1. The additional security that Station to Station Protocol (STS) has compared to Diffie Hellman is that it includes an authentication step which can help reduce man in the middle attacks. With a similar first step of sharing $g^a mod\, p$ and $g^b mod\, p$. Then both parties will sign both the values of $g^a mod\, p$ and $g^b mod\, p$ using their private key. They then subsequently encrypt the digital signature with the session key $k$, $k = g^{ab} mod\, p$ and append their certificate. This allows both parties to verify the authenticity of the other party by checking the validity of the certificate and if the signature is signed with the corresponding public-private key pair. This helps to reduce Man in the Middle attacks as impersonation attempts can be largely reduced.

5

2. No a commitment scheme of $C(m) = H(r \otimes m)$ cannot provide the same security guarantees as it breaks the binding security property of the commitment. There exists another $r', m'$ pair where $r \neq r'$ and $m \neq m'$ but create a commitment $com = H(r \otimes m) = H(r' \otimes m')$. An example would be $r = 0000$, $m = 0000$ and $r' = 1111, m' = 1111$, both which would XOR and form $0000$. An attacker can first send over the commitment of the first pair $r, m$. But later at the verification stage send over the other pair $r', m'$ and the verifier would receive the same commitment.

5

3. This protocol does not satisfy perfect forward secrecy because an eavesdropper can sniff the traffic and have both the messages $A \rightarrow B$: "$I\ am\ A$" || $R$ and $B \rightarrow A$: $E(k, R||K_s)$. After knowing the long-term shared key $k$, the eavesdropper can first decrypt the message sent from B to A to get $R||K_s$. The eavesdropper then can find out $R$ from the first message and eventually get the session key $K_s$ which they can use to decrypt the session traffic with.

5

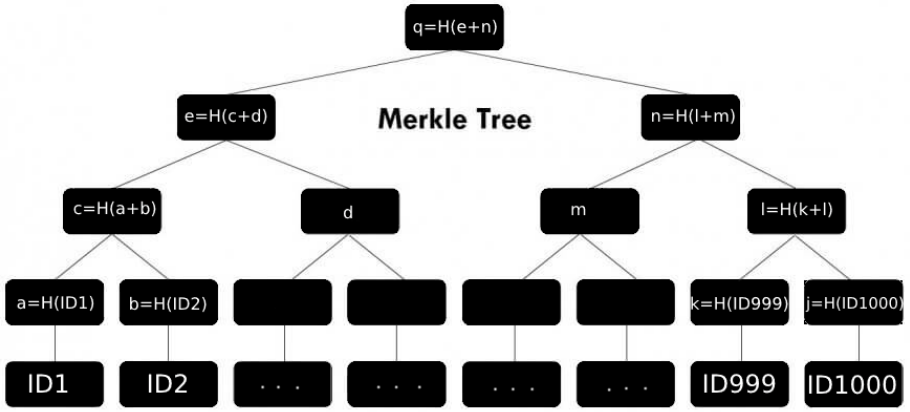
4. C and V can use an encrypt-then-mac scheme to communicate to ensure confidentiality and integrity of the message. C can calculate the ciphertext $c = E(K_{C,V}, p)$. Then C can send V a print job $p$ in this form: $c$ || $HMac_{K_{C,V}}(c)$. The encrypted ciphertext $c$ can help provide confidentiality for the print job and the Keyed HMac can help provide integrity protection to the print job. The printer V needs to first verify the integrity of the ciphertext $c$. If it is correct, V then decrypts the ciphertext $c$ and prints the print job $p$.

5

5. Each leaf node of the Merkle hash tree can be the personal information entries of the employees. Then each entry would be Hashed using a secure hash (eg: SHA256). Hashes of neighbouring entries (eg: ID1 and ID2) would then be concatenated and then hashed again to form the node that connects both their nodes. This would then repeat until the root hash. Alice will only have to store the ID number, the corresponding entry's hash locally and the hash of the root node. To verify data authenticity, Alice will have to check if the hash of the entry corresponding to the ID1 (hash a) is the same as the locally stored hash value. If it is, Alice then works up the Merkle Hash Tree to verify if the hash of the node above is the same as hashing the concatenation of hashes of the ID

8

and its neighbour ID. Alice would then repeat this procedure until the root hash and if they are all correct, this would prove that the data is authentic. A picture depicting this process is shown below using the example of ID1.




6. 1- Eve can first decrypt the initial message from KDC to Alice. This allows Eve to know $K_{a,b}$ and the ticket. Eve can successfully impersonate Alice by replaying the same ticket given previously, and encrypting a new nonce with the same session key $K_{a,b}$. This would thus look like $Eve \rightarrow Bob$: $ticket$ || $E(K_{a,b}, N_E)$. Bob would verify the ticket by decrypting the ticket with his long-term key with the KDC $K_{b,kdc}$ and be convinced that the ticket is valid.

5 2- A possible counter measure would be for the KDC to append a timestamp and TTL to the end of the ticket to give the ticket a limited lifespan. This would look like $ticket = E(K_{b,kdc}, K_{a,b}$ || "$ID_A$" || $TS$ || $TTL)$. Thus when Bob decrypts the ticket using his long-term key with the KDC $K_{b,kdc}$, he is able to check if the ticket has expired and can respond accordingly.

7 2 1- An attack should not be possible. However, if Eve knows exactly Alice's ID and the padding used, the 1st block of plaintext will always be the same. This way, knowing the IV and exact plaintext, Eve can find out the secret key.

4 2- An attack is possible because in counter mode, the 1st block does not affect the 2nd block. Hence, using B as an oracle to determine if an input is correct, Eve can XOR different values to the 2nd block of ciphertext to change the IP address section to her own IP address.

2 3- Since the attacks affect integrity and the predictability of the text, changing to an encrypt then mac scheme can help to ensure integrity of the message.

8. C can intentionally misissue a certificate onto the CT, then they can have an employee to be the first one to report themselves. This would mean that C would be able to earn $1500 from reporting themselves and although they forfeit the $1000 premium, they will still earn a net total of $500.

10

9. 1- This is not a secure ISN because it will always be the same since the Hash of the permanent secret key $K$ will always be the same. This will make the ISN predictable. **3**

2- This is not secured. An attacker can know what the ISN is for a specific Source IP (SIP), Destination IP (DIP), Timestamp (ts) Tuple. Assuming that the destination IP is the server and is always fixed, the attacker can find another source IP timestamp pair where $SIP \otimes ts = SIP' \otimes ts'$ or $SIP \otimes ts = SIP \otimes ts'$. This means that the attacker, knowing an ISN for a fixed SIP, ts pair, can generate other SIP ts pairs to impersonate another victim and predict their ISN, or for the same victim but at a different time. **0**

3- This is not secure since an attacker that knows the Source IP and port, Destination IP and port, can predict an ISN of any time that they want since this ISN computation does not have the secret key. **3**

4- This is not secure. Assuming that the ISN is generated using a function similar to this. generate_ISN(seed) { return PRG(seed); } This means that creating an ISN would always run this function and hence always return the same value, making the ISN predictable. **3**

5- This is not secure because the encryption is deterministic. Encrypting the same numbers will always give the same result. Moreover, whenever the counter spills over the integer limit, the attacker will already know the ISN to use next. This can help make predicting the next ISN predictable since the attacker can keep requesting against the network to increase the counter and force the counter to reset. From there, the ISN will start to repeat again. **0**

10. a)

1- A DoS might not work since B might not have sent a packet to A recently hence the shut-off message will not work. **3**

2- A DoS might not work. M would not be able to send a packet this way to impersonate A. M does not have the private key of A and thus would not be able to sign the correct verification packet **0**

b)

1- A DoS might not work as Server S is able to disable the shutoff protocol on their NIC. **0**

2- A DoS might not work. Unless the Distributed network of DOS attacks on S are able to send more than 7 million packets in 5 minutes to cause a 5 minutes interruption, it might not be feasible. However, a NIC can also clear its bloom filter very frequently (every 30 seconds) meaning that server S would have reduced memory requirements storing all the messages it has received. **0**

3- Yes a DoS might be possible. If M and M' are able to send a significant amount of network traffic on the link S uses, S might not get a turn to send. **3**

11. 1- Since the honeypot will be an SSH honeypot, generating artificial traffic to this SSH honeypot could attract attackers to attempt to access this honeypot as well. This is because this honeypot will look like a real SSH server with connections going to it, making it less suspicious as a honeypot for the attackers. **10**

2- Since Cowrie is a High Interaction HoneyPot, ensuring that the honeypot does not have too much free space on the hard drive can increase traffic. This is because an empty disk could indicate to attackers that it is not a well-used production drive.
3- The honeypot could run relevant old and vulnerable SSH version. This can attract more traffic as many attackers would be scanning systems running vulnerable services to attack.