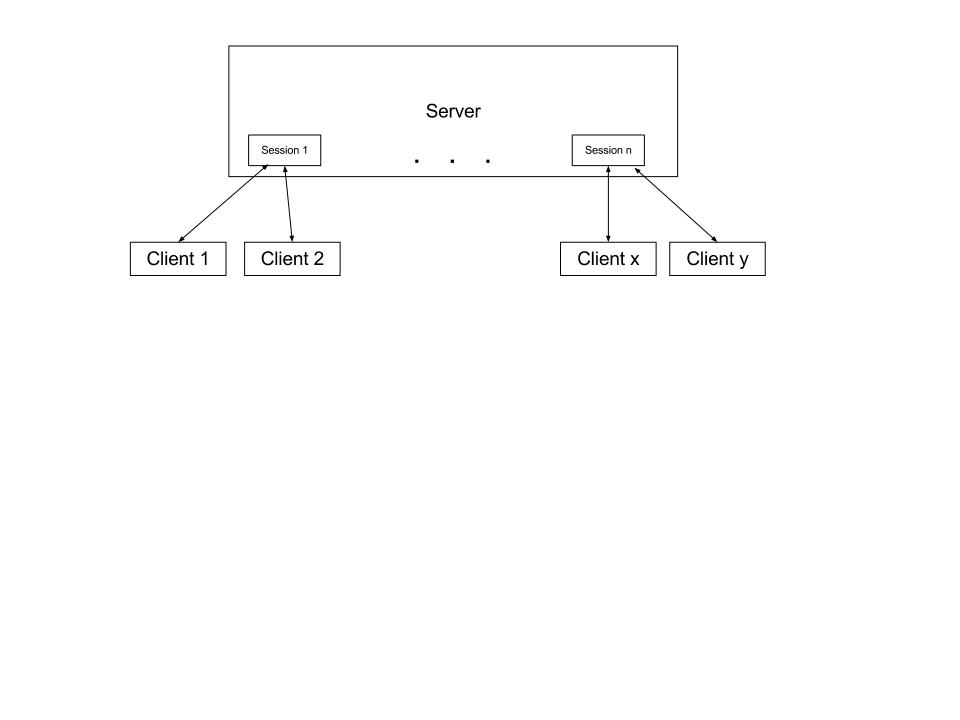
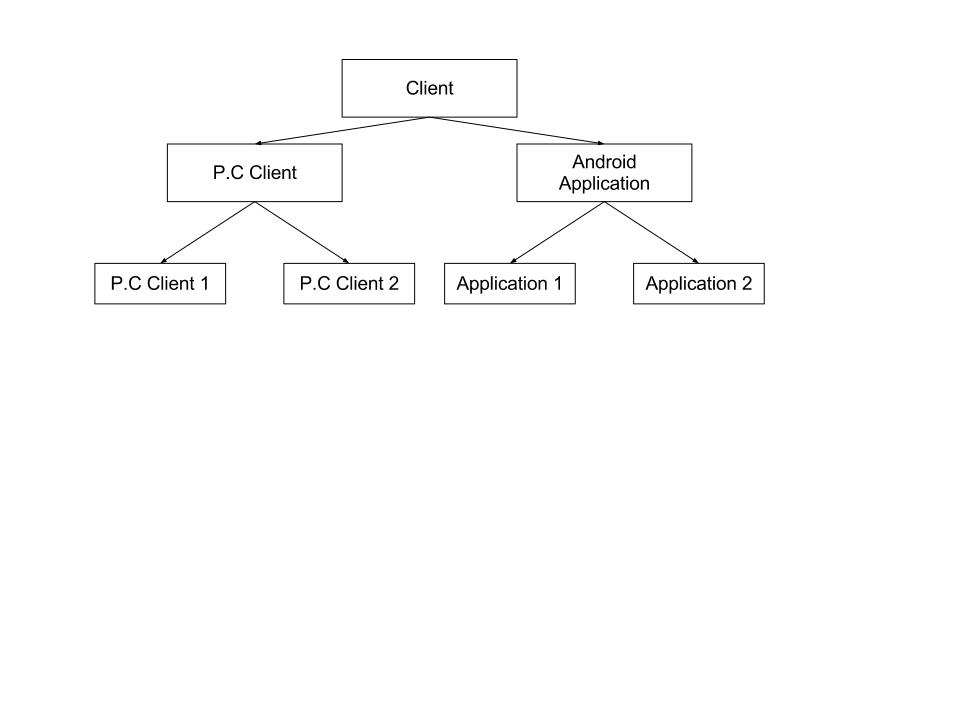
**Design**

As can be seen from the previous sections in this document, this project includes a software development section. The system is built upon a multi-client server facilitating the communication of data between two clients. This data is encrypted (and decrypted) using two different cryptographic schemes.



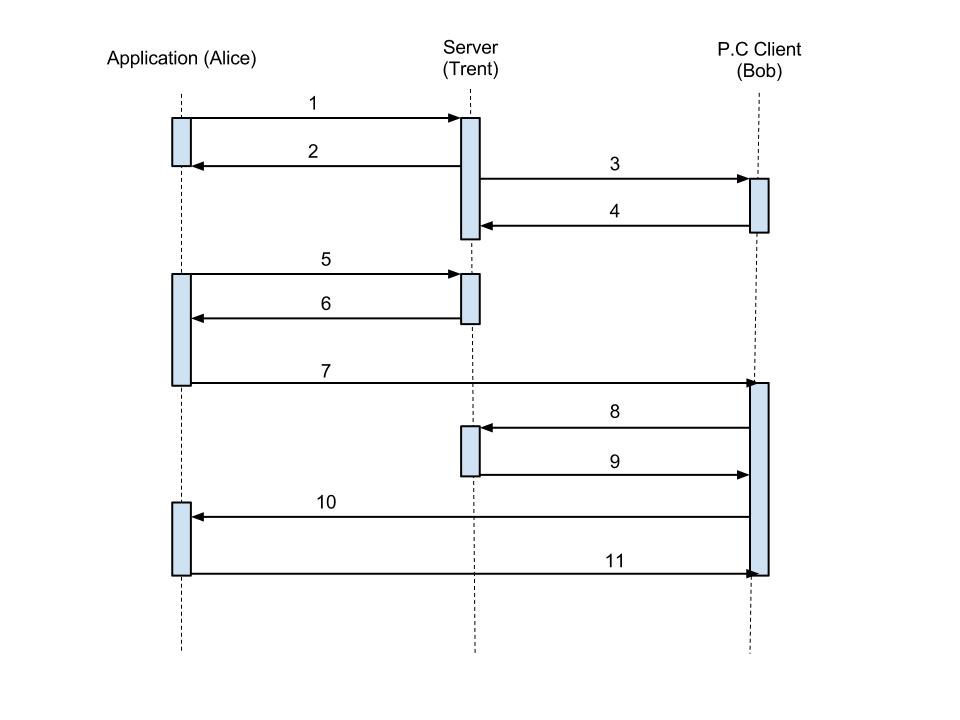
**Figure 1** The server can create many communication sessions from a collection of clients.

The server creates a session between two clients that wish to communicate. Each session is a new thread within the server. This allows the server to establish any number of sessions, to allow for multiple communication links between any two clients.



**Figure 2** Inheritance properties of the clients

Communication between two clients is facilitated via the server. A client can be either a P.C client or an Android Application (client) allowing data to be sent between these two mediums. Compatibility between the clients will be ensured through the use of the XML file structure. For example, if two cryptographic schemes are implemented; Application 1 and P.C Client 1 will utilise scheme 1 and Application 2 and P.C 2 will utilise scheme 2. This can be easily achieved due to the usage of the model in **Fig 1.** The main Android Application sends unencrypted data to the server and receives unencrypted data back. Application 1 (and 2) inherits these features, with the ability to add in their respective methods for encryption. This is the same for the P.C clients.



**Fig 3** Sequence diagram showing distribution of public keys

The encryption schemes used within this project all require the sharing of public keys. **Fig 3** shows how this will be achieved. The placeholder names ‘Alice’, ‘Bob’ and ‘Trent’ have been used following the tradition set out in most cryptography materials.

1. Alice registers her public key KA with Trent
2. Trent sends his public key KT to Alice
3. Bob registers his public key KB with Trent
4. Trent sends his public key KT to Bob
5. Alice sends to Trent: Alice, Bob, Timestamp1
6. Trent sends to Alice: {KB,Bob, Timestamp1}KT-1
7. Alice checks Trent’s signature on “KB,Bob” and the timestamp, creates her nonce NA at random and sends to Bob: {NA , Alice}KB
8. Bob decrypts the message, checks Alice’s ID and sends to Trent: Bob, Alice, Timestamp2
9. Trent sends to Bob: {KA , Alice, Timestamp1}KT-1
10. Bob checks Trent’s signature on “KA, Alice” and the timestamp, creates his nonce NB at random and sends it to Alice: {NA, NB, Bob}KA
11. Alice decrypts and sends to Bob {NB}KB

(Steps 7, 10 and 11 passed through server and directed straight to client)

Note: {M}Kx-1 represents the encryption of message M with the private key of client x, whilst {M}Kx represents the encryption of message M with the public key of x.

Once this protocol has been completed, each client has the other client’s public key and the communication of encrypted data can commence. Steps 1 to 4 are only required when a connection is made between two previously unconnected clients is made or when requested by the user. The user can request to generate a new public-private key pair which will need to be shared again with any client they wish to communicate with.