VSL Protocol

## Introduction

**Caption**

<Datatype Identifier>

<ChildDatatype:RootDatatype Identifier>

<!32> 'Constant length

<!uint> 'Variable length

<!flex> 'Flexible length, specified by the following packet

[Datatype RootIdentifier<On top encryption>]

{Datatype RootIdentifier<Data>}  
((Condition)Optional content)

# VSL 1.3

## General packet Ver. 1.0

<CryptographicAlgorithm:byte Algorithm>...

### CryptoAlgorithm

* None
* RSA\_2048\_OAEP // RSA-2048 with Optimal Asymmetric Encryption Padding
* AES\_256\_CBC\_SP // [Insecure] AES-256 CBC with split packets
* AES\_256\_CBC\_HMAC\_SHA256\_MP3 // AES-256 CBC with HMAC-SHA256, multipacket mode and 3byte length marker
* AES\_256\_CBC\_HMAC\_SHA256\_CTR

## Packet Plaintext Ver. 1.0

<!flex><byte PacketId>...

## Packet RSA-2048 OEAP Ver. 1.0

<!256>[<byte PacketId>...]

## Packet Insecure AES-256 CBC split packet Ver. 1.0

<!16>[<byte PacketID>(<uint PacketLength>)<byte[2+] Salt>...]  
(<!flex>[...])

## Packet AES-256 CBC HMAC-SHA256 MP3 Ver. 1.2

<!(Length + 1) \* 16><UInt24 Length><byte[32] HMAC><byte[16] IV>  
[<byte PacketId>...(<byte PacketID>...)]

## Packet AES-256 CBC HMAC-SHA256 CTR Ver. 1.3

The HMAC is computed only over the ciphertext as IVs are exchanged securely before.

<!(Length + 1) \* 16><UInt24 Length><byte[32] HMAC>  
[<byte PacketId>...)] 'No packet length indicator necessary

## Handshake - Client🡪Server Ver. 1.0

<!1><byte RequestType>

### RequestType

* DirectPublicKey

## KeyExchange RSA Client🡪Server Ver. 1.0

In VSL 1.2 the *ClientIV* and *ServerIV* are concatenated and used as HMAC key.

<!72><byte[32] AesKey><byte[16] ClientIV><byte[16] ServerIV>  
<ushort LatestVSL><ushort OldestVSL><ushort LastestProduct>  
<ushort OldestProduct>

## Certificate [allocated] Client🡨Server Ver. 1.0

## FinishHandshake - / AES Client🡨Server Ver. 1.0

A redirect has to be AES-encrypted to prevent man-in-the-middle-attacks.

<!uint><ConnectionState:byte ConnectionState>  
((ConnectionState == Redirect)<string Address><ushort Port>)

## FinishHandshake - / AES Client🡨Server Ver. 1.2

A redirect has to be AES-encrypted to prevent man-in-the-middle-attacks.

<!uint><ConnectionState:byte ConnectionState>  
((ConnectionState == Redirect)<string Address><ushort Port>)  
((ConnectionState == Compatible)<ushort VSLVersion>  
<ushort ProductVersion>)

### ConnectionState

* CompatibilityMode 'Use VSL protocol 1.1
* Redirect 'Not supported in VSL 1.1/1.2
* NotCompatible 'Connection Denied
* Compatible 'Continue using VSL 1.1 or specified version

## ChangeIV AES Client🡪Server Ver. 1.0

Notifies the server to change send and receive IV for the next packet. This packet is deprecated since VSL 1.2.

<!32><byte[16] ClientIV><byte[16] ServerIV>

## KeepAlive - Client🡨🡪Server Ver. 1.2

To request a keep alive response, send this packet and wait for a response with *Role* *Response*.

<!0><KeepAliveRole:byte Role>

### KeepAliveRole

* Request
* Response

## Accepted AES Client🡨🡪Server Ver. 1.1

<!3><bool Accepted><byte RelatedPacket>  
<ProblemCategory:byte ProblemCategory>

### ProblemCategory

* None

## OpenFileTransfer AES Client🡪Server Ver. 1.1

Requires an Accepted packet.

<!uint><Identifier Identifier><StreamMode:byte StreamMode>

### StreamMode

* GetHeader
* GetFile
* UploadFile

## FileHeader AES Client🡨🡪Server Ver. 1.1

A valid v1.1 FileHeader packet must contain at least 76 bytes with SHA or 44 bytes without SHA.

<!uint><string Name><ulong Length><uint Attributes>  
<DateTime CreationTime><DateTime LastAccessTime>  
<DateTime LastWriteTime><byte[] Thumbnail><byte[32] SHA256>

## FileHeader AES Client🡨🡪Server Ver. 1.2

The *HMAC* is computed using HMAC SHA-256 of the IV and ciphertext. Two 256bit keys for encryption and HMAC have to be provided by overlying protocol. Both HMAC of the packet content and SHA-256 of the file have to be verified by the client. An accepted packet is required to acknowledge a valid FileHeader which needs at least 78 bytes of size. If no AcceptedPacket is sent, the file transfer will time out. To cancel the file transfer after receiving this FileHeader, first acknowledge this packet and then send an AcceptedPacket with RelatedPacket 7 (OpenFileTransfer).

<!uint><ContentAlgorithm:byte Algorithm><ulong Length>  
<ContentAlgorithm:byte FileEncryption>  
((Algorithm.AES...)[*wHMAC*)  
<string Name><uint Attributes><DateTime CreationTime>  
<DateTime LastAccessTime><DateTime LastWriteTime><byte[] Thumbnail>  
<byte[32] SHA256>((FileEncryption.AES...)<byte[32] FileKey>)])

### ContentAlgorithm

* None
* AES\_256\_CBC
* AES\_256\_CBC\_HMAC\_SHA256

## FileDataBlock AES Client🡨🡪Server Ver. 1.1

Carries the data of a file. Each FileDataBlock packet has to be accepted by the receiver. Since version 1.3, this acknowledge is no longer necessary before sending the next block which will significantly improve performance.

<!uint><ulong StartPosition><byte[Length - 8] DataBlock>

## Identifier

<IdentifiactionMode:byte IdentificationMode>  
<IdentificationMode.Type Identifier>

### IdentificationMode

* UInt32
* UInt64
* ByteArray 'Max length of 64k
* String

## End-to-end-encryption

### AES\_256\_CBC (File encryption)

The Algorithm is sent in the FileHeader Packet alias FileMeta.

((Algorithm.AES...)<byte[16] IV>[)...(])

### AES\_256\_CBC\_HMAC\_SHA256

This Algorithm is implemented in *VSL.Crypt.AesStatic* as a high-level API. It is not only mentioned to be used in VSL, also the application built on VSL can make use of it.

<byte[32] HMAC><byte[16] IV>[...]

# VSL 2.0

## General packet (Early\_AES\_256\_CGM)

<uint Length><byte[12] IV>[<byte PacketID>]

<uint Length><byte[12] IV>[<ushort PacketID>]

<uint Length><byte[12] IV>[<byte StreamID><ushort PacketID>]