CypherShield Encryption SDK Quick Start Guide

Table of Contents

Introduction

The CypherShield SDK and companion key certification server comprise a complete post-quantum security model. This system can be used to provide post-quantum point to point encryption for data-in-motion. The system comes with digital signature capability as well as end point authentication.

CypherShield provides a unique encryption envelope that allows developers to augment their current cybersecurity infrastructure. The performance characteristics of the system provides post-quantum security with minimal performance overhead and is compatible with current network transport security models. Furthermore, the low overhead and minimal performance impact make the CypherShield SDK ideal for use in low-powered and Bluetooth beacon solutions.

CypherShield encryption SDK is a client-side component that will be installed or embedded in the client program or software as an add-on library.

A quick summary of what constitutes this SDK:

- Post-Quantum Point-to-Point Network Encryption (data in motion)
- Endpoint certification to prevent man-in-the-middle vulnerabilities
- Envelope encryption technology using Key Encapsulation Techniques (KEM)

Intended Audience

The CypherShield post-quantum solution is ideal for companies and solution providers looking to provide a post quantum safe cybersecurity solution. The system can be deployed alongside existing security solutions thus providing the needed futureproofing for most networks.

Target Customers and Infrastructures:

- Products where Customer own both ends of the network solution
- Product integrations
  - Embedded into customer products such as messaging systems, VPN, network equipment etc.

Licensing

CypherShield comes with an evaluation license. Developers will need to contact CypherEye at sales@cyphereye.com to request an evaluation or production license.

Below is the contact form link to raise a download request.

https://cyphershield.com/software-download/

Evaluation licenses are provided and expire after a set period.

Purchased Production licenses are permanent and never expire.

Getting Started

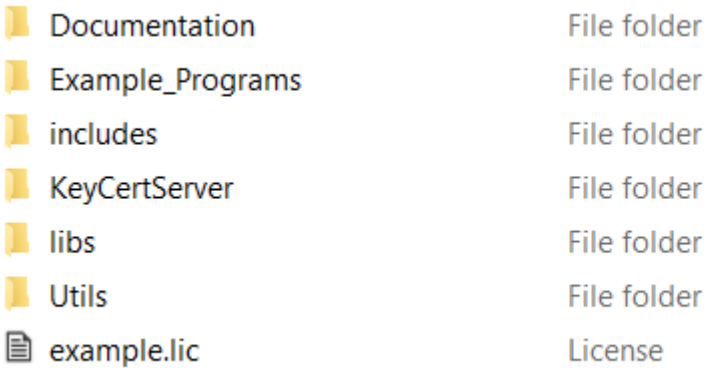
Request a download of the latest version of the CypherShield Encryption SDK package at:

https://cyphershield.com/software-download/

1. The SDK is distributed as a single ZIP archive file.
2. Install the SDK into your development system.

The package will include the following folders:

- libs – C++ software libraries to be linked into the customer's project
- includes – Public SDK header files
- Examples_Programs – Example programs for each supported platform
- Documentation – SDK technical documentation
- KeyCertServer – The CypherShield Key Certification Server deployment system
- Utils – Utility programs provided to support customers for deployment and management
- example.lic - evaluation license file



Supported OS platforms:

- Linux
- Windows
- iOS
- Android

Supported languages:

- Linux, Windows: C++, Node.js
- iOS: Swift, ObjectiveC
- Android: Java

Build and Deploy Instructions

The CypherShield Encryption SDK includes the software components and utilities to embed an advanced security and encryption technology into customer products and infrastructure solutions.

The following steps will be required for organizations to create a security infrastructure to create and support a post-quantum network product solution.

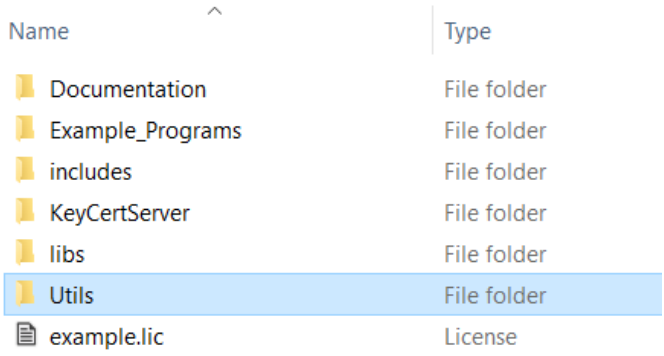
Completing the following tasks are required to enable this.

1. Generating Keys and Certificates
2. Linux Native Solutions
3. Apple iOS
4. Android
5. Node.js (Linux/Windows)

Generating Keys and Certificates

The CypherShield encryption software provides the capability for customers to generate unique encryption and network keys and certificates. These security keys are used by the customer to secure network encryption and communications.

To generate customer public and private keys, a command line utility is provided. This is located inside the Utils folder in the SDK distribution package.



Copy the contents of the Util folder to a Linux Ubuntu or Redhat/CentOS system. The utility for generating customer keys is **genkeypair**.

Create the master certificates

Execute the command

```
./genkeypair /mc
```

This command will generate the following key files

masterpubkey.key

masterpvtkey.key

These certificate files will be required by the SDK client software when building the encryption systems on each various platform. Refer to the examples for each platform.

Create the Server Certificates

Execute the command

```
./genkeypair /sc
```

This command will generate the following key files.

mastersign.key

serverpubkey.key

serverpvtkey.key

The certificate files generated will be required during the installation of the Key Certificate Server. They will be required to be copied and installed on this server.

Note: Refer to the Key Certification Server administration guide included with the SDK distribution package, or available on the CypherShield website at https://cyphershield.com/product-guides

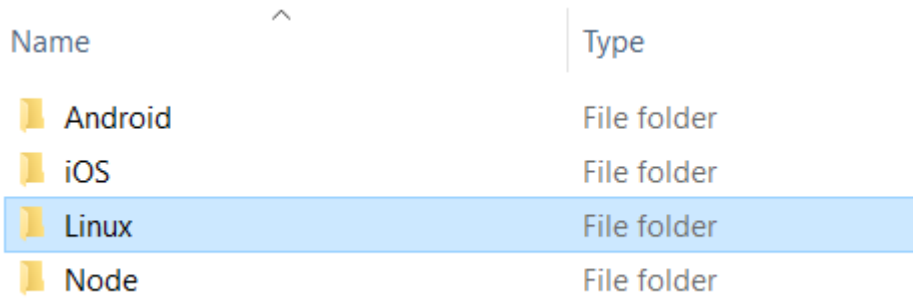
Native Linux Solutions

The CypherShield technologies presented as Native Linux are implementations without platform wrappers required by environments like iOS, Android, Node.js, etc.

The CypherShield Encryption technology can be utilized to provide end-to-end security for a large variety of products and solutions. These solutions include:

- Network Appliances such as routers, switches, and various network provider solutions
- VPN technologies
- Messaging and voice/video
- Others

The SDK Example Programs folder contains examples to implement the CypherShield technology. These examples can be used for a variety of infrastructures including software and hardware implementations.



The Linux native examples are C++ programs implementing CypherShield Encryption directly and is appropriate for numerous environments and solutions.

There is a build script included to build the Linux Native solution inside the Linux folder.

Execute this script ./build.sh

This will build the prerequisite libraries, then build the example.cpp program.

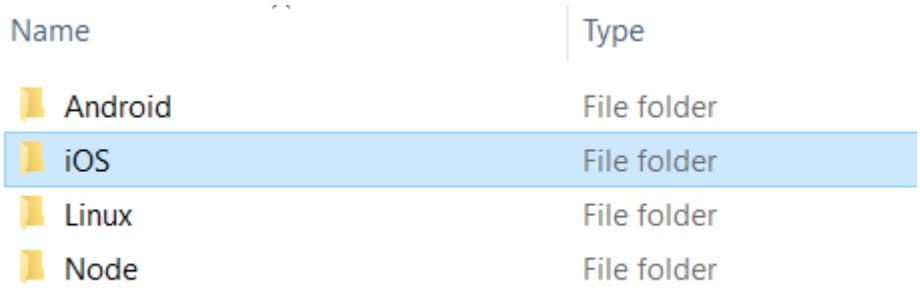
The result will be an executable **example**

Execute the example program. The following results should be displayed.

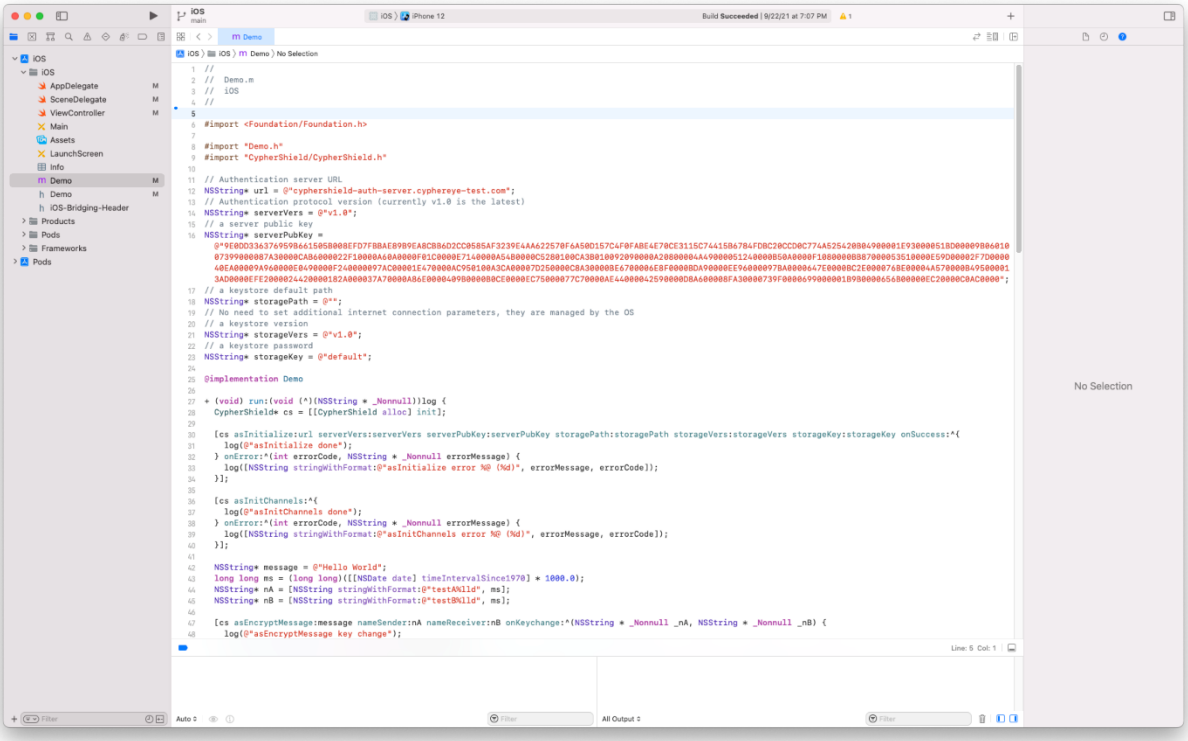
```
./example
asInitialize()
asInitialize() executed
result=0
asInitChannels()
  onkeychange()
  asEncryptMessage()
  onkeychange()
  asDecryptMessage()
  onkeychange()
  asResetSession()
  Message crypt/decrypt: OK
  as_GetChannelKeys()
  as_SetChannelKeys()
  as_GetChannelKeys()
ChannelKeys read/write: OK
  as_GetChannels()
  as_SetChannels()
  as_GetChannels()
Channels read/write: OK
as_DoneChannels()
```

Apple iOS

The examples provided in this iOS folder is a project implementing the CypherShield SDK in a working iOS app.

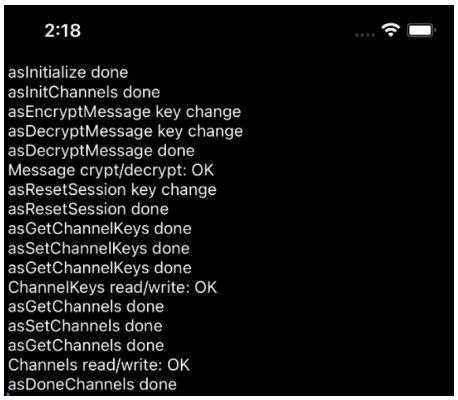


This folder will contain a XCode project with an example application. This app will show examples of how to include each of the major SDK functions into your own application.



Build the example iOS app.

When executing the app, the output should be displayed. This will be similar to the Linux native example program.

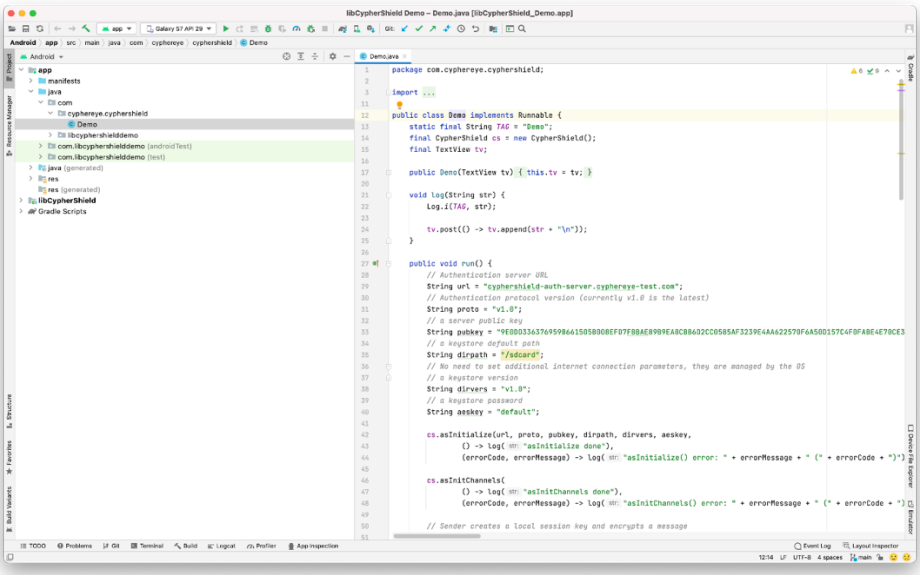


Android

The examples provided in this Android folder is a project implementing the CypherShield SDK in a working Android app.

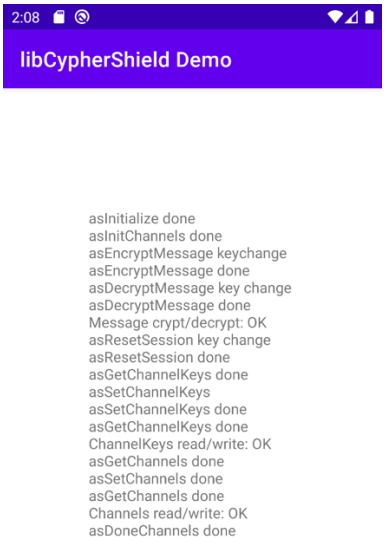
Name	Type
Android	File folder
iOS	File folder
Linux	File folder
Node	File folder

This folder will contain an Android Studio project with an example application. This app will show examples of how to include each of the major SDK functions into your own application.



Build the example Android app.

When executing the app, the output should be displayed. This will be like the Linux native example program.



Node.js

The CypherShield SDK includes an implementation on Node.js including the interfaces and wrappers appropriate to enable node servers to support advanced encryption technologies.