Network Security

Professor Gaurav S. Kasbekar
Department of Electrical Engineering
Indian Institute of Technology, Bombay

Week - 04
Lecture - 24
Authentication: Part 5

Hello, in this lecture, we will discuss authentication using a key distribution center. So, what is the need for a key distribution center? Suppose two network nodes, say Alice and Bob, want to securely communicate over a network. Then, we have seen that they need to first agree on a shared secret key, let's call it K_{AB}. So, subsequently, they can authenticate using one of the protocols that we discussed.

- Recall: they need to first agree on a shared secret key K_{AB}

So, how do Alice and Bob agree on a shared secret key, K_{AB}? One way is to use public key cryptography. But this requires a public key infrastructure. We'll discuss public key infrastructure in subsequent lectures. But for now, we assume that no public key infrastructure is available.

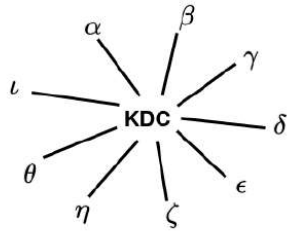
So, in general, public key infrastructure may not always be available. So, we also require authentication protocols in the absence of public key infrastructure. So, suppose there is no public key infrastructure available, we want a mechanism using which any two nodes in a network can agree on a shared secret key. So, one way to do this is to use a trusted node called a Key Distribution Center (KDC). So, this is a trusted node at which the secret keys of all the users are stored.

That is key distribution center shares a secret key with every node in the network, say Alice, Bob, and so on. There may be a key distribution center present in a local area network, which facilitates secure communication between all the nodes in the local area network. So, this key distribution center is an entity that is present inside a network, such as a local area network, which facilitates secure communication among the nodes of that network. Whenever a new node is installed in the network, the new node and the KDC are configured with a shared secret key. This configuration may be done either manually, for

example, the system administrator might type a secret key into the KDC and the same secret key into the node, or via post, and so on.

So, if the KDC is far away from the newly added node, then the secret key might be sent by post to the new node. So, using one of these ways, the newly added node and the KDC are configured with a shared secret key. This shows the KDC along with different nodes in the network. This is the first node in the network. It shares a secret key, α, with the KDC.

- If node α wants to talk to node β:
  - ❑ α connects with KDC and asks for a key with which to talk to β
  - ❑ KDC authenticates α, chooses a random number, say $R_{\alpha\beta}$, and sends $R_{\alpha\beta}$ after encrypting it to α
  - ❑ KDC also encrypts and sends $R_{\alpha\beta}$ to β, with the instruction that it is to be used for communicating with α
    - ○ Alternatively, KDC encrypts $R_{\alpha\beta}$ and gives it to α for forwarding to β
  - ❑ Now α and β have a shared secret key $R_{\alpha\beta}$; they mutually authenticate and then start exchanging data
- Above is an outline with some details omitted; we will later discuss, in detail, protocols for KDC-mediated authentication



This is another node in the network. It shares a secret key, β, with the KDC. So, for example, only this node and the KDC know the secret α. Only this node and the KDC know the secret β, and so on and so forth. So, every node shares a secret key with the KDC.

Now, if node α wants to talk to node β, notice that we are denoting the nodes by $\alpha,\ \beta,\ \gamma$, and so on, as well as their secret keys by $\alpha,\ \beta,\ \gamma$, and so on. So, if node α wants to talk to node β, first α connects with the KDC because the KDC knows a secret key with β, as well as one with α. So, α can securely communicate with the KDC. So, α securely communicates with the KDC and asks for a key with which to talk to β. Notice that α cannot directly contact β because it does not have any shared secret key with β yet.

So, then the KDC authenticates α. The KDC can authenticate α because α and the KDC have a shared secret key. Then, the KDC chooses a random number. Let's call it $R_{\alpha\beta}$ and then sends $R_{\alpha\beta}$ after encrypting it to α. So, the KDC selects a random number $R_{\alpha\beta}$, which will serve as a secret key for the communication between α and β.

And the KDC encrypts that key $R_{\alpha\beta}$ and sends it to α. The KDC also encrypts and sends the same key $R_{\alpha\beta}$ to β with the instruction that it is to be used for communicating with α.

So, once this is done, α and β both know the secret key $R_{\alpha\beta}$ and they can use it for communication among themselves. Another way is this: instead of the KDC having to contact the node β separately, the KDC can encrypt $R_{\alpha\beta}$ and pass it on to α for forwarding to β. So, when α contacts the KDC for a secret key, the KDC provides $R_{\alpha\beta}$ to α as well as encrypts $R_{\alpha\beta}$ using β's key and passes it to α, and then α sends that key to β.

Subsequently, α and β have a shared secret key, $R_{\alpha\beta}$, and they can securely communicate. At this point, α and β have a shared secret key, $R_{\alpha\beta}$; then they mutually authenticate and start exchanging data. They can mutually authenticate using one of the protocols we discussed in previous classes. This procedure is an outline, with some details omitted. We will later discuss, in detail, protocols for KDC-mediated authentication.

First, let's discuss the advantages and disadvantages of a KDC. Some advantages are when a new user is being installed into the network or a user's key is compromised and needs to be changed, then there is a single location, that is the KDC that needs to be configured. So, an alternative is when a new user is installed into the network; the user's information needs to be installed at every server with which the user might require access. This is difficult. So, for example, if a user accesses an email server, a web server, a file server, and so on, in the absence of a KDC, we might require manually configuring the user's secret key at each of these servers with which the user might communicate.

So, it is easier to just configure the secret key of a newly added user at a single point, namely the KDC. So, that's one advantage of a KDC. There is only a single location, the KDC, that needs to be configured whenever a new user is added to the network. But the KDC has some disadvantages. So, these disadvantages are common to a scenario where there is a central entity responsible for functions in a network.

So, these disadvantages are as follows. If the KDC is compromised, then all the network resources are vulnerable. So, note that the KDC has a secret shared key with every node in the network. So, if the KDC is compromised, suppose its database is hacked, then the secret keys with all the nodes in the network are lost. So, they are known to the attacker.

Another disadvantage is that there is a single point of failure. Suppose the KDC fails, for example, it crashes, then no new communication can be initiated. But keys previously distributed can continue to be used. However, if the KDC fails, then if Alice wants to initiate communication with some user with whom she has never communicated, for example, Carol, say, in that case, Alice cannot initiate a communication with another user with whom she has never communicated. So, that's another disadvantage of a KDC.

Another disadvantage is that it can be a performance bottleneck because whenever a node wants to communicate with another node, it has to first contact the KDC. So, all network nodes frequently need to communicate with the KDC. So, it can be a performance bottleneck. So, all these disadvantages arise from the fact that there is a central entity that is responsible for some functions. So, these are the disadvantages of a KDC.

Here is an alternative that overcomes the second and third of these disadvantages. That is, a single point of failure and performance bottleneck. Instead of only one KDC, we have multiple KDCs, all of which share the same database of keys. So, there are multiple copies of the same KDC at different locations. All of them share the same database.

So, this solution improves upon these two disadvantages. Now, there is no longer a single point of failure. If one of the copies of the KDC fails, then one can always connect to another copy of the KDC. It is also not a performance bottleneck because a user is free to connect to any of the copies of the KDC to get a secret key with another user. So, the performance bottleneck disadvantage is elevated.

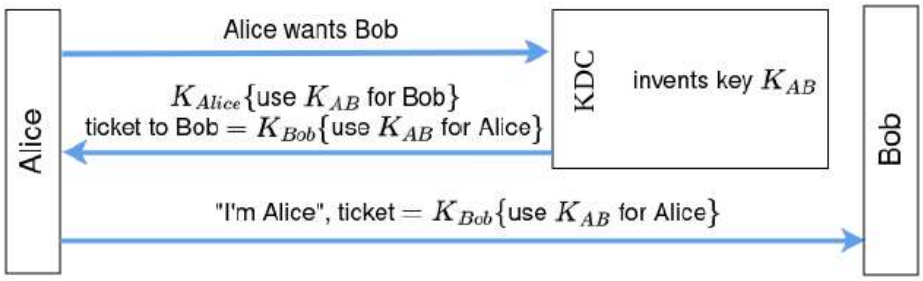
But there is a disadvantage that all copies of the KDC need to be protected. Even if one copy of the KDC is compromised, then the secret keys, all the secret keys in the database are lost. So, the secrets at all the copies of the KDC are compromised because the same secrets are stored at all the copies of the KDC. There is another disadvantage: there is additional cost and complexity. We need multiple copies of the KDC.

Each one incurs some cost and complexity. So, that's a disadvantage. Another disadvantage is the need for replication protocols. So, the same keys have to be stored at all the copies of the KDC, and whenever a key is changed, the change needs to be implemented at the other copies of the KDC as well. So, we require some replication protocols, which maintain synchronism between the keys stored at all the KDCs.

So, it ensures that at any point in time, the same set of secret keys are stored at all the copies of the KDC. These are some disadvantages of a KDC. Now, we will discuss mediated authentication using the KDC. Consider a network in which every user has a secret shared

key with a KDC. Suppose a user, Alice, wants to communicate with Bob; then the protocol shown in this figure can be used.

First, Alice sends a message to the KDC saying that she wants to communicate with Bob. Then, the KDC invents a key, let's call it K_{AB}, and encrypts the key using the secret key that the KDC shares with Alice, that is K_{Alice}. So, the KDC encrypts this message with the key $K_{Alice}\{\text{use } K_{AB} \text{ for Bob}\}$. So now, Alice has got the key K_{AB}, which she can use to securely communicate with Bob. And then, the KDC also sets up a connection with Bob and sends the same key K_{AB} to Bob after encrypting it with the secret key K_{Bob}, which the KDC shares with Bob.



So, at the end of these three messages, Alice and Bob both know the secret K_{AB}. But this has a shortcoming: it increases the load on the KDC. Whenever any user contacts the KDC to get a key, then the KDC has to initiate another piece of communication to the user with which the first user wanted to communicate. So, this increases the load on the KDC, and as we have seen, the KDC can be a performance bottleneck. Hence, the load on the KDC will increase, and it will start providing service slowly to the new connections being initiated to it. So, this is one shortcoming: it increases the load on the KDC since it needs to initiate a connection with Bob and share the K_{AB} with him.

So, an alternative to this procedure is as follows. In the message from the KDC to Alice, that is, this message, a 'ticket' is included, which Alice needs to send to Bob, and this ticket contains the key K_{AB}, encrypted using the secret key K_{Bob}. So, instead of the KDC having to separately contact Bob, the KDC just provides a ticket to Alice, and then Alice can subsequently connect with Bob and share the ticket obtained from the KDC with Bob. The ticket will make known to Bob the secret key K_{AB}, and subsequently, Bob and Alice can securely communicate. So, this figure shows the revised protocol, which uses a ticket instead of the KDC having to initiate communication with Bob.

So, the first message is the same as before. Alice sends a message to KDC saying that she wants to communicate with Bob. Then the KDC sends this message: K_{Alice}\{use K_{AB} for Bob\}. So, this part of the message is the same as in the previous protocol. Along with this message, the KDC sends a ticket to Bob.

The ticket to Bob is this message, {use K_{AB} for Alice}, encrypted using the secret key K_{Bob} that the KDC shares with Bob. So, the KDC sends this message, K_{Alice}\{use K_{AB} for Bob\}. This part of the message is meant for Alice, and this part of the message, the ticket to Bob, is meant for Bob. Then, subsequently, Alice initiates a communication with Bob. She sends the message 'I am Alice' along with this ticket, which Bob can use to obtain the key K_{AB}.

Notice that this ticket cannot be decrypted by Alice because it is encrypted using the key K_{Bob}, which only Bob and the KDC know. Once this ticket is sent to Bob, Bob can decrypt it and obtain the key K_{AB}. Now, Alice and Bob have the secret shared key K_{AB}, which they can subsequently use to authenticate among themselves and communicate. After the protocol shown in this figure is used, mutual authentication between Alice and Bob needs to be performed. So, this can be performed using one of the protocols that we discussed for mutual authentication.

So, the protocols that we discussed assume that Alice and Bob have a secret shared key K_{AB}. So, at the end of this message, Alice and Bob have a secret shared key K_{AB}. They can then use one of the protocols that we discussed for mutual authentication. But this protocol has some vulnerabilities. One example is the following.

Consider an intruder, Trudy, who stole Alice's key and also recorded this message 2 when Alice contacted the KDC to talk to Bob. Later on, Alice changed her key, but Trudy can still use the old key and the recorded message 2 to impersonate herself as Alice to Bob. So, this is an example of a vulnerability in this protocol. So, the disadvantage of this protocol is that, suppose we call the old key of Alice as J_{Alice}. Suppose Trudy stole the key J_{Alice} and recorded this message 2 when Alice contacted the KDC using the old key, that is, J_{Alice}.

Later on, Alice changed her key to K_{Alice}, but Trudy can still use the old key and the recorded message to impersonate herself as Alice to Bob because the KDC had sent K_{AB} using the old key J_{Alice}, which Trudy has stolen. Trudy can decrypt this part of the communication to obtain the key K_{AB} and then send this message: 'I am Alice', ticket = K Bob\{use K_{AB} for Alice\}. At this point, Trudy and Bob have the secret K_{AB} shared among

themselves. Then, Trudy can communicate with Bob, pretending to be Alice. So, this is one vulnerability that this protocol has.

We'll discuss improved protocols which overcome this vulnerability. So, in summary, we introduced the concept of a KDC, that is, a key distribution center, and we introduced some protocols for KDC-mediated authentication, but as we have seen, these protocols have some vulnerabilities. So, in the next lecture, we will discuss improved protocols for KDC-mediated authentication. Thank you.