1. Please explain semantic and anti-semantic keys.

Semantic keys convey meaningful information (like a username in a database), making them predictable and human-readable, while anti-semantic keys are random, meaningless identifiers (like UUIDs) designed for uniqueness and security.

2. What is the meaning of a semantic key, and how can we use authentication and authorization with it?

Explained at question 1! It can be used to *identify* a user, you cannot use it directly for authentication or authorization.

- **Authentication**: A semantic key is not the secret. The user still needs to provide a password or another credential to prove they own that email address.
- **Authorization**: A semantic key can be used as a lookup to find a user's permissions.

3. What does a digital signature mean and how does it work?

A digital signature is a cryptographic mechanism ensuring data integrity and authenticity. It works by hashing a message, encrypting the hash with the sender’s private key, and attaching it. The recipient decrypts the signature with the sender’s public key, verifies the hash, and confirms the message is unchanged and from the sender.

4. What is the difference between hashing and encryption?

Hashing creates a fixed-size, irreversible digest (like SHA256) for data integrity, while encryption reversibly transforms data (like AES) for confidentiality using a key. Hashing verifies data hasn’t changed; encryption protects data from unauthorized access.

5. Please explain in detail the usage of ssh -R, ssh -L, and ssh -D.

```
-L [bind_address:]port:host:hostport
-L [bind_address:]port:remote_socket
-L local_socket:host:hostport
-L local_socket:remote_socket
 Specifies that connections to the given TCP port or Unix socket on the local (client) host are to be
 forwarded to the given host and port, or Unix socket, on the remote side. This works by allocating a socket
 to listen to either a TCP port on the local side, optionally bound to the specified bind_address, or to a
```

```
 Unix socket.  Whenever a connection is made to the local port or socket, the connection is forwarded over the
 secure channel, and a connection is made to either host port hostport, or the Unix socket remote_socket, from
 the remote machine.
```

```
-R [bind_address:]port:host:hostport
-R [bind_address:]port:local_socket
-R remote_socket:host:hostport
-R remote_socket:local_socket
-R [bind_address:]port
 Specifies that connections to the given TCP port or Unix socket on the remote (server) host are to be
 forwarded to the local side.

 This works by allocating a socket to listen to either a TCP port or to a Unix socket on the remote side.
 Whenever a connection is made to this port or Unix socket, the connection is forwarded over the secure
 channel, and a connection is made from the local machine to either an explicit destination specified by host
 port hostport, or local_socket, or, if no explicit destination was specified, ssh will act as a SOCKS 4/5
 proxy and forward connections to the destinations requested by the remote SOCKS client.
```

```
-D [bind_address:]port
Specifies a local “dynamic” application-level port forwarding.  This works by allocating a socket to listen
to port on the local side, optionally bound to the specified bind_address.  Whenever a connection is made to
this port, the connection is forwarded over the secure channel, and the application protocol is then used to
determine where to connect to from the remote machine.  Currently the SOCKS4 and SOCKS5 protocols are
supported, and ssh will act as a SOCKS server.  Only root can forward privileged ports.  Dynamic port
forwardings can also be specified in the configuration file.
```

6. What is a certificate? What does it mean to sign a certificate? Does a certificate simply represent a public key, or is it related to keys in another way?

A certificate is a digital document binding a public key to an identity (like a server). Signing a certificate means a Certificate Authority (CA) uses its private key to encrypt a hash of the certificate, verifying its authenticity. A certificate contains a public key, identity details (like a domain), and metadata, not just the public key, and is paired with a private key stored separately.

7. At the end of this LPIC-1 topic, please reflect on the purpose behind designing such a complex operating system. If you intended to contribute to this project, which part of the OS would interest you most, and which part would you change, and why?

My knowledge of the Linux kernel is limited as I haven't read its source code. However, I understand that its complexity stems from the need for flexibility to support diverse hardware.

Given my general interest in programming, I'd prefer to contribute to the process management and networking parts of the kernel. If I could change one thing, it would be to standardize the commands across all distributions. I wish that for the same purpose, such as package management or firewall configuration, we didn't have to use different commands and syntax.