Homework7: Security - Solutions

COM-208: Computer Networks

With respect to a message m, we define the following security properties:

- **Confidentiality** holds iff, additionally to the sender of m, only the legitimate recipient of m gets to learn the content of m.
- **Authenticity** holds iff the legitimate recipient of m can verify that no third party assumes the identity of the legitimate sender of m.
- **Integrity** holds iff the legitimate recipient of m can verify that no third party has modified m.

Providing authenticity

End-systems Alice and Bob share a secret key K and use the following protocol such that Alice authenticates herself to Bob:

S1. Alice tells Bob that she has something to say.

S2. Bob sends Alice a nonce R.

S3. Alice sends Bob $m, K\{R, m\}$, i.e., her message plus the nonce and message encrypted with their shared secret key.

Questions:

1. Consider an adversary, Persa, who is sitting on the communication channel between Alice and Bob. Does this protocol prevent impersonation and enable key reuse?
2. Consider an adversary, Trudy, who is *not* on the communication channel between Alice and Bob (she is just another end-system somewhere on the Internet). And suppose that while Alice is authenticating herself to Bob, Bob uses the same protocol to authenticate himself to Alice. Can you think how Trudy can take advantage of this situation and impersonate Alice to Bob?

Hint: Consider that, while Alice and Bob authenticate themselves to each other, their messages can be arbitrarily interleaved. Also, consider that, when Alice authenticates herself to Bob, it is Bob that chooses the nonce, and vice versa.

1. Yes, the protocol prevents impersonation, because, except for Bob, only Alice knows K, hence only Alice could have generated $K\{R, m\}$. It also enables key reuse, because Persa cannot infer any information about K from $K\{R, m\}$.
2. Trudy can take advantage as follows:
   - S1. Trudy pretends to be Alice and tells Bob that she has something to say.
   - S2. Bob sends Trudy a nonce R (thinking that he is sending it to Alice).
   - S3. At the same time, Bob tells Trudy that he has something to say (still thinking that he is talking to Alice).
   - S4. Trudy sends Bob the same nonce R.
   - S5. Bob sends Trudy $m, K\{R, m\}$.
   - S6. Trudy sends Bob $m, K\{R, m\}$.

   In summary, by sending Bob the same nonce that he sent her, Trudy successfully impersonated Alice, i.e., made Bob believe that Alice sent him message m. Of course, Trudy cannot make Bob believe that Alice sent him *any* message, she can only make Bob believe that Alice sent him the same message m that he sent.

Routing attack

A router's link-state message includes a list of its directly connected neighbors and the direct costs to these neighbors. Once a router receives link-state messages from all of the other routers, it can create a complete map of the network, run its least-cost algorithm, and configure its forwarding table. One relatively easy attack on the routing algorithm is for the attacker, after intercepting the routers' messages, to arbitrarily modify them so as to distribute bogus link-state messages with incorrect link-state information. How can this be prevented?

Each router can digitally sign the link-state messages it produces. As long as the routers know the public key of a trusted certificate authority, each router can verify the digital signature on every link-state message it receives.

Providing all three properties

A and B are end-systems. M is a malicious adversary sitting on the communication channel between them. A wants to send message m to B. M wants to read and/or modify message m. A knows this and tries to send message m to B in a way that guarantees confidentiality, authenticity, and data integrity.

For each of the following cases, identify if confidentiality and/or authenticity and/or data integrity hold, and justify your answer.

1. A sends: $K_s\{m\}$
2. A sends: $m,\ K_s\{m\}$
3. A sends: $K_A^+\{m\}$
4. A sends: $m,\ K_B^+\{m\}$
5. A sends: $K_A^-\{K_B^+\{m\}\}$

where:

- K_s is a symmetric key, shared only between A and B
- $K_A^+,\ K_A^-$ is A's public and private key, respectively
- $K_B^+,\ K_B^-$ is B's public and private key, respectively

1. Confidentiality holds, because, except for A, only B knows K_s, hence only B can decrypt $K_s\{m\}$. Authenticity and data integrity do not hold, because M can arbitrarily modify the ciphertext, even if she does not know the plaintext, and B has no way to detect it.
2. Confidentiality does not hold because A sends message m in the clear. Authenticity and data integrity hold, because, except for B, only A knows K_s, hence only A could have generated $K_s\{m\}$ which, once decrypted, matches the plaintext m.
3. None of the three properties holds. B does not know K_A^-, hence cannot decrypt $K_A^+\{m\}$. As a result, B can neither read m nor verify that A generated it.
4. None of the three properties holds. Confidentiality does not hold because A sends m in the clear. Authenticity and data integrity do not hold either, because M (or anyone who knows K_B^+) could have generated $K_B^+\{m\}$.

5. Confidentiality holds, because only B knows K_B^-, hence only B can decrypt $K_B^+\{m\}$. Authenticity and data integrity do not hold: B can use K_A^+ to decrypt $K_A^-\{K_B^+\{m\}\}$ and obtain $K_B^+\{m\}$; then use K_B^- to decrypt $K_B^+\{m\}$ and obtain m. However, B does not know m in advance, so she has no way of verifying that the message was generated by A and was not tampered with.

With security fixes

End-systems A and B want to secure their communication against an adversary sitting on the communication channel between them. In each of the following scenarios:

i Identify an existing security problem or weakness and, if applicable, describe an attack that exploits it.

ii Provide a solution that fixes the weakness (i.e. what should A send instead). Make sure to provide enough detail for your solution to be understandable. For example, if you say "A should use a MAC for authentication", but you do not explain how the MAC should be computed, then your answer is not complete.

Scenarios:

1. A wants to send one message m to B, ensuring confidentiality and authenticity. For this, A sends: $H\,(K, m)$.
2. A wants to send one message m to B, ensuring confidentiality and authenticity. For this, A sends: $K\,\{m\}$.
3. A wants to send one message m to B, ensuring confidentiality and authenticity. A knows B's true public key K_B^+. A sends: $K_B^+\left\{m\ ,\ K_B^+\,\{H(m)\}\right\}$.
4. A is sending a number of messages to B. Whenever B receives a message m, B should be able to verify that A indeed sent a message with the same content as m at least once. For this, A appends $H(K)$ to each message it sends.
5. A wants to send one message m to B, ensuring authenticity and data integrity. For this, A cuts m into two pieces, m_1 and m_2, sends first $m_1, H(K)$, then $m_2, H(K)$.
6. A and B are friends that want to have a sensitive online conversation, ensuring confidentiality, authenticity, and data integrity (A receives exactly the sequence of messages sent by B and vice versa). For this, they use SSL as described in class: B sends a nonce and a certificate with its public key to A, A sends a nonce and a master key to B (encrypted with B's public key), and from the master key

and the nonces, they both derive other keys that they use for confidentiality and authenticity.

Hint: In the example we saw in class, B was an online store. In this question, A and B are friends having a sensitive conversation.

where:

- H is a cryptographic hash function that is known to everyone.
- K is a symmetric key, shared only between A and B.
- K_B^+ is B's public key.

1.
   - Neither confidentiality nor authenticity hold. Confidentiality does not hold as B cannot read m, because she cannot invert the hash function H. Authenticity does not hold either because B does not know m and cannot invert H so as to verify that the sender knows the shared secret K.
   - A solution that guarantees both confidentiality and authenticity is: A sends $K\{m\ ,\ H\,(K, m)\}$.
2.
   - Authenticity does not hold: A man-in-the-middle can change the ciphertext $K\,\{m\}$. B will be able to decrypt the message with the shared key K, but the decrypted message will be different from the one that was initially sent by A, and B has no way to detect it.
   - A solution that guarantees both confidentiality and authenticity is: A sends $K\{m\ ,\ H\,(K, m)\}$.
3.
   - Authenticity does not hold: Since both the public key K_B^+ and the hash function H are publicly available, anyone can generate the message $K_B^+\left\{m\ ,\ K_B^+\,\{H(m)\}\right\}$.
   - A solution that guarantees both confidentiality and authenticity is: A sends $K_B^+\left\{m\ ,\ K_A^-\,\{H(m)\}\right\}$.
4.
   - The property described in the formulation does not hold: A man-in-the-middle can change one or more messages sent by A. B has no way to be sure that no one tampered with the messages by only computing $H(K)$, as the latter has nothing to do with the message itself.
   - A solution involves the use of a MAC: A should append $H(m_i, K)$ to every message m_i she sends.

5.
- Authenticity and data integrity of the whole message cannot be guaranteed: A man-in-the-middle can either change m_1 and/or m_2 (as the accompanying $H(K)$ has nothing to do with the message itself), or reorder the two messages.
- A solution that guarantees both authenticity and data integrity is to use a MAC and sequence numbers: A sends first $m_1, 1, H(K, m_1, 1)$, then $m_1, 2, H(K, m_1, 2)$.

6.
- The problem here arises from the fact that A never sends a certificate with its public key to B. Therefore, B cannot authenticate A (only A can authenticate B). A man-in-the-middle can intercept B's message to A, pretend she is A, and carry out the entire conversation with B, pretending that she is A.
- A solution is for A to send its public-key certificate to B along with the nonce and the master key.

The role of sequence numbers

End-systems A and B secure their communication with the following protocol:

S1: A sends a "hello" message with a nonce n_A and a certificate containing her public key (K_A^+)

S2: B responds with a nonce n_B and a certificate containing her public key (K_B^+)

S3: After this exchange, to communicate a message m_i, A sends: $K_B^+ \left\{ m_i, K_A^- \left\{ H\left(n_B, m_i\right) \right\} \right\}$

S4: Similarly, to communicate a message m_j, B sends: $K_A^+ \left\{ m_j, K_B^- \left\{ H\left(n_A, m_j\right) \right\} \right\}$,

where K_A^-, K_B^- are A's and B's private keys and H is a globally known cryptographic hash function.

Questions:

1. Does this protocol guarantee confidentiality?
2. Assume a man-in-the-middle, who records all the messages sent by A, and, when the communication between A and B ends, she resends them to B, trying to impersonate A. Will her attack be successful (or not) and why?
3. Is this protocol vulnerable to any attack(s) other than the ones described above? If yes, briefly describe the attack(s) and provide the changes that make the protocol completely secure. If necessary, you can add new steps in the protocol, but do not modify the existing ones.

1. Yes, the protocol guarantees confidentiality, because of two reasons:
   a) All public and private keys are used correctly for the encryption of the messages: i.e. the public keys of the recipients of the messages are used for encryptions and the private keys of the senders are used for providing a digital signature (e.g. $K_B^- \left\{ H\left(n_A, m_j\right) \right\}$) of the message.
   b) The public keys of both communicating parties have been exchanged with certificates, which guarantees that the true and correct public keys of one are available to the other party.
2. No, the attack cannot be successful because the nonces are correctly used inside the digital signatures of each message.

3. Yes, the protocol is vulnerable to reordering and deletion attacks, because it uses no sequence numbers: A man-in-the-middle can reorder $K_B^+\left\{m_i, K_A^-\left\{H\left(n_B, m_i\right)\right\}\right\}$ or $K_A^+\left\{m_j, K_B^-\left\{H\left(n_A, m_j\right)\right\}\right\}$; or she can delete one of the messages.

   One solution is to have both parties exchange a digital signature of all messages at the end of their communication. I.e., to the above protocol, we add two more steps:

   S1: A sends $K_B^+\left\{K_A^-\left\{H\left(n_B, m_1, m_2, \ldots\right)\right\}\right\}$ to B

   S2: B sends $K_A^+\left\{K_B^-\left\{H\left(n_A, m_1, m_2, \ldots\right)\right\}\right\}$ to A

Out-of-order delivery atop TCP

In class, we said that SSL relies on sequence numbers to prevent an attacker from changing the order of messages (it was the scenario where Alice communicates with an online store, and we want to prevent an attacker sitting on the communication channel between them from swapping Alice's requests to the store). Why do we need this technique? Won't TCP ensure that the online store receives Alice's messages in the correct order?

SSL aims to protect the communication between Alice and the online store against an adversary who is sitting on the communication channel between Alice and the online store (hence can see and potentially change their exchanged packets). TCP cannot guarantee in-order delivery in the face of such an adversary: The adversary can simply swap two TCP segments and change their sequence numbers such that the TCP receiver thinks that the messages are received in order.