Hard Coding Libraries	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Local References Abuse	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Memory Management Mismatch	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Not Caching Objects	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Not Securing Libraries	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Not Using Relative Path	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Excessive Objects	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Unused Method Declaration	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Unused Method Implementation	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Unused Parameters	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

*** 2. How do you evaluate the impact of the following design smells in those software quality attributes?**

Please carefully read the definition of the smells provided below and the reference provided.
(VN: Very Negative, N: Negative, NS: Not significant/Neutral, P: Positive, and VP: Very Positive)

	Expandability	Simplicity	Reusability	Learnability	Understandability	Performance	Modularity	N/A
Not Handling Exceptions	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="radio"/>
Assuming Safe Return Value	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="radio"/>
Excessive Inter-language Communication	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="radio"/>
Too Much Clustering	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="radio"/>
Too Much Scattering	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="radio"/>
Hard Coding Libraries	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="radio"/>
Local References Abuse	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="radio"/>
Memory Management Mismatch	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="radio"/>
Not Caching Objects	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="radio"/>
Not Securing Libraries	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="radio"/>
Not Using Relative Path	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="radio"/>
Excessive Objects	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="radio"/>
Unused Method Declaration	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="radio"/>
Unused Method Implementation	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="radio"/>
Unused Parameters	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="radio"/>

*** 3. Please rank the following design smells from the most harmful to the less harmful**

(Most harmful to the less harmful: 15 -> 1)

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15

Not Handling Exceptions

Assuming Safe Return Value

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15

Excessive Inter-language Communication

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15

Too Much Clustering

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15

Too Much Scattering

Hard Coding Libraries

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

Local References Abuse

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

Memory Management Mismatch

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

Not Caching Objects

Not Securing Libraries

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

Not Using Relative Path

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

Excessive Objects

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

Unused Method Declaration

Unused Method Implementation

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

Unused Parameters

* 4. Task:

a) In your opinion, does the following code(s) contain any occurrence of design smell(implementation and-or design problem)?

```
inline jfieldID getFieldRef(JNIEnv* env, jclass clazz, const char* name, const char* sig) {
jfieldID localField = env->GetFieldID(clazz, name, sig);
return localField;
}
```

☐ Yes

☐ No

5. b) If YES, please provide an explanation or specify the design smell(s) involved?

6. c) If YES, (In your opinion,) What is the motivation behind using this specific way of implementation?

* 7. d) Please rate the severity of the implementation problem (if any), from 1 (Very Low) to 5 (Very High)

1
Very Low

2
Low

3
Medium

4
High

5
Very High

N/A



*** 8. e) If YES, would you apply this refactored solution?**

```
inline jfieldID getFieldRef(JNIEnv* env, jclass clazz, const char* name, const char* sig) {  
    jfieldID localField = env->GetFieldID(clazz, name, sig);  
    if (localField == nullptr) {  
        CONSCRYPT_LOG_ERROR("could not find field %s", name);  
        abort();  
    }  
    return localField;  
}
```



Yes (Refactor with this solution)



Yes (Refactor with an alternative solution)



No (No refactoring)

*** 9. Task:**

a) In your opinion, does the following code(s) contain any occurrence of design smell(implementation and-or design problem)?

```
static jlong NativeCrypto_X509_dup(JNIEnv* env, jclass, jlong x509Ref, CONSCRYPT_UNUSED jobject holder) {  
    CHECK_ERROR_QUEUE_ON_RETURN;  
    X509* x509 = reinterpret_cast<X509*>(static_cast<jlong>(x509Ref));  
    JNI_TRACE("X509_dup(%p)", x509);  
  
    if (x509 == nullptr) {  
        conscrypt::jniutil::throwNullPointerException(env, "x509 == null");  
        JNI_TRACE("X509_dup(%p) => x509 == null", x509);  
        return 0;  
    }  
  
    return reinterpret_cast<jlong>(X509_dup(x509));  
}
```



Yes



No

10. b) If YES, please provide an explanation or specify the design smell(s) involved?

11. c) If YES, (In your opinion,) What is the motivation behind using this specific way of implementation?

*** 12. d) Please rate the severity of the implementation problem (if any), from 1 (Very Low) to 5 (Very High)**

1
Very Low

2
Low

3
Medium

4
High

5
Very High

N/A



*** 13. e) If YES, would you apply this refactored solution?**

```
static jlong NativeCrypto_X509_dup(JNIEnv* env, jclass, jlong x509Ref) {  
    CHECK_ERROR_QUEUE_ON_RETURN;  
    X509* x509 = reinterpret_cast<static_cast<x509Ref>());  
    JNI_TRACE("X509_dup(%p)", x509);  
  
    if (x509 == nullptr) {  
        conscrypt::jniutil::throwNullPointerException(env, "x509 == null");  
        JNI_TRACE("X509_dup(%p) => x509 == null", x509);  
        return 0;  
    }  
  
    return reinterpret_cast<X509_dup(x509)>();  
}
```

☐ Yes (Refactor with this solution)

☐ Yes (Refactor with an alternative solution)

☐ No (No refactoring)

*** 14. Task:**

a) In your opinion, does the following code(s) contain any occurrence of design smell(implementation and-or design problem)?

```
public static void init() {  
    System.loadLibrary("jvaccrypto");  
}
```

☐ Yes

☐ No

15. b) If YES, please provide an explanation or specify the design smell(s) involved?

16. c) If YES, (In your opinion,) What is the motivation behind using this specific way of implementation?

*** 17. d) Please rate the severity of the implementation problem (if any), from 1 (Very Low) to 5 (Very High)**

1
Very Low



2
Low



3
Medium



4
High



5
Very High



N/A



* 18. e) If YES, would you apply this refactored solution?

```
public static void loadLibrary
static {
AccessController.doPrivileged( new PrivilegedAction() {
public static void init() {
System.loadLibrary("jvaccrypto");
}} }
```

- ☐ Yes (Refactor with this solution) ☐ Yes (Refactor with an alternative solution)
- ☐ No (No refactoring)

* 19. Task:

a) In your opinion, does the following code(s) contain any occurrence of design smell(implementation and-or design problem)?

```
inline jclass getGlobalRefToClass(JNIEnv* env, const char* className) {
ScopedLocalRef localClass(env, env->FindClass(className));
jclass globalRef = reinterpret_cast(env->NewGlobalRef(localClass.get()));

return globalRef;
}
```

- ☐ Yes ☐ No

20. b) If YES, please provide an explanation or specify the design smell(s) involved?

21. c) If YES, (In your opinion,) What is the motivation behind using this specific way of implementation?

* 22. d) Please rate the severity of the implementation problem (if any), from 1 (Very Low) to 5 (Very High)

1
Very Low

☐

2
Low

☐

3
Medium

☐

4
High

☐

5
Very High

☐

N/A

☐

* 23. e) If YES, would you apply this refactored solution?

```
inline jclass getGlobalRefToClass(JNIEnv* env, const char* className) {
ScopedLocalRef localClass(env, env->FindClass(className));
jclass globalRef = reinterpret_cast(env->NewGlobalRef(localClass.get()));
if (globalRef == nullptr) {
CONSCRYPT_LOG_ERROR("failed to find class %s", className);
abort();
}
```

```
return globalRef;
}
```

- ☐ Yes (Refactor with this solution)
 ☐ Yes (Refactor with an alternative solution)
 ☐ No (No refactoring)

*** 24. Task:**

a) In your opinion, does the following code(s) contain any occurrence of design smell(implementation and-or design problem)?

```
std::vector certBufferRefs(numCerts);
std::vector certBuffers(numCerts);
for (size_t i = 0; i < numCerts; ++i) {
  ScopedLocalRef certArray(env, reinterpret_cast(env->GetObjectArrayElement(encodedCertificatesJava, i)));
  certBufferRefs[i] = ByteArrayToCryptoBuffer(env, certArray.get(), nullptr);
  if (!certBufferRefs[i]) {
    return;
  }
  certBuffers[i] = certBufferRefs[i].get();
}
}
```

- ☐ Yes
 ☐ No

25. b) If YES, please provide an explanation or specify the design smell(s) involved?

26. c) If YES, (In your opinion,) What is the motivation behind using this specific way of implementation?

*** 27. d) Please rate the severity of the implementation problem (if any), from 1 (Very Low) to 5 (Very High)**

1
Very Low



2
Low



3
Medium



4
High



5
Very High



N/A



*** 28. e) If YES, would you apply this refactored solution?**

```
std::vector<bssl::unique_ptr> certBufferRefs(numCerts);
std::vector<crypto_buffer*> certBuffers(numCerts);
for (size_t i = 0; i < numCerts; ++i) {
  ScopedLocalRef certArray(
    env, reinterpret_cast(
      env->GetObjectArrayElement(encodedCertificatesJava, i)));
  certBufferRefs[i] = ByteArrayToCryptoBuffer(env, certArray.get(), nullptr);
  if (!certBufferRefs[i]) {
```

```

return;
}
certBuffers[i] = certBufferRefs[i].get();
(*env)->DeleteLocalRef(env, certBufferRefs);
}

```

- ☐ Yes (Refactor with this solution)
 ☐ Yes (Refactor with an alternative solution)
 ☐ No (No refactoring)

*** 29. Task:**

a) In your opinion, does the following code(s) contain any occurrence of design smell(implementation and-or design problem)?

```

bool setApplicationProtocols(JNIEnv* e, jbyteArray applicationProtocolsJava) {
  clearApplicationProtocols();
  if (applicationProtocolsJava != nullptr) {
    jbyte* applicationProtocols =
      e->GetByteArrayElements(applicationProtocolsJava, nullptr);
    if (applicationProtocols == nullptr) {
      clearCallbackState();
      JNI_TRACE("appData=%p setApplicationCallbackState => applicationProtocols == null", this);
      return false;
    }
    applicationProtocolsLength = static_cast(e->GetArrayLength(applicationProtocolsJava));
    applicationProtocolsData = new char[applicationProtocolsLength];
    memcpy(applicationProtocolsData, applicationProtocols, applicationProtocolsLength);
  }
  return true;
}

```

- ☐ Yes
 ☐ No

30. b) If YES, please provide an explanation or specify the design smell(s) involved?

31. c) If YES, (In your opinion,) What is the motivation behind using this specific way of implementation?

*** 32. d) Please rate the severity of the implementation problem (if any), from 1 (Very Low) to 5 (Very High)**

1 Very Low	2 Low	3 Medium	4 High	5 Very High	N/A
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

*** 33. e) If YES, would you apply this refactored solution?**

```

bool setApplicationProtocols(JNIEnv* e, jbyteArray applicationProtocolsJava) {
    clearApplicationProtocols();
    if (applicationProtocolsJava != nullptr) {
        jbyte* applicationProtocols =
            e->GetByteArrayElements(applicationProtocolsJava, nullptr);
        if (applicationProtocols == nullptr) {
            clearCallbackState();
            JNI_TRACE("appData=%p setApplicationCallbackState => applicationProtocols == null", this);
            return false;
        }
        applicationProtocolsLength =
            static_cast(e->GetArrayLength(applicationProtocolsJava));
        applicationProtocolsData = new char[applicationProtocolsLength];
        memcpy(applicationProtocolsData, applicationProtocols, applicationProtocolsLength);
        e->ReleaseByteArrayElements(applicationProtocolsJava, applicationProtocols, JNI_ABORT);
    }
    return true;
}

```

- ☐ Yes (Refactor with this solution)
 ☐ Yes (Refactor with an alternative solution)
 ☐ No (No refactoring)

*** 34. Task:**

a) In your opinion, does the following code(s) contain any occurrence of design smell(implementation and-or design problem)?

```

public static PrivateKey wrapKey(PrivateKey key) {
    if (key instanceof RSAPrivateKey) {
        return new OpaqueDelegatingRSAPrivateKey((RSAPrivateKey) key);
    } else if (key instanceof ECPrivateKey) {
        return new OpaqueDelegatingECPrivateKey((ECPrivateKey) key);
    } else {
        fail("Unknown key type: " + key.getClass().getName());
        return null;
    }
}

```

- ☐ Yes
 ☐ No

35. b) If YES, please provide an explanation or specify the design smell(s) involved?

36. c) If YES, (In your opinion,) What is the motivation behind using this specific way of implementation?

*** 37. d) Please rate the severity of the implementation problem (if any), from 1 (Very Low) to 5 (Very High)**

1
Very Low

2
Low

3
Medium

4
High

5
Very High

N/A

* 38. e) If YES, would you apply this refactored solution?

```
public static PrivateKey wrapKey(PrivateKey key) {
    if (key != null){
        if (key instanceof RSAPrivateKey) {
            return new OpaqueDelegatingRSAPrivateKey((RSAPrivateKey) key);
        } else if (key instanceof ECPrivateKey) {
            return new OpaqueDelegatingECPrivateKey((ECPrivateKey) key);
        } else {
            fail("Unknown key type: " + key.getClass().getName());
            return null;
        }
    }
}
```

- ☐ Yes (Refactor with this solution) ☐ Yes (Refactor with an alternative solution)
- ☐ No (No refactoring)

* 39. Task:

a) In your opinion, does the following code(s) contain any occurrence of design smell(implementation and-or design problem)?

```
// --- EVP ---
static native long EVP_get_cipherbyname(String string);
static native void EVP_CipherInit_ex(NativeRef.EVP_CIPHER_CTX ctx, long evpCipher, byte[] key,
byte[] iv, boolean encrypting);
static native int EVP_CipherUpdate(NativeRef.EVP_CIPHER_CTX ctx, byte[] out, int outOffset,
byte[] in, int inOffset, int inLength) throws IndexOutOfBoundsException;
static native int EVP_CipherFinal_ex(NativeRef.EVP_CIPHER_CTX ctx, byte[] out, int outOffset)
throws BadPaddingException, IllegalBlockSizeException;
static native int EVP_CIPHER_iv_length(long evpCipher);
static native long EVP_CIPHER_CTX_new();
static native int EVP_CIPHER_CTX_block_size(NativeRef.EVP_CIPHER_CTX ctx);
static native int get_EVP_CIPHER_CTX_buf_len(NativeRef.EVP_CIPHER_CTX ctx);
static native boolean get_EVP_CIPHER_CTX_final_used(NativeRef.EVP_CIPHER_CTX ctx);
static native void EVP_CIPHER_CTX_set_padding(
NativeRef.EVP_CIPHER_CTX ctx, boolean enablePadding);
static native void EVP_CIPHER_CTX_set_key_length(NativeRef.EVP_CIPHER_CTX ctx, int keyBitSize);
static native void EVP_CIPHER_CTX_free(long ctx);

// --- AEAD ---
static native long EVP_aead_aes_128_gcm();
static native long EVP_aead_aes_256_gcm();
static native long EVP_aead_chacha20_poly1305();
static native long EVP_aead_aes_128_gcm_siv();
static native long EVP_aead_aes_256_gcm_siv();
static native int EVP_AEAD_max_overhead(long evpAead);
static native int EVP_AEAD_nonce_length(long evpAead);
static native int EVP_AEAD_CTX_seal(long evpAead, byte[] key, int tagLengthInBytes, byte[] out,
int outOffset, byte[] nonce, byte[] in, int inOffset, int inLength, byte[] ad)
throws ShortBufferException, BadPaddingException;
static native int EVP_AEAD_CTX_seal_buf(long evpAead, byte[] key, int tagLengthInBytes, ByteBuffer out,
byte[] nonce, ByteBuffer input, byte[] ad)
throws ShortBufferException, BadPaddingException;
```

```

static native int EVP_AEAD_CTX_open(long evpAead, byte[] key, int tagLengthInBytes, byte[] out,
int outOffset, byte[] nonce, byte[] in, int inOffset, int inLength, byte[] ad)
throws ShortBufferException, BadPaddingException;
static native int EVP_AEAD_CTX_open_buf(long evpAead, byte[] key, int tagLengthInBytes, ByteBuffer out,
byte[] nonce, ByteBuffer input, byte[] ad)
throws ShortBufferException, BadPaddingException;
// --- PKCS7 ---
static native long[] X509_CRL_get_REVOKED(long x509CrlCtx, OpenSSLX509CRL holder);
static native String[] get_X509_CRL_ext_oids(long x509CrlCtx, OpenSSLX509CRL holder, int critical);
static native byte[] X509_CRL_get_ext_oid(long x509CrlCtx, OpenSSLX509CRL holder, String oid);
static native void X509_delete_ext(long x509, OpenSSLX509Certificate holder, String oid);
static native long X509_CRL_get_version(long x509CrlCtx, OpenSSLX509CRL holder);
static native long X509_CRL_get_ext(long x509CrlCtx, OpenSSLX509CRL holder, String oid);
static native byte[] get_X509_CRL_signature(long x509ctx, OpenSSLX509CRL holder);
static native void X509_CRL_verify(long x509CrlCtx, OpenSSLX509CRL holder,
NativeRef.EVP_PKEY pkeyCtx) throws BadPaddingException, SignatureException,
NoSuchAlgorithmException, InvalidKeyException, IllegalBlockSizeException;
static native byte[] get_X509_CRL_crl_enc(long x509CrlCtx, OpenSSLX509CRL holder);
static native long X509_CRL_get_lastUpdate(long x509CrlCtx, OpenSSLX509CRL holder);
static native long X509_CRL_get_nextUpdate(long x509CrlCtx, OpenSSLX509CRL holder);
static native long HMAC_CTX_new();
static native void HMAC_CTX_free(long ctx);
static native void HMAC_Init_ex(NativeRef.HMAC_CTX ctx, byte[] key, long evp_md);
static native void HMAC_Update(NativeRef.HMAC_CTX ctx, byte[] in, int inOffset, int inLength);
static native void HMAC_UpdateDirect(NativeRef.HMAC_CTX ctx, long inPtr, int inLength);
static native byte[] HMAC_Final(NativeRef.HMAC_CTX ctx);
static native long X509_REVOKED_dup(long x509RevokedCtx);
static native byte[] i2d_X509_REVOKED(long x509RevokedCtx);
static native String[] get_X509_REVOKED_ext_oids(long x509ctx, int critical);
static native byte[] X509_REVOKED_get_ext_oid(long x509RevokedCtx, String oid);
static native byte[] X509_REVOKED_get_serialNumber(long x509RevokedCtx);
static native long X509_REVOKED_get_ext(long x509RevokedCtx, String oid);

```

☐ Yes

☐ No

40. b) If YES, please provide an explanation or specify the design smell(s) involved?

41. c) If YES, (In your opinion,) What is the motivation behind using this specific way of implementation?

* 42. d) Please rate the severity of the implementation problem (if any), from 1 (Very Low) to 5 (Very High)

1
Very Low



2
Low



3
Medium



4
High



5
Very High



N/A



*** 43. e) If YES, would you apply this refactored solution?**

```
public class EVP {
    static native long EVP_get_cipherbyname(String string);
    static native void EVP_CipherInit_ex(NativeRef.EVP_CIPHER_CTX ctx, long evpCipher, byte[] key,
    byte[] iv, boolean encrypting);
    static native int EVP_CipherUpdate(NativeRef.EVP_CIPHER_CTX ctx, byte[] out, int outOffset,
    byte[] in, int inOffset, int inLength) throws IndexOutOfBoundsException;
    static native int EVP_CipherFinal_ex(NativeRef.EVP_CIPHER_CTX ctx, byte[] out, int outOffset)
    throws BadPaddingException, IllegalBlockSizeException;
    static native int EVP_CIPHER_iv_length(long evpCipher);
    static native long EVP_CIPHER_CTX_new();
    static native int EVP_CIPHER_CTX_block_size(NativeRef.EVP_CIPHER_CTX ctx);
    static native int get_EVP_CIPHER_CTX_buf_len(NativeRef.EVP_CIPHER_CTX ctx);
    static native boolean get_EVP_CIPHER_CTX_final_used(NativeRef.EVP_CIPHER_CTX ctx);
    static native void EVP_CIPHER_CTX_set_padding(
    NativeRef.EVP_CIPHER_CTX ctx, boolean enablePadding);
    static native void EVP_CIPHER_CTX_set_key_length(NativeRef.EVP_CIPHER_CTX ctx, int keyBitSize);
    static native void EVP_CIPHER_CTX_free(long ctx);
}

public class AEAD {
    static native long EVP_aead_aes_128_gcm();
    static native long EVP_aead_aes_256_gcm();
    static native long EVP_aead_chacha20_poly1305();
    static native long EVP_aead_aes_128_gcm_siv();
    static native long EVP_aead_aes_256_gcm_siv();
    static native int EVP_AEAD_max_overhead(long evpAead);
    static native int EVP_AEAD_nonce_length(long evpAead);
    static native int EVP_AEAD_CTX_seal(long evpAead, byte[] key, int tagLengthInBytes, byte[] out,
    int outOffset, byte[] nonce, byte[] in, int inOffset, int inLength, byte[] ad)
    throws ShortBufferException, BadPaddingException;
    static native int EVP_AEAD_CTX_seal_buf(long evpAead, byte[] key, int tagLengthInBytes, ByteBuffer out,
    byte[] nonce, ByteBuffer input, byte[] ad)
    throws ShortBufferException, BadPaddingException;
    static native int EVP_AEAD_CTX_open(long evpAead, byte[] key, int tagLengthInBytes, byte[] out,
    int outOffset, byte[] nonce, byte[] in, int inOffset, int inLength, byte[] ad)
    throws ShortBufferException, BadPaddingException;
    static native int EVP_AEAD_CTX_open_buf(long evpAead, byte[] key, int tagLengthInBytes, ByteBuffer out,
    byte[] nonce, ByteBuffer input, byte[] ad)
    throws ShortBufferException, BadPaddingException;
}

public class X509_REVOKED{
    static native long[] X509_CRL_get_REVOKED(long x509CrCtx, OpenSSLX509CRL holder);
    static native String[] get_X509_CRL_ext_oids(long x509CrCtx, OpenSSLX509CRL holder, int critical);
    static native byte[] X509_CRL_get_ext_oid(long x509CrCtx, OpenSSLX509CRL holder, String oid);
    static native void X509_delete_ext(long x509, OpenSSLX509Certificate holder, String oid);
    static native long X509_CRL_get_version(long x509CrCtx, OpenSSLX509CRL holder);
    static native long X509_CRL_get_ext(long x509CrCtx, OpenSSLX509CRL holder, String oid);
    static native byte[] get_X509_CRL_signature(long x509ctx, OpenSSLX509CRL holder);
    static native void X509_CRL_verify(long x509CrCtx, OpenSSLX509CRL holder,
    NativeRef.EVP_PKEY pkeyCtx) throws BadPaddingException, SignatureException,
    NoSuchAlgorithmException, InvalidKeyException, IllegalBlockSizeException;
    static native byte[] get_X509_CRL_crl_enc(long x509CrCtx, OpenSSLX509CRL holder);
    static native long X509_CRL_get_lastUpdate(long x509CrCtx, OpenSSLX509CRL holder);
    static native long X509_CRL_get_nextUpdate(long x509CrCtx, OpenSSLX509CRL holder);
}

public class HMAC{
    static native long HMAC_CTX_new();
    static native void HMAC_CTX_free(long ctx);
}
```



```
static native void HMAC_Init_ex(NativeRef.HMAC_CTX ctx, byte[] key, long evp_md);
static native void HMAC_Update(NativeRef.HMAC_CTX ctx, byte[] in, int inOffset, int inLength);
static native void HMAC_UpdateDirect(NativeRef.HMAC_CTX ctx, long inPtr, int inLength);
static native byte[] HMAC_Final(NativeRef.HMAC_CTX ctx);
}
```

- ☐ Yes (Refactor with this solution)
 ☐ Yes (Refactor with an alternative solution)
 ☐ No (No refactoring)

*** 44. Task:**

a) In your opinion, does the following code(s) contain any occurrence of design smell(implementation and-or design problem)?

```
static native long SSL_set_mode(long ssl, NativeSsl ssl_holder, long mode);

static void setEnabledCipherSuites(long ssl, NativeSsl ssl_holder, String[] cipherSuites, String[] protocols) {
  checkEnabledCipherSuites(cipherSuites);
  String maxProtocol = getProtocolRange(protocols).max;
  List opensslSuites = new ArrayList();
  for (int i = 0; i < cipherSuites.length; i++) {
    String cipherSuite = cipherSuites[i];
    if (cipherSuite.equals(TLS_EMPTY_RENEGOTIATION_INFO_SCSV)) {
      continue;
    }
    if (cipherSuite.equals(TLS_FALLBACK_SCSV)
        && (maxProtocol.equals(SUPPORTED_PROTOCOL_TLSV1)
            || maxProtocol.equals(SUPPORTED_PROTOCOL_TLSV1_1))) {
      SSL_set_mode(ssl, ssl_holder, NativeConstants.SSL_MODE_SEND_FALLBACK_SCSV);
      continue;
    }
    opensslSuites.add(cipherSuiteFromJava(cipherSuite));
  }
  SSL_set_cipher_lists(ssl, ssl_holder, opensslSuites.toArray(new String[opensslSuites.size()]));
}
```

- ☐ Yes
 ☐ No

45. b) If YES, please provide an explanation or specify the design smell(s) involved?

46. c) If YES, (In your opinion,) What is the motivation behind using this specific way of implementation?

*** 47. d) Please rate the severity of the implementation problem (if any), from 1 (Very Low) to 5 (Very High)**

1
Very Low

2
Low

3
Medium

4
High

5
Very High

N/A

* 48. e) If YES, would you apply this refactored solution?

```
static void setEnabledCipherSuites(long ssl, NativeSsl ssl_holder, String[] cipherSuites,String[] protocols) {
    checkEnabledCipherSuites(cipherSuites);
    String maxProtocol = getProtocolRange(protocols).max;
    List opensslSuites = new ArrayList();
    for (int i = 0; i < cipherSuites.length; i++) {
        String cipherSuite = cipherSuites[i];
        if (cipherSuite.equals(TLS_EMPTY_RENEGOTIATION_INFO_SCSV)) {
            continue;
        }
        if (cipherSuite.equals(TLS_FALLBACK_SCSV)
            && (maxProtocol.equals(SUPPORTED_PROTOCOL_TLSV1)
            || maxProtocol.equals(SUPPORTED_PROTOCOL_TLSV1_1))) {
            SSL_set_mode(ssl, ssl_holder, NativeConstants.SSL_MODE_SEND_FALLBACK_SCSV);
            continue;
        }
        opensslSuites.add(cipherSuiteFromJava(cipherSuite));
    }
    SSL_set_cipher_lists(ssl, ssl_holder, opensslSuites.toArray(new String[opensslSuites.size()]));
}
static native long SSL_set_mode(long ssl, NativeSsl ssl_holder, long mode);
```

☐ Yes (Refactor with this solution)

☐ Yes (Refactor with an alternative solution)

☐ No (No refactoring)

* 49. Task:

a) In your opinion, does the following code(s) contain any occurrence of design smell(implementation and-or design problem)?

```
static native long SSL_set_timeout(long ssl, NativeSsl ssl_holder, long millis);
static native void SSL_SESSION_free(long sslSessionNativePointer);

void doFree(long context) {
    NativeCrypto.SSL_SESSION_free(context);
}

static void NativeCrypto_SSL_SESSION_free(JNIEnv* env, jclass, jlong ssl_session_address) {
    CHECK_ERROR_QUEUE_ON_RETURN;
    SSL_SESSION* ssl_session = to_SSL_SESSION(env, ssl_session_address, true);
    JNI_TRACE("ssl_session=%p NativeCrypto_SSL_SESSION_free", ssl_session);
    if (ssl_session == nullptr) {
        return;
    }
    SSL_SESSION_free(ssl_session);
}
```

☐ Yes

☐ No

50. b) If YES, please provide an explanation or specify the design smell(s) involved?

51. c) If YES, (In your opinion,) What is the motivation behind using this specific way of implementation?

* 52. d) Please rate the severity of the implementation problem (if any), from 1 (Very Low) to 5 (Very High)

1 Very Low	2 Low	3 Medium	4 High	5 Very High	N/A
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

* 53. e) If YES, would you apply this refactored solution?

```
static native void SSL_SESSION_free(long sslSessionNativePointer);

void doFree(long context) {
    NativeCrypto.SSL_SESSION_free(context);
}

static void NativeCrypto_SSL_SESSION_free(JNIEnv* env, jclass, jlong ssl_session_address) {
    CHECK_ERROR_QUEUE_ON_RETURN;
    SSL_SESSION* ssl_session = to_SSL_SESSION(env, ssl_session_address, true);
    JNI_TRACE("ssl_session=%p NativeCrypto_SSL_SESSION_free", ssl_session);
    if (ssl_session == nullptr) {
        return;
    }
    SSL_SESSION_free(ssl_session);
}
```

- ☐ Yes (Refactor with this solution) ☐ Yes (Refactor with an alternative solution)
- ☐ No (No refactoring)

* 54. Task:

a) In your opinion, does the following code(s) contain any occurrence of design smell(implementation and-or design problem)?

```
static byte[] ecSignDigestWithPrivateKey(PrivateKey javaKey, byte[] message) {
    String keyAlgorithm = javaKey.getAlgorithm();
    if (!"EC".equals(keyAlgorithm)) {
        throw new RuntimeException("Unexpected key type: " + javaKey.toString());
    }
    return signDigestWithPrivateKey(javaKey, message, "NONEwithECDSA");
}
```

- ☐ Yes ☐ No

55. b) If YES, please provide an explanation or specify the design smell(s) involved?

56. c) If YES, (In your opinion,) What is the motivation behind using this specific way of implementation?

* 57. d) Please rate the severity of the implementation problem (if any), from 1 (Very Low) to 5 (Very High)

1
Very Low



2
Low



3
Medium



4
High



5
Very High



N/A



* 58. e) If YES, would you apply this refactored solution?

```
static byte[] ecSignDigestWithPrivateKey(PublicKey javaKey, byte[] message) {  
    String keyAlgorithm = javaKey.getAlgorithm();  
    if (keyAlgorithm != null){  
        if (!"EC".equals(keyAlgorithm)) {  
            throw new RuntimeException("Unexpected key type: " + javaKey.toString());  
        }  
        return signDigestWithPrivateKey(javaKey, message, "NONEwithECDSA");  
    }  
    return null;  
}
```

☐ Yes (Refactor with this solution)

☐ Yes (Refactor with an alternative solution)

☐ No (No refactoring)

Your responses have been registered!

Thank you for taking the time to complete the survey, your input is valuable to us.