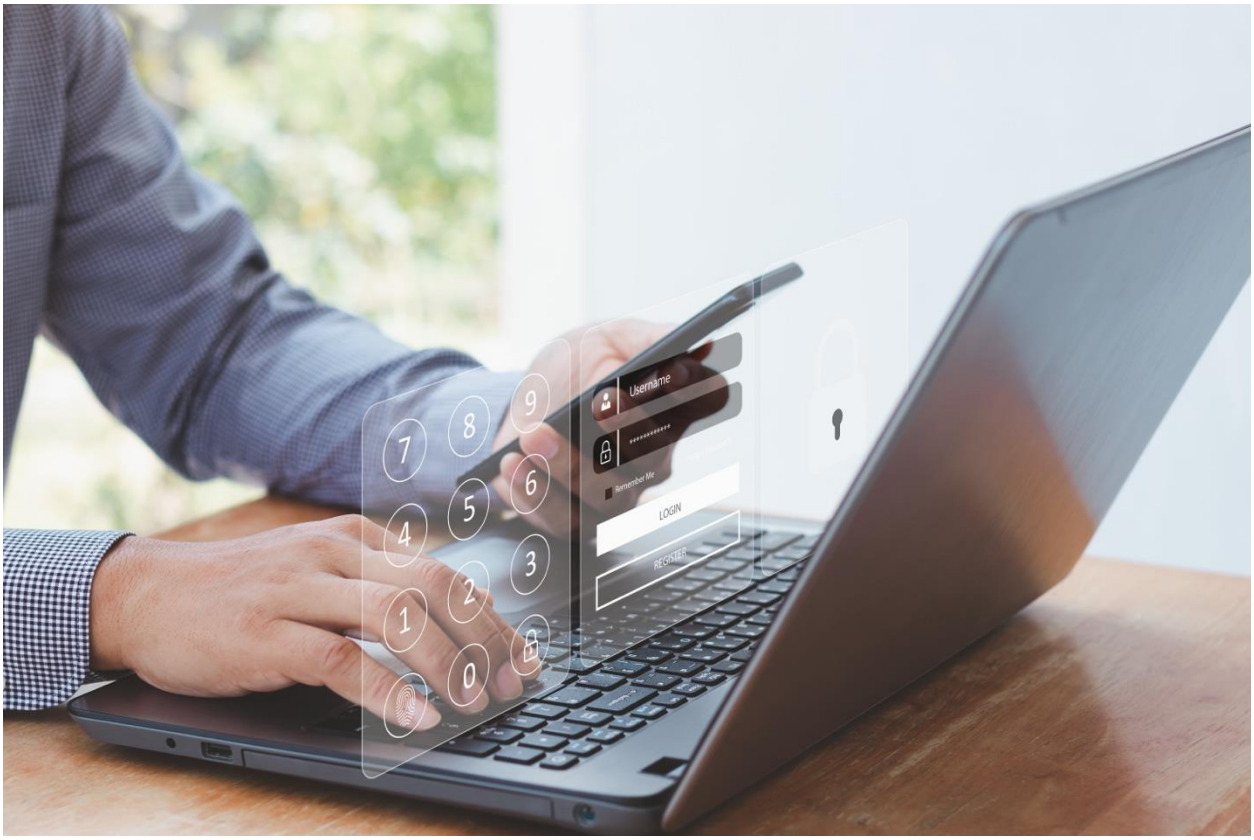
XOTIC Encryption

API Documentation

CHOL Inc
CHOL Irvine, California

Table of Contents

Section 1 – String Encryption



Overview

String encryption functions allow for the direct encipherment of text. XOTIC includes four [4] different API functions (modes of operation) for enciphering text. String encipherment is useful within the scope of applications, websites, text messages, and other general programming constructs where text-based data can be obfuscated.

XOTIC String encryption is the simplest form of encryption and is generally recommended as the starting place for engineers when familiarizing themselves with the XOTIC Cryptosystem.

Example:

```
String myText = “secure text message to send to my colleague”;

XoticString xs = XoticEncryption.xoticString();

Object[] result = xs.doEncryptString(myText, strength);  // int value <1-9>
```

XoticString API – Encryption

1.2.1 doEncryptString – Mode (1 of 4) – Basic Encryption API

Mode-1 (basic) "String encrypt" function allows direct programmatic encryption of text from API input parameters while using an XOTIC “system generated key” by strength setting.

```
/**
 * ENCRYPT A STRING – XOTIC String Encryption API
 * @param clearTextInput
 * @param strength
 * @return Object Array - [0]CipherText, [1]KeyString (Generated by Algorithm)
 * @throws Exception
*/
public Object[] doEncryptString(String clearTextInput, int strength) throws
                                                    Exception;
```

Input Parameters:

- The first input parameter “**clearTextInput**” is the "cleartext" (unencrypted String) that is meant to be encrypted.

- The second parameter “**strength**” is the numeric (integer) strength setting for the XOTIC encryption mechanism. Strength setting must be a number between <1-9>.

Output – Return Object:

XOTIC String encryption returns and **Object[]** Array as a primitive encapsulation type that contains several useful bits of information. Data returned by the basic String encryption operation are as follows:

- The first output parameter Object[0] is the “**CipherText** “(result) of the encryption job. Ciphertext is comprised of encrypted data of the type **byte[]**.

- The second output parameter Object[1] is the “**KeyString**” (autogenerated) by the encryption job. KeyString is comprised of the “.XKF” format XOTIC encryption key material and must be stored, saved, or shared in order to apply subsequent decryption (reversal back to cleartext) for the ciphertext produced in parameter[1].

Note: Any errors or runtime exceptions will generate a standard exception to be thrown.

1.2.2 doEncryptString – Mode (2 of 4) – Advanced Encryption – FIPS / HMAC MODE(S)

Mode-2 (advanced) "String encrypt" function incorporates AES-256 post processing "FIPS Mode" which is a (key-wrapping) feature for output of FIPS compliant encryption keys. "HMAC Mode" produces ciphertext with signed Encryption via AEAD/HMAC.

```
/**
 * ENCRYPT A STRING – XOTIC String Encryption API
 * @param clearTextInput
 * @param strength
 * @param fipsMode - Invokes AES 256 Key-Wrapping as a post-process
 * @param hmacSigned - Invokes Authenticated (AEAD) with HMAC Signed Header
 * @return Object Array - [0]CipherText, [1]KeyString (Generated by Algorithm)
 * @throws Exception
*/
public Object[] doEncryptString(String clearTextInput, int strength, boolean
                        fipsMode, boolean hmacSigned) throws Exception;
```

Input Parameters:

- The first input parameter "**clearTextInput**" is the "cleartext" (unencrypted String) that is meant to be encrypted.

- The second parameter "**strength**" is the numeric (integer) strength setting for the XOTIC encryption mechanism. Strength setting must be a number between <1-9>.

- The third parameter (**Boolean**) controls FIPS mode ON/OFF via true or false. When activated, FIPS mode invokes AES 256 Key-Wrapping as a post-process.

- The fourth parameter (**Boolean**) controls HMAC mode ON/OFF via true or false. When activated, HMAC mode incorporates Authenticated Encryption with Associated Data (AEAD) and HMAC Signed Header (signed encryption) during and throughout the encipherment process. Boolean flags may be used together in any combination of [TRUE / FALSE].

Output – Return Object:

XOTIC String encryption returns and **Object[]** Array as a primitive encapsulation type that contains several useful bits of information. Data types returned by the advanced String encryption in Mode-2 of operation identical to those returned by standard (basic) encryption.

Note: Any errors or runtime exceptions will generate a standard exception to be thrown.

1.2.3 doEncryptString – Mode (3 of 4) – User Defined Key Mode

Mode-3 (basic) "String encrypt" function allows users to supply their own key without utilizing the internal "system-generated" key mechanism controlled by dial-strength. In this way, users can encrypt with "key re-use" or external random-number generators if so desired.

```
/**
 * ENCRYPT A STRING - XOTIC String Encryption API
 * @param clearTextInput
 * @param keyString – pre-defined DSG
 * @return Object Array - [0]CipherText, [1]KeyString (Provided by Client)
 * @throws Exception
*/
public Object[] doEncryptString(String clearTextInput, String keyString) throws
                                                    Exception;
```

Input Parameters:

- The first input parameter "**clearTextInput**" is the "cleartext" (unencrypted String) that is meant to be encrypted.

- The second parameter "**keyString**" is a user-defined cryptographic key. In order to function properly, Key Strings must be compatible with XOTIC Key Format ".XKF".

  Note: XOTIC Key Strings "**.XKF**" may also be generated separately using the **XOTIC Key Tool** Feature.

Output – Return Object:

XOTIC String encryption returns and **Object[]** Array as a primitive encapsulation type that contains several useful bits of information. Data returned by the basic String encryption operation are as follows:

- The first output parameter Object[0] is the "**CipherText** "(result) of the encryption job. Ciphertext is comprised of encrypted data of the type **byte[]**.

- The second output parameter Object[1] is the "**KeyString**" (in the case of Mode-3) operation is an echo-back of the User Defined Key that was input as the second parameter. In this way, the return signature of this function would be indistinguishable from the return of the other functions that rely on system-generated keys.

Note: Any errors or runtime exceptions will generate a standard exception to be thrown.

1.2.4 doEncryptString – Mode (4 of 4) – User Defined Key - with (*optional*) HMAC MODE

Mode-4 (advanced) "String encrypt" function allows users to supply their own key without utilizing the internal "system-generated" key mechanism controlled by dial-strength. In this way, users can encrypt with "key re-use" or external random-number generators if so desired. This mode expands upon Mode-3 in that it also includes the optional AEAD/HMAC Signed encryption.

```
/**
 * ENCRYPT A STRING - XOTIC String Encryption API
 * @param clearTextInput
 * @param keyString – pre-defined DSG
 * @param hmacSigned - Invokes Authenticated (AEAD) with HMAC Signed Header
 * @return Object Array - [0]CipherText, [1]KeyString (Provided by Client)
 * @throws Exception
*/
public Object[] doEncryptString(String clearTextInput, String keyString,
                    boolean hmacSigned) throws Exception;) throws Exception;
```

Input Parameters:

- The first input parameter "**clearTextInput**" is the "cleartext" (unencrypted String) that is meant to be encrypted.

- The second parameter "**keyString**" is a user-defined cryptographic key. In order to function properly, Key Strings must be compatible with XOTIC Key Format ".XKF".

- The third parameter (**Boolean**) controls HMAC mode ON/OFF via true or false. When activated, HMAC mode incorporates Authenticated Encryption with Associated Data (AEAD) and HMAC Signed Header (signed encryption) during and throughout the encipherment process.

Output – Return Object:

XOTIC String encryption returns and **Object[]** Array as a primitive encapsulation type that contains several useful bits of information. Data returned by the basic String encryption operation are as follows:

- The first output parameter Object[0] is the "**CipherText** "(result) of the encryption job. Ciphertext is comprised of encrypted data of the type **byte[]**.

- The second output parameter Object[1] is the "**KeyString**" (in the case of Mode-3) operation is an echo-back of the User Defined Key that was input as the second parameter. In this way, the return signature of this function would be indistinguishable from the return of the other functions that rely on system-generated keys.

XoticString API – Decryption

1.3.1 doDecryptString – Mode (1 of 2) – String Decryption API

Mode-1 (basic) "String decrypt" function allows direct programmatic decryption of ciphertext (encrypted materials) based on binary and text-based parameters set by the API Client.

```
/**
 * DECRYPT A STRING – XOTIC String Decryption API
 * @param cipherText [binary]
 * @param xoticKeyString –
 * @return String - clearText
 * @throws Exception
*/
public String doDecryptString(byte[] cipherText, String xoticKeyString) throws
                                                    Exception;
```

Input Parameters:

- The first input parameter "**cipherText**" is the fully encrypted binary (byte array) object set into the API from the result of some other separate XOTIC encryption job.

- The second input parameter "**xoticKeyString**" is a user-defined cryptographic key. In order to function properly, Key Strings must be compatible with XOTIC Key Format ".XKF".

Note: XOTIC is only capable of decrypting materials enciphered by the XOTIC Cryptosystem.

Output – Return Object:

XOTIC String decryption returns a String object as the result of decipherment.

Note: Any errors or runtime exceptions will generate a standard exception to be thrown.

1.3.2 doDecryptString – Mode (2 of 2) – String Decryption API - with (*optional*) FIPS MODE

Mode-2 (advanced) "String decrypt" function allows direct programmatic decryption of ciphertext (materials) based on binary and text input parameters set by the API Client.

```
/**
 * DECRYPT A STRING – XOTIC String Decryption API
 * @param cipherText [binary]
 * @param xoticKeyString –
 * @param fipsMode - Invokes AES 256 Key UN-Wrapping as a pre-process
 * @return String - clearText
 * @throws Exception
*/
public String doDecryptString(byte[] cipherText, String xoticKeyString,
                    boolean fipsMode) throws Exception;) throws Exception;
```

Input Parameters:

- The first input parameter "**cipherText**" is the fully encrypted binary (byte array) object set into the API from the result of some other separate XOTIC encryption job.

- The second input parameter "**xoticKeyString**" is a user-defined cryptographic key. In order to function properly, Key Strings must be compatible with XOTIC Key Format ".XKF" or ".XKFF".

- The third parameter (**Boolean**) controls FIPS mode ON/OFF via true or false. When activated, FIPS mode invokes AES 256 Key UN-Wrapping as a pre-process.

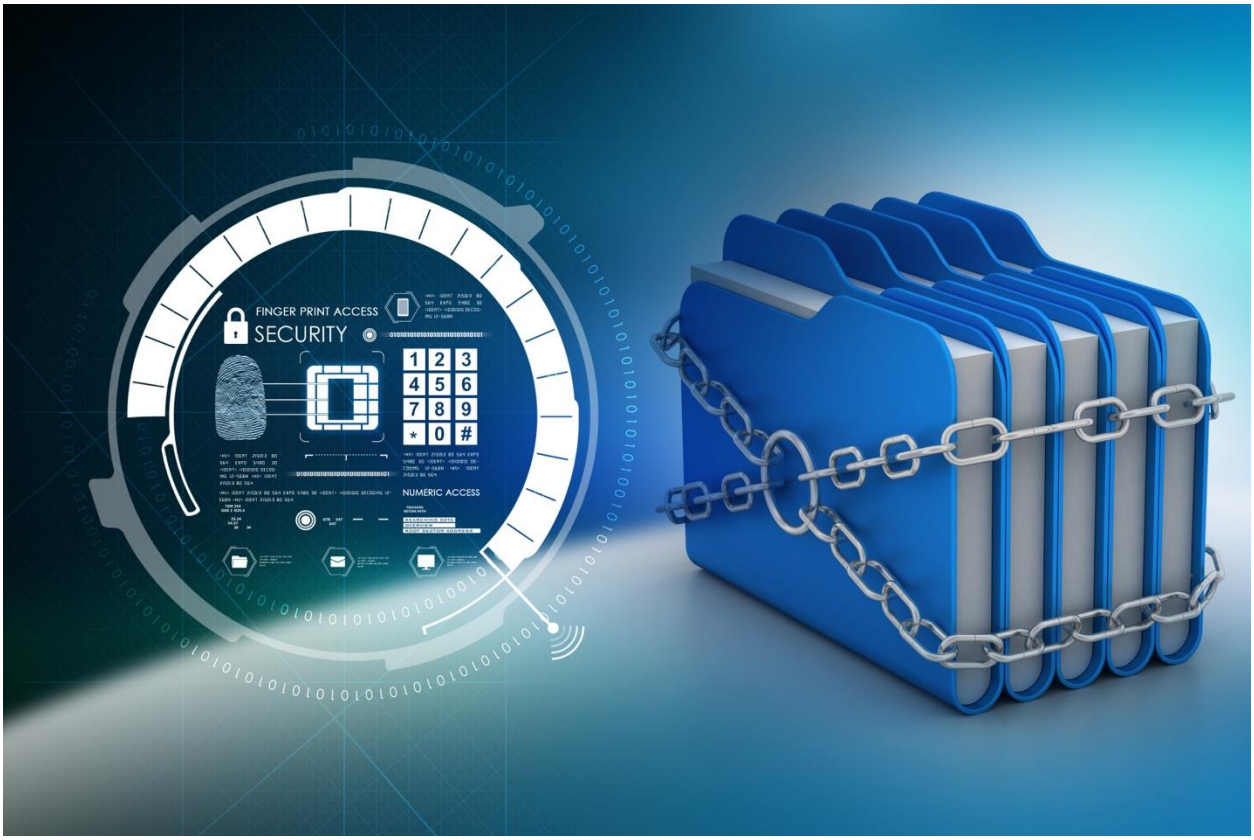
Note: When using FIPS Mode (TRUE), then the XOTIC FIPS Key Format ".XKFF" is expected in the xoticKeyString input. FIPS mode will not decrypt standard ".XKF" keys, and vice-versa.

Output – Return Object:

XOTIC String decryption returns a String object as the result of decipherment.

Note: Any errors or runtime exceptions will generate a standard exception to be thrown.

Section 2 – File Encryption



Overview

File encryption functions allow for the direct encipherment of binary files on a local file system. Supported file systems are any operating system capable of running a Java Runtime Environment (JRE), such as Windows, Mac, Linux, etc. JRE must be from version 1.6 (circa year 2006) or greater.

XOTIC includes four [4] different API functions (modes of operation) for enciphering files. File encipherment is useful within the scope of applications, backups, storage and management, and other general programming constructs where binary file-based data can be obfuscated. XOTIC File encryption leverages standard Java I/O libraries for direct local file access and is subject to any user-permission or security restrictions that may govern the Java Virtual Machine's underlying access to the file system. Remote file system / networked file system access is not included in this API.

Example:

```
String myFilePath = "c:\\temp\\myDog.jpeg";

XoticFile xf = XoticEncryption.xoticFile();

 xf.doEncryptFile(myFilePath, strength);  // int value <1-9>
```

XoticFile API – Encryption

2.2.1 doEncryptFile – Mode (1 of 4) – Basic Encryption API

Mode-1 (basic) "File encrypt" function allows direct programmatic encryption of local files from API input parameters while using an XOTIC “system generated key” by strength setting.

```
/**
 * ENCRYPT A FILE - XOTIC File Encryption API
 * @param absolutePathToClearTextFile
 * @param strength
 * @return long integer - time stamp milliseconds for encryption
 * @throws Exception
*/
public long doEncryptFile(String absolutePathToClearTextFile, int strength) throws
                                                    Exception;
```

Input Parameters:

- The first input parameter “**absolutePathToClearTextFile**” is the System file path which holds the files that need to be encrypted.

- The second parameter “**strength**” is the numeric (integer) strength setting for the XOTIC encryption mechanism. Strength setting must be a number between <1-9>.

Output – Return Object:

XOTIC File encryption returns time interval that each file took to complete the encryption.

Note: Any errors or runtime exceptions will generate a standard exception to be thrown.

2.2.2 doEncryptFile – Mode (2 of 4) – Advanced Encryption – FIPS / HMAC MODE(S)

Mode-2 (advanced) "File encrypt" function incorporates AES-256 post processing "FIPS Mode" which is a (key-wrapping) feature for output of FIPS compliant encryption keys. "HMAC Mode" produces ciphertext with signed Encryption via AEAD/HMAC.

```
/**
 * ENCRYPT A FILE - XOTIC File Encryption API
 * @param absolutePathToClearTextFile
 * @param strength
 * @param fipsMode - Invokes AES 256 Key-Wrapping as a post-process
 * @param hmacSigned - Invokes Authenticated (AEAD) with HMAC Signed Header
 * @return long integer - time stamp milliseconds for encryption
 * @throws Exception
*/
public long doEncryptFile(String absolutePathToClearTextFile, int strength , boolean
                          fipsMode, boolean hmacSigned) throws Exception;
```

Input Parameters:

- The first input parameter "**absolutePathToClearTextFile**" is the System file path which holds the files that need to be encrypted.
- The second parameter "**strength**" is the numeric (integer) strength setting for the XOTIC encryption mechanism. Strength setting must be a number between <1-9>.
- The third parameter "**fipsMode**" (**Boolean**) controls FIPS mode ON/OFF via true or false. When activated, FIPS mode invokes AES 256 Key-Wrapping as a post-process.
- The fourth parameter "**hmacSigned**" (**Boolean**) controls HMAC mode ON/OFF via true or false. When activated, HMAC mode incorporates Authenticated Encryption with Associated Data (AEAD) and HMAC Signed Header (signed encryption) during and throughout the encipherment process. Boolean flags may be used together in any combination of [TRUE / FALSE].

Output – Return Object:

XOTIC File encryption returns time interval that each file took to complete the encryption.

Note: Any errors or runtime exceptions will generate a standard exception to be thrown.

2.2.3 doEncryptFile – Mode (3 of 4) – User Defined Key Mode

Mode-3 (basic) "File encrypt" function allows users to supply their own key without utilizing the internal "system-generated" key mechanism controlled by dial-strength. In this way, users can encrypt with "key re-use" or external random-number generators if so desired.

```
/**
 * ENCRYPT A FILE - XOTIC File Encryption API
 * @param absolutePathToClearTextFile
 * @param keyString
 * @return long integer - time stamp milliseconds for encryption
 * @throws Exception
*/
public long doEncryptFile(String absolutePathToClearTextFile, String keyString) throws
                                                    Exception;
```

Input Parameters:

- The first input parameter "**absolutePathToClearTextFile**" is the System file path which holds the files that need to be encrypted.

- The second parameter "**keyString**" is a user-defined cryptographic key. In order to function properly, Key Strings must be compatible with XOTIC Key Format ".XKF".

  Note: XOTIC Key Strings "**.XKF**" may also be generated separately using the **XOTIC Key Tool** Feature.

Output – Return Object:

XOTIC File encryption returns time interval that each file took to complete the encryption.

Note: Any errors or runtime exceptions will generate a standard exception to be thrown.

2.2.4 doEncryptFile – Mode (4 of 4) – User Defined Key - with (*optional*) HMAC MODE

Mode-4 (advanced) "File encrypt" function allows users to supply their own key without utilizing the internal "system-generated" key mechanism controlled by dial-strength. In this way, users can encrypt with "key re-use" or external random-number generators if so desired. This mode expands upon Mode-3 in that it also includes the optional AEAD/HMAC Signed encryption.

```
/**
 * ENCRYPT A FILE - XOTIC File Encryption API
 * @param absolutePathToClearTextFile
 * @param keyString - pre-defined DSG
 * @param hmacSigned - Invokes Authenticated (AEAD) with HMAC Signed Header
 * @return long integer - time stamp milliseconds for encryption
 * @throws Exception
*/
public long doEncryptFile(String absolutePathToClearTextFile, String keyString,
                          boolean hmacSigned) throws Exception;
```

Input Parameters:

- The first input parameter "**absolutePathToClearTextFile**" is the System file path which holds the files that need to be encrypted.

- The second parameter "**keyString**" is a user-defined cryptographic key. In order to function properly, Key Strings must be compatible with XOTIC Key Format ".XKF".

  Note: XOTIC Key Strings "**.XKF**" may also be generated separately using the **XOTIC Key Tool** Feature.

- The third parameter "**hmacSigned**" (**Boolean**) controls HMAC mode ON/OFF via true or false. When activated, HMAC mode incorporates Authenticated Encryption with Associated Data (AEAD) and HMAC Signed Header (signed encryption) during and throughout the encipherment process.

Output – Return Object:

XOTIC File encryption returns time interval that each file took to complete the encryption.

Note: Any errors or runtime exceptions will generate a standard exception to be thrown.

XoticFile API – Decryption

2.3.1 doDecryptFile – Mode (1 of 1) – File Decryption API

Mode-1 (basic) "File decrypt" function allows direct programmatic decryption of ciphertext (Encrypted materials) based on binary and text-based parameters set by the API Client.

```
/**
 * DECRYPT A FILE - XOTIC File Decryption API
 * @param absolutePathToClearTextFile
 * @param absolutePathToKeyFile
 * @return long integer - time stamp milliseconds for encryption
 * @throws Exception
*/
public long doDecryptFile(String absolutePathToClearTextFile, String
                                absolutePathToKeyFile) throws Exception;
```

Input Parameters:

- The first input parameter "**absolutePathToEncryptedInputFile**" is the system path to the files that need to be encrypted using XOTIC.

- The second input parameter "**absolutePathToKeyFile**" is the path to the user-defined cryptographic key. To function properly, Key files must be compatible with XOTIC Key Format ".XKF".

Note: XOTIC is only capable of decrypting materials enciphered by the XOTIC Cryptosystem.

Output – Return Object:

XOTIC File decryption returns time interval that each file took to complete the decryption.

Note: Any errors or runtime exceptions will generate a standard exception to be thrown.

Section 3 – Media (Streaming) Encryption

Overview

Media encryption functions allow for the streaming (continuous or contiguous) encipherment of binary data streams in real-time. XOTIC supports concurrent and parallelized encryption and decryption streams such that multiple encryption "jobs" can remain "open" across multiple "segments" (packets) of binary data.

XOTIC includes four [5] different API functions (modes of operation) for enciphering binary data / digital media. Media encipherment is useful within the scope of streaming video, database encryption, live feeds, and other general or generic forms of binary data.

Example:

```
byte[] myDataPacket =  InputStreamReader.next();      // pull in a packet of binary data

XoticMedia xm = XoticEncryption.xoticMedia();

Object[] result = xs.doEncryptDataWithInit (myDataPacket, strength, true);  // true – terminate job
```

3.2.1 doEncryptDataWithInit – Mode (1 of 5) – Streaming Encryption API - Initializing

Mode-1 (basic) "Media encrypt" function allows direct programmatic encryption of stream-based media file from API input parameters while using an XOTIC “system generated key”. This function will start a new job for each media file, if there is a job running, then it utilizes that job itself. We can conditionally terminate a job as well. It returns ciphertext, key and jobID(conditionally).

```
/**
 * ENCRYPT BINARY MEDIA - XOTIC Media Encryption API
 * @param clearText
 * @param strength
 * @param terminateJob
 * @return Object[] [cipherText, job number, DSG Cycle]
 * @throws Exception
*/
public long doEncryptDataWithInit(String clearText, int strength, boolean
                                  terminateJob) throws Exception;
```

Input Parameters:

- The first input parameter “**clearText**” is the "cleartext" (unencrypted String) that is meant to be encrypted.
- The second parameter “**strength**” is the numeric (integer) strength setting for the XOTIC encryption mechanism. Strength setting must be a number between <1-9>.
- The third parameter “**terminateJob**” (**Boolean**) controls the job activity ON/OFF via true or false. When activated, the job that is running will be terminated.

Output – Return Object:

XOTIC Media encryption returns and **Object[]** Array as a primitive encapsulation type that contains several useful bits of information. Data returned by the basic Media encryption operation are as follows:

- The first output parameter Object[0] is the “**CipherText** “(result) of the encryption job. Ciphertext is comprised of encrypted data of the type **byte[]**.
- The second output parameter Object[1] is the “**KeyString**” (in the case of Mode-3) operation is an echo-back of the User Defined Key that was input as the second parameter. In this way,

the return signature of this function would be indistinguishable from the return of the other functions that rely on system-generated keys.

- The third output parameter Object[2] is "**encryptionJobNumber**" is the numeric (integer) job number for identifying each job that are used to encrypt the media file.

Note: Any errors or runtime exceptions will generate a standard exception to be thrown.

3.2.2 doEncryptDataWithInit – Mode (2 of 5) – Advanced Encryption – FIPS / HMAC MODE(S)

Mode-2 (advanced) "Media encrypt" function allows direct programmatic encryption of stream-based media file from API input parameters while using an XOTIC “system generated key based on binary and text-based parameters set by the API Client.

```
/**
 * ENCRYPT BINARY MEDIA - XOTIC Media Encryption API
 * @param clearText
 * @param strength
 * @param terminateJob
 * @param fipsMode - Invokes AES 256 Key-Wrapping as a post-process
 * @param hmacSigned - Invokes Authenticated (AEAD) with HMAC Signed Header
 * @return Object[] [cipherText, job number, DSG Cycle]
 * @throws Exception
*/
public long doEncryptDataWithInit(String clearText, int strength, boolean
                        terminateJob, boolean fipsMode, boolean hmacSigned) throws Exception;
```

Input Parameters:

- The first input parameter “**clearText**” is the "cleartext" (unencrypted String) that is meant to be encrypted.

- The second parameter “**strength**” is the numeric (integer) strength setting for the XOTIC encryption mechanism. Strength setting must be a number between <1-9>.

- The third parameter “**terminateJob**” (**Boolean**) controls the job activity ON/OFF via true or false. When activated, the job that is running will be terminated.

- The fourth parameter (**Boolean**) controls FIPS mode ON/OFF via true or false. When activated, FIPS mode invokes AES 256 Key-Wrapping as a post-process.

- The fifth parameter (**Boolean**) controls HMAC mode ON/OFF via true or false. When activated, HMAC mode incorporates Authenticated Encryption with Associated Data (AEAD) and HMAC Signed Header (signed encryption) during and throughout the encipherment process. Boolean flags may be used together in any combination of [TRUE / FALSE].

Output – Return Object:

XOTIC Media encryption returns and **Object[]** Array as a primitive encapsulation type that contains several useful bits of information. Data returned by the basic Media encryption operation are as follows:

- The first output parameter Object[0] is the “**CipherText** “(result) of the encryption job. Ciphertext is comprised of encrypted data of the type **byte[]**.

- The second output parameter Object[1] is the “**KeyString**” (in the case of Mode-3) operation is an echo-back of the User Defined Key that was input as the second parameter. In this way, the return signature of this function would be indistinguishable from the return of the other functions that rely on system-generated keys.

- The third output parameter Object[2] is “**encryptionJobNumber**” is the numeric (integer) job number for identifying each job that are used to encrypt the media file.

Note: Any errors or runtime exceptions will generate a standard exception to be thrown.

3.2.3 doEncryptData – Mode (3 of 5) – Streaming Encryption API- Continuing

Mode-3 (basic) "Media encrypt" function allows direct programmatic encryption of stream-based media file from API input parameters while using an XOTIC “encryption job number”. This

function will continue a job for each media file. We can conditionally terminate a job as well. It returns ciphertext.

```
/**
 * ENCRYPT BINARY MEDIA - XOTIC Media Encryption API
 * @param clearText
 * @param encryptionJobNumber
 * @param terminateJob
 * @param hmacSigned - Invokes Authenticated (AEAD) with HMAC Signed Header
 * @return byte[] ciphertext
 * @throws Exception
*/
public long doEncryptData(String clearText, long encryptionJobNumber, boolean
                          terminateJob, boolean   hmacSigned) throws Exception;
```

Input Parameters:

- The first input parameter "**clearText**" is the "cleartext" (unencrypted String) that is meant to be encrypted.
- The second parameter "**encryptionJobNumber**" is the numeric (long) job number for identifying each job that are used to encrypt the media file.
- The third parameter "**terminateJob**" (**Boolean**) controls the job activity ON/OFF via true or false. When activated, the job that is running will be terminated.

Output – Return Object:

XOTIC String encryption returns an array object called "**CipherText** "(result) of the encryption job. Ciphertext is comprised of encrypted data of the type **byte**[].

Note: Any errors or runtime exceptions will generate a standard exception to be thrown.

3.2.4 doEncryptData – Mode (4 of 5) – Advanced Encryption – HMAC MODE(S)

Mode-4 (basic) "Media encrypt" function allows direct programmatic encryption of stream-based media file from API input parameters while using an XOTIC "encryption job number". This function will continue a job for each media file. We can conditionally terminate a job as well. It returns ciphertext. "HMAC Mode" produces ciphertext with signed Encryption via AEAD/HMAC.

```
/**
 * ENCRYPT BINARY MEDIA - XOTIC Media Encryption API
 * @param clearText
 * @param encryptionJobNumber
 * @param terminateJob
 * @return byte[] ciphertext
 * @throws Exception
*/
public long doEncryptData(String clearText, long encryptionJobNumber, boolean
                                        terminateJob) throws Exception;
```

Input Parameters:

- The first input parameter "**clearText**" is the "cleartext" (unencrypted String) that is meant to be encrypted.
- The second parameter "**encryptionJobNumber**" is the numeric (integer) job number for identifying each jobs that are used to encrypt the media file.
- The third parameter "**terminateJob**" (**Boolean**) controls the job activity ON/OFF via true or false. When activated, the job that is running will be terminated.
- The fourth parameter "**hmacSigned**" (**Boolean**) - Invokes Authenticated Encryption with Associated Data (AEAD) and HMAC Signed Header.

Output – Return Object:

XOTIC String encryption returns an array object called "**CipherText** "(result) of the encryption job. Ciphertext is comprised of encrypted data of the type **byte**[].

Note: Any errors or runtime exceptions will generate a standard exception to be thrown.

3.2.5 doEncryptData – Mode (5 of 5) – User Defined Key - with (*optional*) HMAC MODE

Mode-5 (basic) "Media encrypt" function allows direct programmatic encryption of stream-based media file from API input parameters while allows users to supply their own key without utilizing the internal "system-generated" key mechanism controlled by dial-strength. This function will continue a job for each media file. We can conditionally terminate a job as well. It returns ciphertext. "HMAC Mode" produces ciphertext with signed Encryption via AEAD/HMAC.

```
/**
```

```
 * ENCRYPT BINARY MEDIA - XOTIC Media Encryption API
 * @param clearText
 * @param xoticKeyString
 * @param terminateJob
 * @return byte[] ciphertext
 * @throws Exception
*/
public long doEncryptData(String clearText, String xoticKeyString, boolean
                                        terminateJob) throws Exception;
```

Input Parameters:

- The first input parameter "**clearText**" is the "cleartext" (unencrypted String) that is meant to be encrypted.
- The second input parameter "**xoticKeyString**" is a user-defined cryptographic key. In order to function properly, Key Strings must be compatible with XOTIC Key Format ".XKF".
- The third parameter "**terminateJob**" (**Boolean**) controls the job activity ON/OFF via true or false. When activated, the job that is running will be terminated.
- The fourth parameter "**hmacSigned**" (**Boolean**) - Invokes Authenticated Encryption with Associated Data (AEAD) and HMAC Signed Header.

Output – Return Object:

XOTIC String encryption returns an array object called "**CipherText** "(result) of the encryption job. Ciphertext is comprised of encrypted data of the type **byte**[].

Note: Any errors or runtime exceptions will generate a standard exception to be thrown.

XoticMedia API – Decryption

2.3.1 doDecryptDataWithInit – Mode (1 of 4) – Media Decryption API

Mode-1 (basic) "Media decrypt" function allows direct programmatic decryption of ciphertext (encrypted materials) based on binary and text-based parameters while using an XOTIC "decryption job number". This function will continue a job for each media file. We can conditionally terminate a job as well. It returns ciphertext(binary packet).

```
/**
 * DECRYPT BINARY MEDIA - XOTIC Media Decryption API
```

```
 * @param cipherText
 * @param decryptionJobNumber
 * @param terminateJob
 * @return Object[] [clearText, job number, DSG Cycle]
 * @throws Exception
*/
public long doDecryptDataWithInit(String cipherText, long decryptionJobNumber, boolean
                        terminateJob) throws Exception;
```

Input Parameters:

- The first input parameter "**cipherText**" is the "cleartext" (unencrypted String) that is meant to be decrypted.

- The second parameter "**decryptionJobNumber**" is the numeric (integer) job number for identifying each jobs that are used to decrypt the media file.

- The third parameter "**terminateJob**" (**Boolean**) controls the job activity ON/OFF via true or false. When activated, the job that is running will be terminated.

Output – Return Object:

XOTIC Media decryption returns and **Object[]** Array as a primitive encapsulation type that contains several useful bits of information. Data returned by the basic Media decryption operation are as follows:

- The first output parameter Object[0] is the "**clearText** "(result) of the decryption job.

- The second output parameter Object[1] is the "**KeyString**" (in the case of Mode-3) operation is an echo-back of the User Defined Key that was input as the second parameter. In this way, the return signature of this function would be indistinguishable from the return of the other functions that rely on system-generated keys.

- The third output parameter Object[2] is "**encryptionJobNumber**" is the numeric (integer) job number for identifying each job that are used to encrypt the media file.

Note: Any errors or runtime exceptions will generate a standard exception to be thrown.

2.3.2 doDecryptDataWithInit – Mode (2 of 4) –Media Decryption API - with (*optional*) FIPS MODE

Mode-2 (basic) "Media decrypt" function allows direct programmatic decryption of ciphertext (encrypted materials) based on binary and text-based parameters while using an XOTIC “decryption job number”. This function will continue a job for each media file. We can conditionally terminate a job as well. It returns ciphertext(binary packet). This function incorporates AES-256 post processing “FIPS Mode” which is a (key-wrapping) feature for output of FIPS compliant encryption keys.

```
/**
 * DECRYPT BINARY MEDIA - XOTIC Media Decryption API
 * @param cipherText
```

```
 * @param decryptionJobNumber
 * @param terminateJob
 * @param fips
 * @return Object[] [clearText, job number, DSG Cycle]
 * @throws Exception
*/
public long doDecryptDataWithInit(String cipherText, long decryptionJobNumber, boolean
                        terminateJob, boolean  fips) throws Exception;
```

Input Parameters:

- The first input parameter "**clearText**" is the "cleartext" (unencrypted String) that is meant to be decrypted.
- The second parameter "**encryptionJobNumber**" is the numeric (integer) job number for identifying each jobs that are used to decrypt the media file.
- The third parameter "**terminateJob**" (**Boolean**) controls the job activity ON/OFF via true or false. When activated, the job that is running will be terminated.
- The third parameter (**Boolean**) controls FIPS mode ON/OFF via true or false. When activated, FIPS mode invokes AES 256 Key-Wrapping as a post-process.

Output – Return Object:

XOTIC Media decryption returns and **Object[]** Array as a primitive encapsulation type that contains several useful bits of information. Data returned by the basic Media decryption operation are as follows:

- The first output parameter Object[0] is the "**clearText** "(result) of the decryption job.
- The second output parameter Object[1] is the "**KeyString**" (in the case of Mode-3) operation is an echo-back of the User Defined Key that was input as the second parameter. In this way, the return signature of this function would be indistinguishable from the return of the other functions that rely on system-generated keys.
- The third output parameter Object[2] is "**encryptionJobNumber**" is the numeric (integer) job number for identifying each job that are used to encrypt the media file.

Note: Any errors or runtime exceptions will generate a standard exception to be thrown.

2.3.3 doDecryptData – Mode (3 of 4) – Media Decryption API – Continuing

Mode-3 (basic) "Media decrypt" function allows direct programmatic decryption of ciphertext (encrypted materials) based on binary and text-based parameters set by the API Client. It returns ciphertext(binary packet).

```
/**
 * DECRYPT BINARY MEDIA - XOTIC Media Decryption API
 * @param cipherText
 * @param keyString
 * @return byte[] clearText
```

```
 * @throws Exception
*/
public long doDecryptData(String cipherText, long keyString) throws Exception;
```

Input Parameters:

- The first input parameter "**cipherText**" is the "cleartext" (unencrypted String) that is meant to be decrypted.

- The second parameter "**keyString**" is a user-defined cryptographic key. In order to function properly, Key Strings must be compatible with XOTIC Key Format ".XKF".

  Note: XOTIC Key Strings "**.XKF**" may also be generated separately using the **XOTIC Key Tool** Feature.

Output – Return Object:

XOTIC Media decryption returns an array object called "**ClearText** "(result) of the decryption job.

Note: Any errors or runtime exceptions will generate a standard exception to be thrown.

2.3.4 doDecryptData – Mode (4 of 4) – Media Decryption API - User Defined Key

Mode-4 (basic) "Media decrypt" function allows direct programmatic decryption of ciphertext (encrypted materials) based on binary and text-based parameters set by the API Client. This function will continue a job for each media file. We can conditionally terminate a job as well. It returns ciphertext(binary packet).

```
/**
 * DECRYPT BINARY MEDIA - XOTIC Media Decryption API
 * @param cipherText
 * @param keyString
 * @param terminateJob
 * @return byte[] clearText
```

```
 * @throws Exception
*/
public long doDecryptData(String cipherText, long keyString, boolean
                                    terminateJob) throws Exception;
```

Input Parameters:

- The first input parameter "**clearText**" is the "cleartext" (unencrypted String) that is meant to be encrypted.
- The second parameter "**decryptionJobNumber**" is the numeric (integer) job number for identifying each jobs that are used to encrypt the media file.
- The third parameter "**terminateJob**" (**Boolean**) controls the job activity ON/OFF via true or false. When activated, the job that is running will be terminated.

Output – Return Object:

XOTIC Media decryption returns an array object called "**ClearText** "(result) of the decryption job.

Note: Any errors or runtime exceptions will generate a standard exception to be thrown.

Section 4 – XOTIC Key Tool

Overview

XOTIC Key Tool is an additional API that allows for the generation of XOTIC compatible encryption keys that can be used with subsequent user-defined-key modes of encryption. User generated keys leverage a combination of User supplied SALT (password) and XOTIC CSRNG Strength settings.

Example:

```
String converted = convert(password);

XoticJobVO xoticJobVO= vectorize(strength, converted);

Object[] rval = KeyFileGenerator.generateXoticKeyString(xoticJobVO);
```

XOTIC API – Key Generator

4.2.1 genKey – USER GENERATED XKF STRING

XOTIC Key Tool - "genKey" function allows for the generation of XOTIC compatible encryption keys that can be used with subsequent user-defined-key modes of encryption.

```
/**
 * User selected password and dial strength render XKF
 * @param strength
 * @param password
 * @return String
 * @throws Exception
*/
public long genKey(String strength, long password) throws Exception;
```

Input Parameters:

- The first parameter "**strength**" is the numeric (integer) strength setting for the XOTIC encryption mechanism. Strength setting must be a number between <1-9>.

- The second parameter "**password**" (**String**) is text which represents the password for creating XOTIC key.

Output – Return Object:

The return object is the "**xoticKeyString**" is a user-defined cryptographic key generated

Note: Any errors or runtime exceptions will generate a standard exception to be thrown.