

IT2760/CS3760: Coursework 4

Solutions

1. The following three access control mechanisms were discussed in the course notes:

- access control matrices,
- capabilities,
- access control lists.

For each access control mechanism, describe the complexity of:

- determining authorised access during execution;
- adding accesses for a new subject;
- deleting all accesses for a particular subject;
- determining all subjects which have access to a particular object;
- creating a new object to which all subjects have access by default.

[15 marks]

Access control matrix:

- *determining authorised access during execution: this is simple (just look up the appropriate entry in the table);*
- *adding accesses for a new subject: not so simple – a new row in the matrix will need to be created and completed;*
- *deleting all accesses for a subject: relatively simple, since it simply means deleting a row in the access control matrix;*
- *determining all subjects which have access to a particular object: relatively simple, since it involves checking the relevant column of the matrix;*
- *ease of creating a new object to which all subjects have access by default: relatively simple, since it simply means creating a column in the access control matrix with all entries 'positive'.*

Capabilities (i.e. a list for each subject indicating the objects that can be accessed, and that corresponds to a row of an access control matrix):

- *determining authorised access during execution: simple;*
- *adding access for a new subject: simple;*
- *deleting access by a subject: simple;*
- *determining all subjects which have access to a particular object: not simple (possibly infeasible), since all capabilities held by every subject will need to be checked;*

- *creating a new object to which all subjects have access by default: not so simple, since every subject will need to be given a new capability.*

Access control list (i.e. a list for each object that indicates the subjects that have access to that object, and that corresponds to a column of an access control matrix):

- *determining authorised access during execution: reasonable, as long as access to the per-object access control list is straightforward;*
- *adding access for a new subject: not simple, since every object's access control list will need to be modified (at least all those objects to which the new subject is to be granted access);*
- *deleting access by a subject: not simple, since every object's access control list will need to be checked, and, if necessary, modified;*
- *determining all subjects which have access to a particular object: simple – just check the ACL;*
- *creating a new object to which all subjects have access by default: simple.*

2. Give an advantage and a disadvantage of each of the following three types of non-repeating value, as used to counter replay attacks on authentication protocols (i.e. to provide 'freshness checking'):

- random numbers (unpredictable nonces),
- sequence numbers (logical timestamps), and
- clock-based timestamps.

[6 marks]

Nonces: *They have the advantage that they require the minimum of stored 'state information', and they also provide linking between the various messages of a protocol. They also do not require special clock hardware to be provided. They have the disadvantage of typically requiring an additional message to be sent.*

Sequence numbers: *They have the advantage of requiring one less message than nonce-based protocols, and they also do not require special clock hardware to be provided. However they have the disadvantage of requiring sequence numbers to be set up and maintained for every party which might need to be authenticated. They also cannot be used to detect 'forced delays' in messages, and do not provide automatic linking between the messages of a protocol.*

Clock-based time-stamps: *As with sequence numbers they have the advantage of requiring one less message than nonce-based protocols. They have the disadvantage that they require the maintenance of synchronised clocks, which not only means using a time synchronisation mechanism at regular intervals, but also requires additional hardware*

(namely a clock). They also do not provide automatic linking between the messages of a protocol.

3. Suppose parties A and B , who share a secret key K_{AB} , use the following unilateral authentication mechanism (i.e. a mechanism which enables B to authenticate A but not vice versa):

$$A \rightarrow B: e_{K_{AB}}(t_A)$$

where

- $A \rightarrow B: X$ means that A sends B the message X ;
- $e_K(X)$ denotes the symmetric encryption of data X using the secret key K , where the encryption algorithm provides confidentiality and integrity protection; and
- t_A is a timestamp generated by A .

Identify two possible types of attack (other than the use of a 'weak' encryption algorithm or loss of secrecy of the key).

[10 marks]

'Reflection attacks' are possible, where C can persuade A to generate a message which C can then use to impersonate B back to A . This would leave A thinking it was communicating with an authenticated B , when in fact A was actually communicating with C ! The vulnerability of the mechanism described is due to the symmetry of messages travelling in opposite directions.

This attack is called a reflection attack because of the way A 's message is mirrored back to itself. Although an individual user might detect this simultaneous use of exactly the same protocol messages (even though they are sent in opposite directions), in a networked environment where computers might be simultaneously communicating with many other entities, it would typically go unnoticed.

Replays of authentication messages are possible within the 'window of acceptance' of B , unless B retains a log of recently received messages.

If a third party could manipulate the clocks of B , then B could be made to accept replays of 'old' authentication messages.

4. Suppose parties A and B use the following unilateral authentication mechanism:

$$A \rightarrow B: t_A || s_A(t_A)$$

where

- $s_A(X)$ denotes a digital signature computed by A (using A 's private signature key) on data X ; and
- t_A is a timestamp generated by A .

Suppose also that B possesses the public verification key of A (so that B can verify A 's signature). Identify two possible types of attack on this

protocol (other than the use of a ‘weak’ signature algorithm or loss of secrecy of A’s private key). Compare these with the possible attacks on the protocol in the previous question.

[10 marks]

The following types of attack are possible:

Suppose C wishes to masquerade as A to B. C first persuades A to authenticate itself to C using:

$$A \rightarrow C : t_A \parallel s_A(t_A).$$

C can then use this message to masquerade as A to B, by sending:

$$C \rightarrow B : t_A \parallel s_A(t_A).$$

After these exchanges are complete, A thinks it has authenticated itself to C, whereas in fact C has ensured it has really been authenticated by B. Whilst this may appear to be an unimportant attack, if additional data is included within the scope of the signature (perhaps a message from A to C), then C can convince B that this is a message from A to B, which could have serious consequences.

Replays of authentication messages are possible within the ‘window of acceptance’ of B, unless B retains a log of recently received messages.

If a third party could manipulate the clocks of B, then B could be made to accept replays of ‘old’ authentication messages.

The attacks on the timestamp element (i.e. the second and third attacks) are precisely the same as in the previous question. However the first type of attack reveals a significant difference. In this case, the origin of the message is clear (because of the signature); however the destination is completely unrestricted (because of the availability of A’s public key). In the previous case, the origin /destination pair are fixed (because of the use of a shared secret key), but can be switched in an undetectable way.

5. Describe the Kerberos protocol for client-server authentication. Include a high-level description of the messages in the protocol, and describe the nature of the pre-existing relationships needed to support it.

[15 marks]

Kerberos makes use of symmetric encryption, manipulation detection codes (MDCs), and two trusted third parties as follows:

An Authentication Server (AS) shares a long-term key with the client with whom mutual authentication is provided at login. The AS gives to the client a ticket granting ticket and short-term key.

A Ticket Granting Server (TGS) performs mutual authentication with the client based on the short-term key and ticket granting ticket. The TGS then provides the client with tickets for access to further servers (S) demanding authentication.

There are three stages to the protocol:

Stage 1 – Client-AS interaction:

A long term key derived from the user password, and shared between the client and AS, is used to encrypt a message from AS which contains a short-term key for the client to share with the TGS. The message also contains a ticket granting ticket for the client to pass on the TGS. This message is encrypted using a key shared by both AS and TGS, and includes the short-term key, the client's identity and other data.

Stage 2 – Client-TGS interaction:

The Client presents a request to the TGS for access to server S, along with the ticket granting ticket and a message authenticating itself to TGS.

TGS checks the ticket granting ticket and extracts the short term key which it uses to authenticate the client. If all is OK, TGS issues a session key and a ticket to the client. The ticket is a message encrypted using a key shared by both TGS and S that includes the session key.

Stage 3 – Client-Server interaction:

Client presents ticket along with message authenticating itself to S. S checks the ticket and extracts the session key, which it uses to authenticate client. If all is OK, S grants access to client. Optionally S sends the client a message authenticating itself to the client.

Access to the AS is infrequent, reducing exposure of the long-term key. Each access to AS allows many accesses to the TGS, over the period of validity of the ticket granting ticket. Each access to the TGS using the short-term key may allow many accesses to S. Each access to S uses the session key during its period of validity, reducing the exposure of the short-term key. Thus security depends on a hierarchy of keys in which the length of the duration of the validity of a key is in inverse proportion to its exposure.