Secure Multipoint Message Delivery Protocol

Andrew Weinrich
10 June 2003

The Secure Multipoint Message Delivery Protocol is designed for applications that need a way to distribute arbitrary-size and -format messages to a number of hosts, while ensuring that the following three conditions are met:

- **Transmission security**
  Only intended recipients should be able to read messages
- **Sender verification**
  Recipients should be able to prove that a message came from the sender it claims to be sent from
- **Versatility**
  The protocol should support sending messages to multiple recipients, different levels of transport reliability, synchronous and asynchronous responses, and provide a method of indicating high- and medium-level purpose of the message

The first two goals are met by using public/private keys for signing messages and encrypting session keys. Messages are structured in a format that meets all the conditions of the third goal.

Security and Verification

To support transmission security and sender authentication, every host participating in the protocol must have at least one public/private key pair. The private keys will be used for signing messages, and public keys will be used for encrypting incoming messages.

Although one key pair is the minimum, it is allowable for every separate combination of sender, receiver, application, and function to have unique signing and transmission keys. It is the responsibility of the involved hosts to know which keys to use at what times.

The generally accepted method for using public/private keys to encrypt large amounts of data is to generate a symmetric session key, encrypt that session key with the recipient's public key, and use the symmetric key to encrypt the actual message. This hybrid approach gains both the versatility of public keys and the speed of symmetric algorithms. SMMDP will follow this model.

Message Format

As the name implies, SMMDP is designed to pass single-piece messages, and not streamed or interactive data. As a result, all transmissions are packaged into single units with a clearly defined format.

Message Data

The protocol is agnostic to the type of data being sent in messages, with two exceptions: each chunk of data is assigned an enumerated application code and function code. This allows for a system, for example the Content Delivery Subscription System, to effectively discriminate betweens both broad kinds of traffic and the meanings of individual messages.

For example, requests to the CDSS subscription server could have Application Code 1. Different requests (to allow a subscriber, to remove a subscriber, etc) would then have different function codes. Actual events between hosts could have Application Code 2, and different types of events would be assigned uniquefunction codes.

This level of description is minimal, and has no bearing on how the protocol actually treats messages. It exists only to allow a high-level filtering mechanism for purposes such as key exchange.

Message Format

SMMDP messages will have the following general format. Actual field size is dependent on the cryptographic algorithms used.

1. The IP address of the sender
2. Message configuration profile and flags. This field defines the layout and format of the following data message. Configuration information would include, but not be limited to, the following:
    1. Identifiers of encryption protocols used for session key encryption, digital signing, and symmetric message encryption
    2. Key sizes for above
    3. Number of recipients

   This information is necessary to allow recipients to properly process the message without prior negotiation.
3. The application code and function code of the message. These must be in plaintext, so that if the transmission and signing keys are dependent upon the combination of the two, the proper decryption/authentication keys can be selected.
4. At least one of the following sets (the exact number is specified in the configuration profile)
    - Receiver IP address
    - Message hash signed with the sender's signature key for this receiver, encrypted with the recipient's public key
    - Session key encrypted with the recipient's public key (for efficiency, may be included in the same encryption block as the signed hash)
5. The message data, encrypted with the symmetric session key.

Message Diagram

```
+------------------------------+
|       Sender IP Address      |
+------------------------------+
|Message Configuration Profile |
+------------------------------+
|        Application Code      |
+------------------------------+
|          Function Code       |
+------------------------------+
|     Receiver #1 IP Address   |
+------------------------------+
|Signed Hash enc w/ R1 pub key |
+------------------------------+
|Session Key enc w/ R1 pub key |
+------------------------------+
|     Receiver #2 IP Address   |
+------------------------------+
|Signed Hash enc w/ R2 pub key |
+------------------------------+
|Session Key enc w/ R2 pub key |
+------------------------------+
               /  /
              /  /
               /  /
              /  /
+------------------------------+
|                              |
|                              |
|                              |
|   Encrypted message data     |
|                              |
|                              |
|                              |
+------------------------------+
```

Delivery methods

Due to the wide range of uses for this protocol, the libraries must support several different ways of delivering messages:

- Reliable delivery to a single recipient
- Reliable delivery to a single recipient with a response (a la HTTP)
- Reliable delivert to multiple recipients
- Unreliable delivery to a single recipient
- Unreliable delivery to multiple recipients

This implies that the library will have to support transport via TCP, UDP, and various forms of multicast. The sending application specifies what transport characteristics are required when beginning a communication.

Implementation

Encryption System

Both private-key/public-key encryption (for the digital signatures and delivery security) and symmetric encryption (for the actual encryption of data) are needed for this protocol. The OpenSSL libraries are the best option for providing these facilities, for the following reasons:

- OpenSSL is found on every Linux installation, and many other Unix and Unix-like systems, so no additional distribution of source code or libraries would be needed
- While the OpenSSL libraries are written in C, wrappers exist for a number of other mainstream languages, including Perl and Python, allowing SMMDP applications to be written in the most convenient environment for each host
- OpenSSL supports all the necessary encryption technologies, including RSA public/private key pairs generation and encryption, symmetric encryption in a number of protocols (DES, 3DES, Blowfish, etc), and message hashing using MD5, SHA-1, and other methods

Key Exchange and Storage

In order for messages to be successfully transmitted, the sender must know all the transmission public keys of the recipients, and the recipients must all known the signing public key of the sender. This matter becomes more complicated when senders and receivers use different transmission and signing for various combinations of hosts, applications, and functions, as is allowed. A key exchange system is not included as part of SMMDP; it is assumed that for every message, the protocol library must be explicity informed of what keys are to be used, both public and private

Nevertheless, for networks with large numbers of hosts or function, some automated system of exchanging and recording the proper keys for a given situation is highly desirable. For an example of how to design a secure key exchange system for SMMDP, please see the Content Delivery Subscription System documentation.

Some kind of key storage system is also desirable to secure the host’s private keys on disk. Several system (such as encrypting all key pairs in a passphrase-secured file) are available, and one of these should be implemented at a level just above SMMDP.