Network Security

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

Week - 04
Lecture - 21
Authentication: Part 2

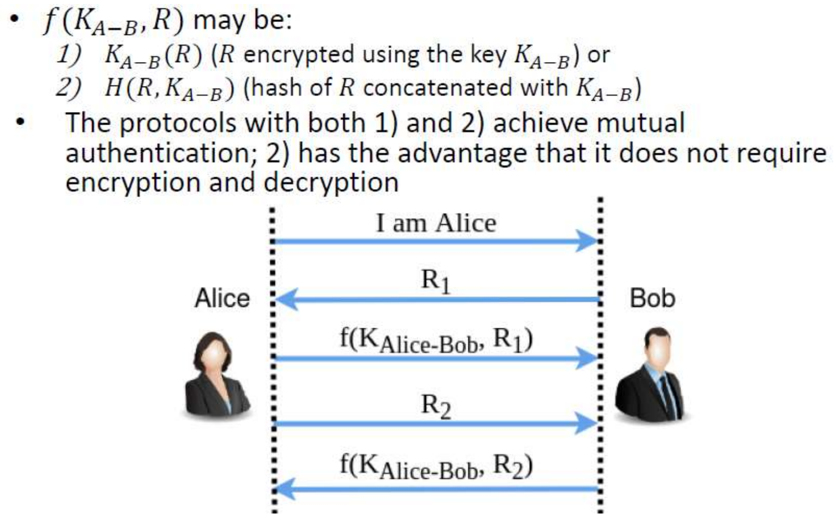
Hello, in this lecture, we'll continue our discussion of authentication. So, recall that ap4.0, the protocol which is shown in this figure, this is vulnerable to the server database reading attack, which we discussed in the previous lecture. It has another weakness, that is, Bob authenticates Alice but Alice does not authenticate Bob. That is, Bob is able to verify whether it is indeed Alice on the other side, but Alice is not able to verify whether it's indeed Bob on the other side. So, if there's an intruder on the path between Alice and Bob, then the intruder can intercept the 'I am Alice' message, and the intruder can send any number R to Alice, and ignore her response $K_{A\text{-}B}(R)$.

Alice thinks she's communicating with Bob. So, this is a one-way authentication protocol. Alice authenticates herself to Bob, but Bob doesn't authenticate himself to Alice. Now, suppose Alice and Bob have a shared symmetric key, $K_{A\text{-}B}$. How can we modify this protocol so that it achieves mutual authentication that is Bob authenticates Alice and vice versa?

So, both parties must authenticate themselves. So, we require mutual authentication. So, one way is this: after Alice sends the message 'I am Alice' to Bob, we simply do an authentication exchange in each direction. So, we just repeat the protocol used for one-way authentication in each direction to achieve mutual authentication. This shows the protocol.

Alice sends 'I am Alice' to initiate the authentication. Bob sends a challenge, R_1, to Alice. Then, Alice sends a response to the challenge. Depending on which protocol out of the ones that we discussed in the previous lecture we are using, this $f(K_{A\text{-}B}, R)$ may be $K_{A\text{-}B}(R)$, that is R encrypted using the key $K_{A\text{-}B}$, or $H(R, K_{A\text{-}B})$, that is the hash of R concatenated with $K_{A\text{-}B}$. In either case, it is a response to the challenge, which demonstrates that the responder has knowledge of the shared secret key, $K_{A\text{-}B}$.

So, f($K_{A\text{-}B}$, R) is either $K_{A\text{-}B}$(R) or H(R, $K_{A\text{-}B}$). So, in either case, it demonstrates that Alice has knowledge of $K_{A\text{-}B}$. So, Bob sends the challenge R_1 to Alice, and Alice responds with her response f($K_{A\text{-}B}$, R_1). So, this proves that it is indeed Alice on this side. Now, Alice sends her challenge to Bob, that is R_2, and Bob responds with his response to the challenge, f($K_{A\text{-}B}$, R_2).



- $f(K_{A-B}, R)$ may be:
  1) $K_{A-B}(R)$ (R encrypted using the key K_{A-B}) or
  2) $H(R, K_{A-B})$ (hash of R concatenated with K_{A-B})
- The protocols with both 1) and 2) achieve mutual authentication; 2) has the advantage that it does not require encryption and decryption

Alice
Bob
I am Alice
R_1
f($K_{Alice\text{-}Bob}$, R_1)
R_2
f($K_{Alice\text{-}Bob}$, R_2)

So, regardless of how we select f($K_{A\text{-}B}$, R) from among these alternatives, the protocols achieve mutual authentication. This second version, where f($K_{A\text{-}B}$, R) is H(R, $K_{A\text{-}B}$), has the advantage that it does not require encryption and decryption, so it is computationally more efficient than the first version, where f($K_{A\text{-}B}$, R) is H(R, $K_{A\text{-}B}$). This is one technique to achieve mutual authentication. We just repeat the one-way authentication protocol in each direction. Now, this protocol that we just discussed is inefficient because it uses five messages. So, we can see that there are five messages used in this protocol.

So, it has a lot of communication overhead. A modified version that uses only three messages is shown in this figure. Alice sends 'I am Alice' along with her own challenge, that is, R_2 to Bob. So, R_1 and R_2 are nonces. Bob responds to Alice's challenge, that is, f($K_{A\text{-}B}$, R_2), along with his own challenge, that is R_1, to Alice, and Alice then responds to the challenge of Bob, that is, f($K_{A\text{-}B}$, R_1).

So, is this modified version secure? At first, it looks like it is similar to the protocol in the previous slide. Alice sends a challenge, Bob responds to it, and Bob sends a challenge, and Alice responds to it. But this protocol is not secure. It is vulnerable to an attack known as a 'reflection attack'.

We'll discuss the reflection attack on the next slide. Suppose it is possible for a client to open multiple sessions to a server. This is often possible. For example, we can open different web sessions to the same server. Then in Intruder, Trudy can fraudulently authenticate herself as Alice to Bob as follows.

First, Trudy initiates a mutual authentication as in the above three message protocol. So, this is the mutual authentication session initiated by Trudy pretending to be Alice. Trudy sends 'I am Alice' along with Trudy's challenge R_2 to Bob. Bob computes $f(K_{A-B}, R_2)$, that's a response to Trudy's challenge, and sends it along with his own challenge R_1 to Trudy. Now, Trudy does not have the secret K_{A-B} to respond to this challenge R_1, so Trudy is unable to compute $f(K_{A-B}, R_1)$.

But then, when Trudy receives R1 and $f(K_{A-B}, R_2)$ from Bob, Trudy initiates a new session by sending 'I am Alice, R_1', where this R_1 is the same as this quantity, which is Bob's challenge. Now, Trudy sends it as her own challenge to Bob. So, Trudy sends "I am Alice, R_1" to Bob and then Bob computes $f(K_{A-B}, R_1)$ and then sends it to Trudy. Trudy then abandons the second session and goes back to use this $f(K_{A-B}, R_1)$, which is sent by Bob, to complete the first session. So, Trudy sends $f(K_{A-B}, R_1)$ to Bob, and hence, that's a correct response to Bob's challenge.

So, Trudy's authentication as Alice is successful. This attack is known as the reflection attack. Using this, an intruder can authenticate themselves as Alice to Bob. So, this is known as a reflection attack because when a challenge is sent by Bob to Alice, then if there's an intruder, Trudy, in place of Alice, then they can just reflect that challenge. So, here is the reflected challenge.

So, Trudy just reflects that challenge, the same challenge R_1 to Bob. And Bob himself provides the response to the challenge. Hence, this is known as the reflection attack. Hence, this protocol that we discussed is vulnerable to the reflection attack. So, what are some defenses against the reflection attack?

One defense is we can use different keys to authenticate Alice to Bob and to authenticate Bob to Alice. So far we have assumed that the same key K_{A-B} is used to authenticate Alice to Bob and to authenticate Bob to Alice, but we can use say different keys, maybe, say K_A, to authenticate Alice to Bob and K_B to authenticate Bob to Alice. So, we can use different keys to authenticate Alice to Bob and Bob to Alice. So, if this is done, then the reflection attack does not work. So, the previous protocol modified with the fact that different keys are used to authenticate in the two directions, so that protocol works correctly.

Another defense is when Alice sends "I am Alice, R_2" to Bob, Bob responds with R_1, f(K_{A-B}, R_2, Bob), where this ‘Bob’ is Bob’s identifier, and Alice responds with f(K_{A-B}, R_1, Alice), where ‘Alice’ is Alice’s identifier. This mechanism also works. So, in this case, Trudy cannot reflect Bob’s challenge because of the presence of the identifier in this quantity that is used to compute the function f. So, this defense also works correctly. So, using one of these defenses, we can defend against a reflection attack and we can go back to this protocol which uses only three messages.

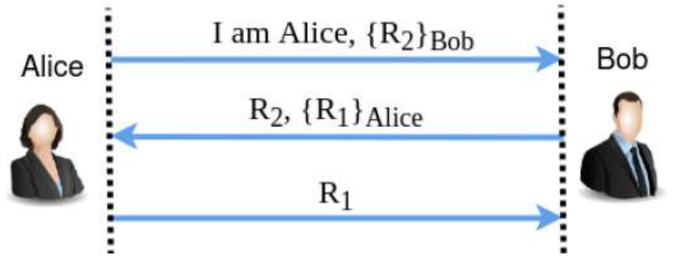
2) When Alice sends “I’m Alice, R_2” to Bob, Bob responds with $R_1, f(K_{Alice-Bob}, R_2, \text{Bob})$; then Alice responds with $f(K_{Alice-Bob}, R_1, \text{Alice})$

Yeah, so this protocol which uses only three messages can be used along with the two defenses that we discussed, one of the two defenses that we discussed. And either of those defenses will defend it against a reflection attack. Now, let’s discuss mutual authentication using public keys. Suppose Alice and Bob know each other’s public keys, how can we perform mutual authentication? So, we can perform it as in this figure.

Alice sends ‘I am Alice’ along with $\{R_2\}_{Bob}$ to Bob. So, Alice selects a nonce R_2 and encrypts it using Bob’s public key, and the result is $\{R_2\}_{Bob}$, which Alice sends to Bob. Bob decrypts $\{R_2\}_{Bob}$ using his private key to get R_2 and sends it to Alice. Bob also selects his own challenge R_1 and encrypts it using Alice’s public key. The result is $\{R_1\}_{Alice}$, and Bob combines R_2 and $\{R_1\}_{Alice}$ and sends them to Alice.

- Are these protocols vulnerable to reflection attack?

❑No

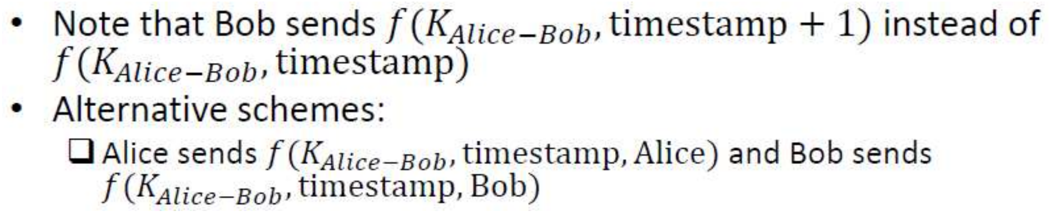


Alice then decrypts $\{R_1\}_{Alice}$ using her private key and sends the result R_1 to Bob. So, this protocol successfully achieves mutual authentication. Alternatively, Alice sends ‘I am Alice’ along with a nonce R_2 to Bob. Then, Bob responds with R_2 signed with his private key along with his own challenge R_1, and Alice responds with R_1 signed with her private key. So, this protocol also works correctly. So, are these protocols vulnerable to the reflection attack?

No, they are not vulnerable because Alice and Bob have different public key, private key pairs. So, because of this lack of symmetry, the reflection attack does not work. So, when Bob sends a challenge to Alice, Alice cannot use the response of Bob to her own challenge to respond to Bob's challenge because the secrets used in the two directions are different. So, Alice's private key is different from Bob's private key, and hence, this protocol is not vulnerable to the reflection attack. So, both of these protocols are secure against the reflection attack.

Now, notice that the above protocols use three messages for mutual authentication. Suppose Alice and Bob have a shared symmetric key K_{A-B} and approximately synchronized clocks. Can we modify the above protocol based on a shared symmetric key so that it uses only two messages instead of three? So, can we use the fact that Alice and Bob have approximately synchronized clocks to reduce the overhead from three messages to only two messages? So, mutual authentication can be performed as shown in this figure.

Alice sends 'I am Alice' along with f(K_{A-B}, timestamp) to Bob. Now, Bob responds with f(K_{A-B}, timestamp+1). So, this f(K_{A-B}, timestamp) demonstrates that Alice knows the secret K_{A-B}, and this message f(K_{A-B}, timestamp+1) demonstrates that Bob knows the secret K_{A-B}. So, notice that Bob sends f(K_{A-B}, timestamp+1) instead of f(K_{A-B}, timestamp) because f(K_{A-B}, timestamp) has already been sent over the communication channel, so anyone who sniffs the channel knows this f(K_{A-B}, timestamp), so it doesn't make sense for Bob to just send the same quantity. So, hence, Bob increments the timestamp and sends f(K_{A-B}, timestamp+1), which is proof that Bob knows K_{A-B}.



- Note that Bob sends $f(K_{Alice-Bob}, \text{timestamp} + 1)$ instead of $f(K_{Alice-Bob}, \text{timestamp})$
- Alternative schemes:
  - ❑ Alice sends $f(K_{Alice-Bob}, \text{timestamp, Alice})$ and Bob sends $f(K_{Alice-Bob}, \text{timestamp, Bob})$

Again, here this function f may be K_{A-B}(timestamp) or it may be H(timestamp, K_{A-B}). Here are alternative schemes. Alice sends f(K_{A-B}, timestamp, Alice), where 'Alice' is Alice's identifier, and Bob sends f(K_{A-B}, timestamp, Bob). So, here in this protocol, we use the same timestamp for authentication in the two directions. But we use the identifier of the sender to break the symmetry in the two directions.

So, the messages sent by Alice and Bob become different, and hence each one can demonstrate that they know the secret $K_{A\text{-}B}$. Alice and Bob use the same timestamp but different keys. That's another scheme using which they can mutually authenticate themselves. So, actions similar to the ones that we discussed for one-way authentication using timestamps must be performed. For example, Bob must check that the timestamp used by Alice is within an acceptable clock skew, and Bob must remember the timestamps used by Alice in the past until they expire.

So, we discussed one-way authentication when Alice and Bob have approximately synchronized clocks. So, the principles that we discussed in that context are also applicable here. Now, suppose mutual authentication is performed as in this figure shown here. Alice sends "I am Alice, f($K_{A\text{-}B}$, timestamp)" to Bob, and then Bob sends f($K_{A\text{-}B}$, timestamp+1). One attack on this is that an intruder, say Trudy, can later on use this value sent by Bob, that is, f($K_{A\text{-}B}$, timestamp+1), to authenticate themselves as Alice to Bob or to another server that shares the same key $K_{A\text{-}B}$ with Alice.

In this authentication, Alice sent the message f($K_{A\text{-}B}$, timestamp) to Bob. During this authentication, Trudy eavesdropped on the conversation and intercepted f($K_{A\text{-}B}$, timestamp+1). Later on, Trudy sends the message "I am Alice, f($K_{A\text{-}B}$, timestamp+1)" to Bob. So, that timestamp is different; the timestamp used here is $K_{A\text{-}B}$, timestamp+1, which is different from the original timestamp. So, the message is different. If Bob checks whether the timestamp used is the same as in a previous authentication, then Bob will find that it's not the same timestamp; it's a different timestamp.

So, Bob may accept the authentication. So, this is one attack that can be performed. Or Trudy can intercept this message and use this value to authenticate herself to another server that shares the same key with Alice. So, a defense against the above attack is the following. Alice sends f($K_{A\text{-}B}$, timestamp, Bob), where 'Bob' is Bob's identifier, and Bob sends f($K_{A\text{-}B}$, timestamp, Alice), where 'Alice' is Alice's identifier.

Also, Alice does not use the same timestamp again to authenticate to any server. Since the server's identifier has to be appended to the message before finding the function f, this message will become different for different servers. So, even if Trudy records the messages being exchanged between Alice and Bob in this conversation, those messages won't enable Trudy to authenticate to another server because the server's identifier would be different. But notice that this f($K_{A\text{-}B}$, timestamp, Alice) has Alice's identifier. So, Trudy should not be able to use this to pretend as Bob and to pretend as some other server, say Bob'.

And then, when Alice tries to authenticate herself to the other server, Bob’, Trudy can send this message, which was recorded from this exchange, to authenticate as Bob’. So, to prevent that, Alice does not use the same timestamp again to authenticate to any server. This concludes our discussion of mutual authentication. Thank you.