

# COMP 4307/5327 Computer Security

## Tutorial 2

1. What is the apparent purpose of this protocol, what assumptions appear to have been made and to what attacks is it vulnerable?

A → B : ID<sub>A</sub>

B → A : ID<sub>B</sub>

A → B : K<sub>AB</sub>{ID<sub>A</sub>, ID<sub>B</sub>, timestamp}

It is a one-way authentication protocol, A is proving its identity to B

The key assumption is that A and B share a key, there is also the assumption that A and B can do the cryptography. The second assumption is so basic that it is rarely stated.

It is vulnerable to the replay attack. The attacker can simply record the third message and use it to pretend to be A (if they use it quickly enough, before the acceptable window for use of the timestamp expires).

2. The following protocol is designed to be used for mutual authentication, with symmetric key technology. However, it is vulnerable to the reflection attack. Alter this protocol so that it will no longer be vulnerable to this attack. Do **not** add extra steps to or delete existing steps from the protocol. You may also **not** delete any of the information sent in the existing steps. Assume that the two participants already share the secret K<sub>AB</sub>.

A → B : ID<sub>A</sub> , R<sub>1</sub>

B → A : K<sub>AB</sub>{R<sub>1</sub>},R<sub>2</sub>

A → B : K<sub>AB</sub>{R<sub>2</sub>}

Show them how it is vulnerable to the reflection attack. There are a number of ways to fix this. To be cruel, I forbade the most obvious, which is to insist on the initiator authenticating first.

The next most obvious way to fix it is to stop a message from B being used as a message to B. There are a number of ways to do this, here's two

Include the sender and receiver's names (in fixed order) in the encryption

A → B : ID<sub>A</sub> , R<sub>1</sub>

B → A : K<sub>AB</sub>{B,A,R<sub>1</sub>},R<sub>2</sub>

A → B : K<sub>AB</sub>{A,B,R<sub>2</sub>}

Or use nonces or sequence numbers

A → B : ID<sub>A</sub> , R<sub>1</sub>, N<sub>1</sub>

B → A : K<sub>AB</sub>{R<sub>1</sub>, N<sub>1</sub>, N<sub>2</sub>},R<sub>2</sub>, N<sub>2</sub>

A → B : K<sub>AB</sub>{R<sub>2</sub>, N<sub>2</sub>}

Basically, anything to distinguish a step 2 message from a step 3 message. Note that just putting in timestamps won't work, as the message could be reflected back before the acceptable window for the timestamp expires.

3. For **each** message in the following protocol state what the recipient of that message knows on receipt and analysis of the message. After which message is A assured of B's identity? After which message is B assured of A's identity?

$A \rightarrow KDC : ID_A, ID_B, N_{A1}$   
 $KDC \rightarrow A : K_A\{N_{A1}, ID_B, K_B\{ID_A, N_{A1}, N_{KDC1}, ts_1\}\}$   
 $A \rightarrow B : K_B\{ID_A, N_{A1}, N_{KDC1}, ts_1\}$   
 $B \rightarrow A : K_B\{ID_A, N_{A1}, N_{KDC1}, N_{B1}, ts_2\}$   
 $A \rightarrow KDC : ID_A, K_A\{N_{A2}, ID_B, K_B\{ID_A, N_{A1}, N_{KDC1}, N_{B1}, ts_2\}\}$   
 $KDC \rightarrow A : K_A\{N_{A2}, ID_B, K_{AB}, K_B\{K_{AB}, ID_A, N_{B1}\}\}$   
 $A \rightarrow B : K_B\{K_{AB}, ID_A, N_{B1}\}$

**Message 1 :** KDC knows that somebody claiming to be A wants to talk to B

**Message 2:** A knows that the reply has come from the KDC (as only it could have encrypted  $N_{A1}$  under  $N_{A1}$ ) and that third part of the message is to be passed on to B as the KDC wishes to know whether B wants to talk to A.

**Message 3:** B knows that A wishes to talk to it. It can tell this from the structure of the message, which can only have been put together by the KDC. It knows that the request is current, from the value of the timestamp. It does **not** know that this message has come from A, as the message could have been intercepted and replayed

**Message 4:** A has a reply, but does not know whether this reply is actually from B

**Message 5:** The KDC knows that B wants to talk to A, as only B could have taken the nonces from A and the KDC and formed them up into the inner part of the message. The KDC further knows that the replay from B is fresh, by the value of the second timestamp. The KDC does not know that this message can directly from A or was replayed, but that really does not matter

**Message 6:** A knows the shared key to use with B. A also knows that it is talking to B, as it trusts the KDC to only send message 6 if the contents of message 4 are ok (ie could only have come from B)

**Message 7:** B knows the shared key to use with A and that it must be communicating with A as only could have recovered the contents of message 7 from message 6

**Note:** A and B are both willing to live with the chance that messages 6 and 7 could have been intercepted and then sent from somewhere other than the supposed originator as only A and B could recover the key  $K_{AB}$  from these messages, so an attacker would learn nothing

4. Consider the following outline of a protocol

A customer authorises a transfer from his bank account to another bank account by sending the following information to bank

Customer's account details, destination account details, time, amount, etc

This information is encrypted under a symmetric key generated by the customer. The symmetric key is sent along with all the other information, but it is encrypted under the public key of the bank. The customer signs the message with their private key. Assume that the customer and bank securely know each other's public keys. Assume that account details are name of account holder and account number.

Can you think of a way to subvert this process?

Account details are reasonably easy to get hold of – ATM statements, intercepting full statements in the mail, even asking for account numbers when registering on a pay per view website.

Speaking of such a website, all you then have to do is ask someone to sign something when they want to access the website (to prove their identity). All you have to do is ask them to sign the above information and you've done it.