CSSE4603/7032 Models of Software Systems

Assignment 1: Needham-Schroeder Protocol

Samuel Teed 43211915

April 4, 2016

Let the given set

$$[Comm]$$

denote the set of all possible communications.

Let *Message* and *EncryptedMessage* denote the sets of messages and encrypted messages, respectively.

$$Message, EncryptedMessage : \mathbb{P}\, Comm$$

The actual way a message or encrypted message is internally structured is of no concern. Then a key can be thought of as an injective (one-to-one) function that takes an element of type communication and converts it into an encrypted message. Hence we can define the set of all keys by

$$Key == Comm \rightarrowtail EncryptedMessage$$

Suppose the agents and server are identified via their names, and we have the given set

$$[Name]$$

of all possible names.

The following functions allow us to extract information from messages. For a request message sent to a server, *this* returns the name of the agent which sent the message, and *other* returns the name of the agent with which it wants to communicate. For a message with a key, *key* returns the key, and *other* returns the agent that can be communicated with using that key.

$$\begin{array}{l} this : Message \pfun Name \\ key : Message \pfun Key \\ other : Message \fun Name \\ \hline \mathrm{dom}\, this \cap \mathrm{dom}\, key = \varnothing \\ \mathrm{dom}\, this \cup \mathrm{dom}\, key = Message \end{array}$$

The predicate states that no message is both a request message (in the domain of *this*) and a key message (in the domain of *key*). Also, all messages are either request or key messages.

An agent is modelled as having a name, the name of the server (which is not an agent), and a table whose entries map names of other agents and the server to the key used to communicate with them.

Agent

$$\begin{array}{l} name, serverName : Name \\ table : Name \pfun Key \\ \hline name \neq serverName \end{array}$$

Initially, the table contains a single key for communication between the agent and the server.

Init

$$Agent$$

$$\operatorname{dom} table = \{serverName\}$$

An agent may request a key for communication with another agent for which it does not already have a key. This request is sent to the server via a non-encrypted message containing the agent's name and that of the other agent.

RequestKey

$$\Xi Agent$$
$$request! : Message$$

$$\exists\, receiver : Name \bullet$$
$$receiver \notin dom(table)$$
$$this(request!) = name$$
$$other(request!) = receiver$$

An agent may receive a key via an encrypted message from the server and add it to its table. This message also contains the name of the other agent with which the agent can communicate using the received key. A second encrypted message is also received from the server, decrypted and its further encrypted content is sent to the other agent.

ReceiveKeyFromServer

$$\Delta Agent$$
$$messageReceived1?, messageReceived2? : EncryptedMessage$$
$$decryptedMessage1, decryptedMessage2! : Message$$

$$messageReceived1? = table(serverName)(decryptedMessage1)$$
$$table' = table \cup \{(other(decryptedMessage1), key(decryptedMessage1))\}$$
$$messageReceived2? = table(serverName)(decryptedMessage2!)$$

An agent may receive a key via an encrypted message from another agent (but encrypted with the agent's key for communication with the server) and add it to its table. The message also contains the name of the other agent with which the agent can communicate using the received key.

ReceiveKeyFromAgent

$$\Delta Agent$$
$$messageReceived? : EncryptedMessage$$
$$decryptedMessage : Message$$

$$messageReceived? = table(serverName)(decryptedMessage)$$
$$table' = table \cup \{(other(decryptedMessage), key(decryptedMessage))\}$$