1. *Symmetric-Key* Encryption
   1. Introduction

It encrypts the communication between a sender and receiver using symmetric-key encryption.

* 1. Description
     1. Exchanging secret keys
* The program first starts a KeyManager process, which handles the key requests from sender and receiver processes. We assume that both the sender and receiver are already registered with the KeyManager.
* Sender starts a KeyClient process, which sends a KeyRequest message to KeyManager. The KeyManager authenticates the sender, creates a SharedKey, encapsulates it in KeyResponse message, and sends it to sender.
* Receiver also creates a KeyClient, which sends a KeyRequest to KeyManager. The KeyManager again authenticates the receiver, creates a SharedKey, encapsulates it in KeyResponse message, and sends it to receiver.
* If KeyManager can’t authenticate any process, it sends an empty KeyResponse and the respective process terminates itself on receiving null Key.
* Figure (1 &2) describes the process of exchanging secret keys.
  + 1. Message Communications between Sender and Receiver
* Before sending a protocol message, Sender encrypts the message with the SharedKey.
* After receiving the message, Receiver decrypts the Message with the SharedKey.
  1. Application Material
* Developers would be provided with the following classes:
* *Encryption:* A data structure containing elements to measure performance.
* *KeyManager:* It authenticates the processes and provides the shared key.
* *KMClient:* It sends the authentication information to the KeyManager and requests the shared key.
* *KeyRequest:* A protocol message used to request the SharedKey.
* *KeyResponse:* A protocol message used by the KeyManager to send the SharedKey.
* *SharedKey:* This class encapsulates the shared key information.
  1. Figures

# 

Figure 1: Data Structures for Symmetric-Key Encryption

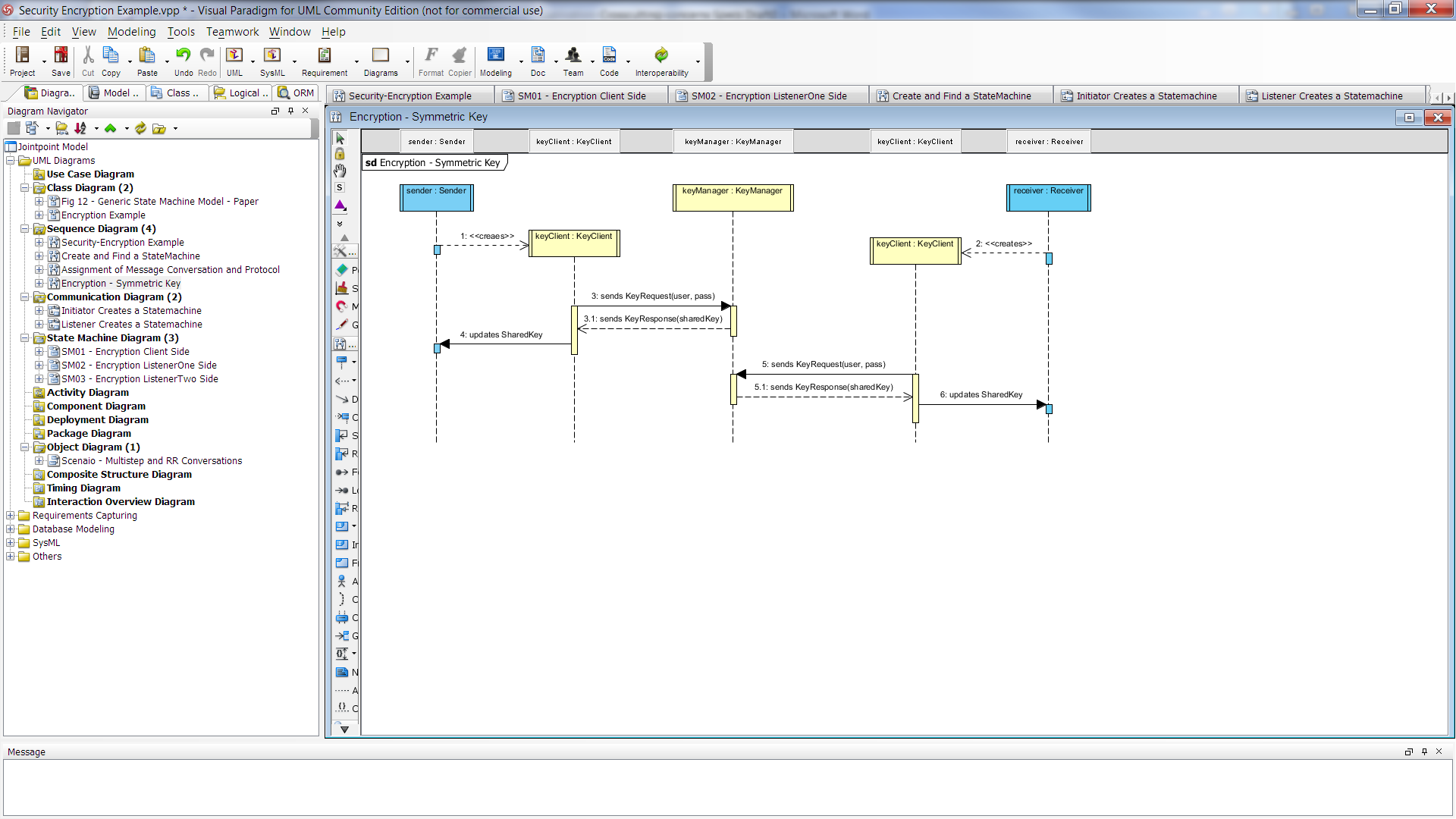


Figure 2: Process of exchanging shared keys