Key Distribution

Q1: Why is key distribution a problem? How can we share keys with friends in the crypto class? How many keys would each of us need to keep to be able to securely communicate with each other in the class? How many keys would be needed in total?

A1: In the schemes we have seen so far, the private keys existed on both sender and the receiver. In practice, how would this be realized in the first place? Even if it was possible, for N parties, each party would need to store on the order of $O(N)$ keys! The initial sharing can happen by literally meeting beforehand and handing the key, or in another case use a trusted third-party, which in this case might be Alptekin hoca. In that case he would store N keys, and we would obtain the key we need to communicate with our friends by first querying it securely from Alptekin hoca.

Q2: How can I share a key with my friend living in Australia?

A2: You can:

- Go to Australia and hand the key to your friend.
- Your friend can come here and obtain the key.
- If you are both a member of a Key-Distribution Center, you can obtain your keys from there.
- You can refer to a Certificate Authority, and do a Diffie-Hellman Key Exchange.

Q3: Are there any trusted third parties in the world, that are used today? Provide multiple examples. Discuss the trust put on those parties, and the possible amount of damage if they misbehave.

A3: How to arrange for TTP is actually unsolved. There may be certificate authorities (CA) for this purpose. An example is DigiNotar (now defunct). There are also free non-profit CA's such as CAcert and Let's Encrypt. The biggest CA is IdenTrust. If the CA can be broken or subverted, the security is completely lost. Impersonation attacks may becomes possible. An example is with VeriSign CA, that issued two certificates to a person claiming to be Microsoft. Later in 2011 Comodo and DigiNotar was compromised by Iranian hackers.

Q4: Describe a man-in-the-middle attack. Why is the Diffie-Hellman protocol insecure against a man-in-the-middle attack? Talk about some other protocols insecure against a man-in-the-middle attack. Also discuss some measures that can be employed to make the Diffie-Hellman protocol resistant to a man-in-the-middle attack.

A4: An example is when the adversary is doing an attack-in-the-middle by changing y_A to 1 and y_B to 1. This way, both parties will agree on a key, but the keys are known anyways, they will be 1. Another attack is when the attacker impersonates both parties and runs the protocol with both of them, therefore knows the key. Then, the attacker will stay in the middle and forward the messages in between by decrypting and encrypting and so on.

To avoid this, we might need a certificate authority, or again some kind of trusted third party, which provide a secure channel for the parties to execute the protocol over.

Q5: Consider the following key-exchange protocol:

a. Alice chooses uniform $k, r \in \{0,1\}^n$, and sends $s := k \oplus r$ to Bob.
b. Bob chooses uniform $t \in \{0,1\}^n$, and sends $u := s \oplus t$ to Alice.
c. Alice computes $w := u \oplus r$ and sends w to Bob.
d. Alice outputs k and Bob outputs $w \oplus t$.

Show that Alice and Bob output the same key. Analyze the security of the scheme (i.e., either prove its security or show a concrete attack).

A5: For every ranodm bit-string a, I will also denote the party that it was created at, for example Alice chooses $a \in \{0,1\}^n$ will be a_A.

- $s_A = k_A \oplus r_A$
- $u_B = s_A \oplus t_B = k_A \oplus r_A \oplus t_B$
- $w_A = u_B \oplus r_A = k_A \oplus r_A \oplus t_B \oplus r_A$
- Alice outputs: k_A.
- Bob outputs: $w_A \oplus t_B = k_A \oplus r_A \oplus t_B \oplus r_A \oplus t_B = k_A$

It is thus shown that they both output the same key k_A.

To show the security, let us check the key-exchange experiment. The transcript is: (s_A, u_B, w_A) and the common key is k_A. A bit is chosen $b \leftarrow \{0,1\}$ and if $b = 0$ then $k' = k$, otherwise $k` \leftarrow \{0,1\}$. An adversary $\mathcal{A}$ is given the transcript (s_A, u_B, w_A) and k'. Then it outputs $b' \in \{0,1\}$. The probabilitiy that $b' = b$ should be negligible in the security parameter for all PPT adversaries $\mathcal{A}$.

We will not show an adversary that can win this game: $\mathcal{A}$ first does $s_A \oplus t_B = k_A \oplus r_A \oplus k_A \oplus r_A \oplus t_B$ and obtains t_B. Then, it does $w_A \oplus t_B$ and obtains k_A. It then check whether $k_A = k'$. If $b = 0$ then $\mathcal{A}$ wins with 1 probability. If $b = 1$, then $\mathcal{A}$ wins with $1 - \texttt{negl}(n)$ probability, where the negligible comes from the fact that uniformly chosen k' is actually equal to k. As a result, $\Pr[\mathcal{A} \text{ wins}] = 1 - \mathtt{negl}(n)$, therefore this key-exchange protocol is not secure.