In this assignment you will implement various parts of a secure network protocol stack. Please follow the instructions carefully and turn in the results as directed before the listed deadlines. For this assignment you are free to create the application using any compiled language (e.g., C, C++, Java, ..., not Perl, Python, ...) and support libraries that come with the Ubuntu default installation, `build-essentials` or packages obtainable from `apt-get`. You should also feel to use the CSE545 infrastructure used in the previous assignments, but you should not feel bound to it.

Assignment #3 : due April 5, 2011, 11:59pm

1. Write code and makes files for a program `cse545link`. This program will support communication between two endpoints over UDP. The program specification is given by the following help output:

```
USAGE : cse545link <IP> <port> <cert> <name> [-v] [-h]
```

where:

- IP - the IP address of the other side of the communication
- port - the port of the other side of the communication
- cert - a file containing the X.509 certificate to be used for authentication
- name - a string name of the endpoint
- -v - verbose mode (output detailed information to log)
- -h - help (prints out this help information)

2. The endpoints will perform the following actions to initiate a session:

   (a) exchange identities
   (b) establish a session key
   (c) negotiate keep-alive period

   Thereafter, they will send a keep-alive message at the scheduled interval. Also send 100-byte random value every 1-10 sections (randomly distributed). This is the payload delivered by the protocol.\(^1\)

3. All payload data must be delivered reliably and totally ordered. This implicitly requires you detect packet loss, perform retransmissions, cache out of order packets, etc. You can achieve these goals in any way that makes sense, but good design, documentation, and interfaces will be rewarded with good grades.

   *Note*: these functions are **not** optional, as they are essential to the correct operation of the next assignment.

4. Design a suite of security protocols that makes
   a) the initial exchange protocol resistant from manipulation by the adversary, i.e., key negotiation with authenticated identities and session parameters,
   b) ensures the integrity, confidentiality, and authenticity of the payload and
   c) the ordering and reliability of payload delivery not manipulatable by adversaries.

   Note that you are going to use the certificate file passed into the program as initial authentication, then possibly use session keys to achieve security goals. You can use any cryptographic algorithms you wish for these purposes, but DSA (signatures), AES (encryption), and MD5 (hashing) are good choices.

   *Developer note*: it is wise to build the link layer implementation in as a module that you can access from later programs. You will be building on these for assignment #4, so be sure to be diligent in creating APIs.

---

\(^1\)Random link level payloads should be disabled in assignment #4.
Assignment #4: due April 28, 2011, 4:15pm

1. Write code and makes files for a program cse545network. This program will support an ad hoc network over the cse545link interfaces running on a single host. The program specification is given by the following help output:

   USAGE : cse545network <port> <port range> <cert> <name> [-v]

   where:
   port - the local port number to bind to
   port range - the scan port range, e.g., 6000-6010
   cert - a file containing the X.509 certificate to be used for authentication
   name - a string name of the endpoint
   -v - verbose mode (output detailed information to log)

2. The first phase of the protocol searches the local host for other clients. The client (cse545network program) will scan all ports (except its own) in the port range. When one is detected, it is used as the “graft” point. If no graft point is found, then the node can assume that it is the first element in the network, and can proceed to steady state operation.

3. Once a graft point has been detected, a connection to it will be established using the link layer. The graft point will forward its routing table held at graft point to the client. A sample table will contains entries:

<table>
<thead>
<tr>
<th>Time</th>
<th>Name</th>
<th>IP address</th>
<th>Port</th>
<th>Peers</th>
</tr>
</thead>
<tbody>
<tr>
<td>1300807247</td>
<td>Bob</td>
<td>192.168.27.4</td>
<td>6000</td>
<td>Alice John</td>
</tr>
<tr>
<td>1300807253</td>
<td>Alice</td>
<td>192.168.27.4</td>
<td>6001</td>
<td>Bob John</td>
</tr>
<tr>
<td>1300807122</td>
<td>John</td>
<td>192.168.27.4</td>
<td>6002</td>
<td>Bob Alice</td>
</tr>
</tbody>
</table>

   Each entry in the table is known as a route assertion, where time is the unix time at which the assertion was made. The route topology will always form a ring, in which each node has exactly two peers (unless there are fewer than 3 nodes in the network).

4. The client should arbitrarily pick one of graft point’s neighbor links to insert himself on; essentially expanding the ring by one. The client should establish communication to the neighbor and send messages to both indicating that it has inserted itself into ring. The neighbor and graft point should accept.

5. Each node that has had its peering relations changed (the client, the graft point, and the neighbor) should send a new route assertion to each client in the network and receive an acknowledgement of the receipt over the link payload interface. This should be performed by (recursively) forwarding a payload packet to the node on the path shortest to the destination (path vector) over the link payload interface.

6. Each node should send a 100-byte packet at a random interval (1-10 seconds apart) to a random node. This too should use the forwarding algorithm described in the preceding step.

7. You are to design cryptographic algorithms that perform achieve the following goals:
   (a) The authenticity and integrity of each route assertion must be maintained.
   (b) The authenticity and integrity of each control message must be maintained.
   (c) All nodes must be sure that they have the most recent assertion, and that old assertions cannot be replayed.
   (d) No node should be able to silently drop payload packets without being detected.
   (e) Any other properties you think would be desirable.
Both Assignments

1. For each assignment (#3 and #4), create a project document describing the design of your system, with a section on each of the layers. The document should:

   (a) Provide a brief description of each of the main modules
   (b) Detail the cryptographic protocols you define both in text and as message flows. A justification on why these protocols are complete and secure should be made, i.e., be sure to identify exactly. Make sure every property is discussed.
   (c) Describe how other properties (reliability, ordering, routing) are performed and why they are robust.

   The document should be very clear and reflect the actual workings of the code.

2. Create a zipped tarball file containing a single directory tree named assign3 (or assign4, respectively) including the code and document. Create a README that indicates the steps necessary to install the proper support libraries and build the assignment. Email the tarball to mcdaniel@cse.psu.edu by the assignment deadline. The tarball should be named LASTNAME-assign3.tgz (or LASTNAME-assign4.tgz), where LASTNAME is your last name.

3. Note: the last class of the semester (April 28) will be used for demo the course projects. More details as the event gets closer.

**Note:** Like all assignments in this class you are prohibited from copying any content from the Internet or sharing ideas, code, configuration, text or anything else or getting help from anyone in or outside of the class. Consulting online sources is acceptable, but under no circumstances should anything be copied. Failure to abide by this requirement will result dismissal from the class.