Computer and Network Security

Dr. Chan Yeob Yeun

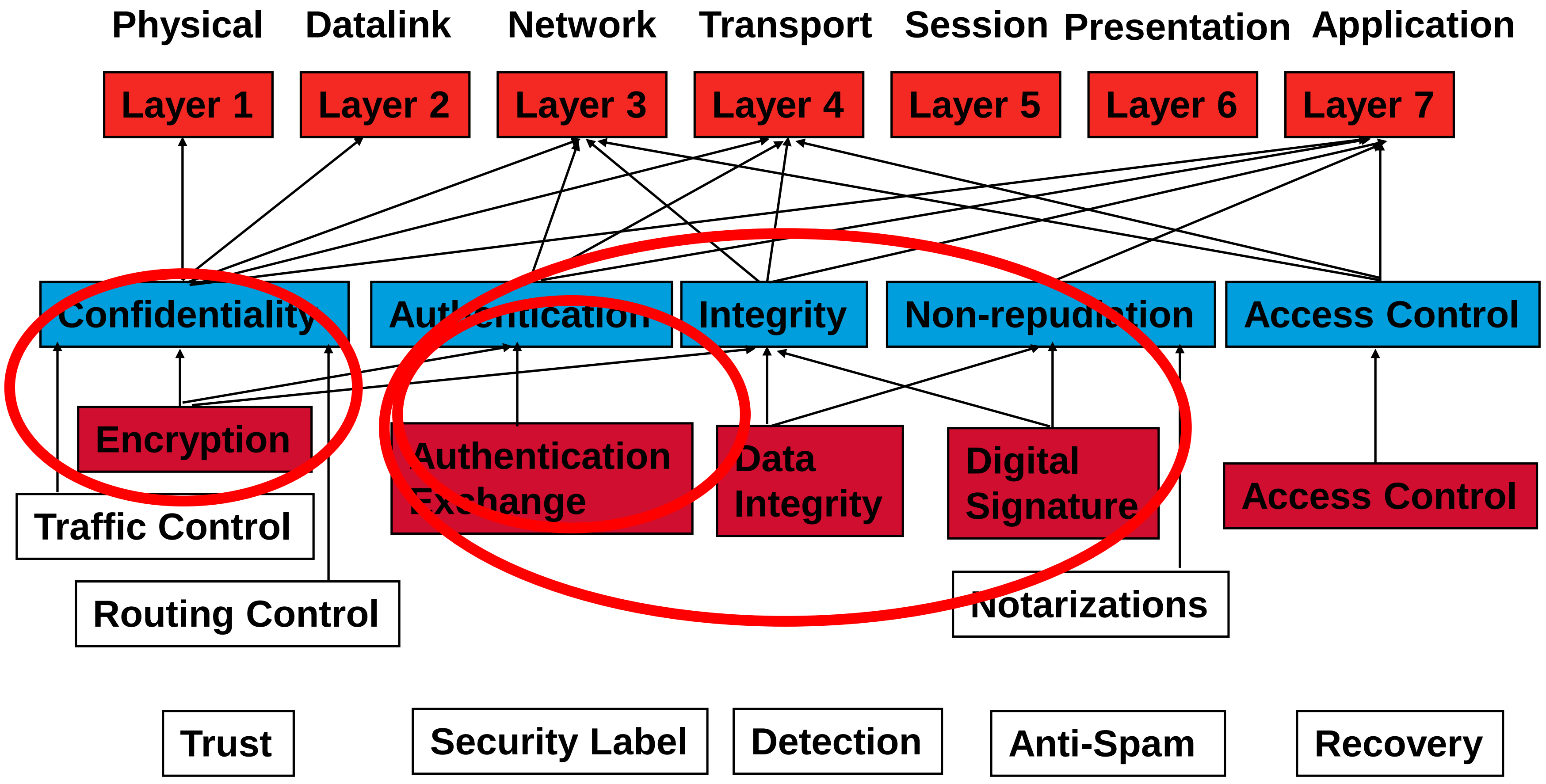
Week 10-11

جامعة خليفة
Khalifa University

Weekly Lecture Plan

Wk	Contents	Cmt	Wk	Contents	Cmt
1	**Introduction**		9	**Foundations of Network Security II**	
2	**Foundations of Computer Security**	**Tutorial Assig Plan**	10	**Network-Based Threats and Attacks**	
3	**Identification and Authentication I**		11	**Network Security Protocols I**	
4	**Identification and Authentication II**	**Quiz 1**	12	Network Security Protocols II	Quiz 3
5	**Access Control**		13	Firewalls	
6	**Modern Computer Attacks**		14	IDS / IPS	Assig Submit
7	**Malicious Code**	**Assig Confirm**	15	Revision and Presentation	
8	**Foundations of Network Security I**	**Quiz 2**	16	Exam	

What is Network Security ?



Security Protocols

Consider a variety of examples of security protocols.

Consider examples based on all three types of cryptographic mechanism and all three types of freshness mechanism.

Example 1 (time-stamp & encipherment)

This example can be found in clause 5.1.1 of ISO/IEC 9798-2

$$\begin{tikzcd}
A \arrow[r, "M_1 = Text2 \,\|\, eK_{AB}(T_A \,\|\, B \,\|\, Text1)"] & B
\end{tikzcd}$$

$x \,\|\, y$	denotes the concatenation of data item x and y
K_{AB}	denotes a secret key shared by A and B
eK_{AB}	denotes encryption using the shared secret key
T_A	denotes a time-stamp generated by A

Example 1 – discussion I

Consider four requirements for B:

1. Can B be sure M1 was generated by A? Given the encryption function provides origin/integrity and Text1 enables Text2 to checked, then Yes.

2. Can B sure M1 is fresh? Given T_A is sufficient recent and no similar message has been received in current time window then Yes.

Example 1 – discussion II

3. Can B be sure M1 was intended for it? Since B is included in the enciphered string, Yes
4. Is not applicable here.

Note that 3 explains why the name of B is included in M1

Use of Text1 and Text2 will depend on the application domain (Text1 might be used for session key transfer).

Example 2 (nonce & integrity mechanism)

This example can be found in clause 5.1.2 of ISO/IEC 9798-4

$$\begin{tikzcd}[column sep=huge] A \arrow[r, shift right, "M_2 = Text3 \| fK_{AB}(R_B \| B \| Text2)"'] & B \arrow[l, shift right, "M_1 = R_B \| Text1"'] \end{tikzcd}$$

fK_{AB} denotes a cryptographic check value (the out of data integrity mechanism) computed using shared key

R_B denotes random nonce generated by B

Example 2 – discussion I

Consider four requirements: for B

1. Can B be sure M2 was generated by A? Given the integrity mechanism is sound, then Yes.

2. Can B sure M2 is fresh? Given R_B is unpredictable then Yes.

3. Can B be sure M2 was intended for it? Since B is included in the scope of the check value, Yes.

Example 2 – discussion II

4. Can B be sure that M2 is a reply to M1? Yes, because of the inclusion of R_B

If random nonce generation difficult (and a predictable counter and instead) then an alternative protocol which give s properties 2 and 4 can be produced by modifying protocol to include protection to M1.

Example 2a (nonce & integrity mechanism)

This example can be found in clause 5.1.2 of ISO/IEC 9798-4

$$\begin{tikzcd}[column sep=huge] A \arrow[r, shift right, swap, "M_2 = Text3 \| fK_{AB}(R_B \| B \| Text2)"] & B \arrow[l, shift right, swap, "M_1 = R_B \| Text1 \| fK_{AB}(R_B \| Text1')"] \end{tikzcd}$$

Example 2a - discussion

Properties 2 and 4 are now provided even if R_A is predictable, since the impersonating entity Eve cannot now persuade entity A to generate message M2 ‘ahead of time’, since Eve can no longer produce an acceptable message M1.

Note that this is still a unilateral and not a mutual authentication protocol, i.e. A still cannot authenticate B, since A has no guarantees about the freshness of message M1.

Example 3 (nonce & encipherment)

This example can be found in clause 5.2.2 of ISO/IEC 9798-2

$$\begin{tikzcd}[column sep=huge, row sep=large]
A \arrow[r, leftarrow, shift left=6, "M_1 = R_B \,||\, Text1"] \arrow[r, "M_2 = Text3 \,||\, eK_{AB}(R_A \,||\, R_B \,||\, B \,||\, Text2)"] \arrow[r, leftarrow, shift right=6, "M_3 = Text5 \,||\, eK_{AB}(R_B \,||\, R_A \,||\, A \,||\, Text4)"'] & B
\end{tikzcd}$$

Example 3 – discussion I

Consider four requirements for A:

1. Can A be sure M1 and M3 were generated by B? A can check M3 because of the encipherment. Checking M1 is more difficult; however A can reason that B would not have sent M3 unless it was B who sent message M1, and hence we have “Yes”

2. Can A be sure M1 and M3 are fresh? Given R_A is unpredictable then A can check freshness of M3, and hence M1 (since M3 include R_B) – so Yes

Example 3 – discussion II

3. Can A be sure that M1 and M3 were intended for it? A can check that M3 is a reply to M2 and hence M3 must be for A. Thus, M1 was also for A (since M3 includes R_B) – so “Yes”.
4. Can A be sure that M3 was a reply to M2? Yes, because of the inclusion of R_A and R_B.

Example 3 – discussion III

B requirements are:

1. Can B be sure M2 was generated by A? Yes, because of encipherment.
2. Can B be sure M2 is fresh? Yes, because of inclusion of R_B (R_B must be unpredictable)
3. Can B be sure M2 is intended for it? Yes, because of the inclusion of the name B.
4. Can B be sure that M2 is reply to M1? Yes, because of the inclusion of RB.

If generating random nonces is difficult, then a modified protocol can be used (as in Example 2a this will involve cryptographically protecting the first message)

Example 4 (time-stamp & integrity mechanism)

This example can be found in clause 5.2.1 of ISO/IEC 9798-4

$$\begin{tikzcd}[column sep=huge] A \arrow[r, shift left=2, "M_1 = T_A \| Text2 \| fK_{AB}(T_A \| B \| Text1)"] & B \arrow[l, shift left=2, "M_2 = T_B \| Text4 \| fK_{AB}(T_B \| A \| Text3)"'] \end{tikzcd}$$

Note that, in order for A and B to perform their checks, A and B must have the means to obtain the data strings Text3 and Text1 respectively.

One possibility is that Text4 (Text2) contains a copy of Text3 (Text1), perhaps in enciphered form. Another possibility is that A and B can predict what the strings look like in advance.

Text1-Text4 are data strings, whose use will depend on the application

Example 4 – discussion I

Consider four requirements for A:

1. Can A be sure M2 was generated by B? Given that integrity mechanism is sound then “Yes”

2. Can A be sure M2 is fresh? Given T_B is sufficiently recent and what no similar message has been received in the current time window – “Yes”

3. Can A be sure that M2 is intended for it? “Yes”, because of the inclusion of A’s name in M2.

Example 4 – discussion II

4. Can A be sure that M2 was a reply to M1? **Not necessary!** There is nothing to link the two messages together

Property 4 can be guaranteed by including an identifier in M1 (Text1) and M2 (Text2)

Example 4 – discussion III

Four requirements for B:

1. Can B be sure M1 was generated by A? Given that the integrity mechanism is sound then "Yes"
2. Can B be sure M2 is fresh? Given T_A is sufficient recently and no similar message has been received in current time window – "Yes"
3. Can B be sure M1 is intended for it? Yes, because of the inclusion of the name B.
4. Does not apply

Example 5 (time-stamp & signature)

This example can be found in clause 5.2.1 of ISO/IEC 9798-3

$$\begin{tikzcd}[column sep=huge] A \arrow[r, shift left=2, "M_1 = T_A \| Text2 \| sS_A(T_A \| B \| Text1)"] & B \arrow[l, shift left=2, "M_2 = T_B \| Text4 \| sS_B(T_B \| A \| Text3)"] \end{tikzcd}$$

S_A and S_B are the private signature keys of A and B respectively

sS_A denotes the signature function computed using private key S_A

Note that, in order for A and B to perform their checks, A and B must have the means to obtain the data strings Text3 and Text1 respectively.

Example 5 – discussion

The analysis of this protocol is very similar to that Example 4.

Note that this means that, just like in Example 4, property 4 is not automatically satisfied

Example 6 (nonce & signature)

This example can be found in clause 5.2.2 of ISO/IEC 9798-3

$$\begin{tikzcd}[column sep=huge, row sep=large]
A \arrow[r, leftarrow, shift left=6, "M_1 = R_B \,\|\, Text1"] \arrow[r, "M_2 = R_A \,\|\, Text3 \,\|\, sS_A(R_A \,\|\, R_B \,\|\, B \,\|\, Text2)" description] \arrow[r, leftarrow, shift right=6, "M_3 = Text5 \,\|\, sS_B(R_B \,\|\, R_A \,\|\, A \,\|\, Text4)"'] & B
\end{tikzcd}$$

Example 6 - discussion

The analysis is very similar to Example 3.

As with Example 3, R_A needs be unpredictable, although R_B does not seem to need to be.

6 Comparisons and verification

Consider relative merits of different techniques

Possible extra properties required from authentication protocols.

Because of difficulty in designing sound protocols, many formal techniques devised to try and analyse protocols

Time-stamps vs. Nonces

Advantages of time-stamps:

Less message (typically one less),

Fits well to client-server model (e.g. RPC)

Disadvantages of time-stamps:

Need for synchronise clocks (and log of recently received messages),

Problems with property 4, i.e relating message of protocol together

Linking messages

Why do we want property 4? The need for this property depends on the application of the authentication protocol. If the protocol is used for:

Time synchronise, or Database query protection, then linking of a “request” message to a “response” message is needed (to prevent a malicious interceptor “shuffling” responses to requests issued within a short time of one another).

To address this problem, time-stamp protocols can use a “transaction ID” to achieve this linking of a request to a reply

Summary

A possible model for the use of authentication protocols based on the two basic types of “freshness” mechanism is as follows”:

Use random or encrypted nonce protocols for time synchronisation,

Use time-stamp based protocols for secure distributed applications, with transaction IDs if necessary to line messages.

Extra properties

Proposed list of four desirable properties for an authentication protocol is certainly not definitive.

Some specialised applications of authentication protocols extra properties may be needed.

One possible additional property:

5 After A(B) receives Mi then A(B) can be sure that B(A) knows the plaintext version of any enciphered data included in Mi.

Reason for this is illustrated by an X.400-1988 problem

Part of X.400 protocol

This example can be found in clause 5.1.1 of ISO/IEC 9798-3

$$\begin{tikzcd}[column sep=huge] A \arrow[r, "T_A \| B \| Text2 \| eP_B(Text2) \| sS_A(T_A \| B \| Text1 \| eP_B(Text2))"] & B \end{tikzcd}$$

This message use both public key encryption and a digital signature algorithm.

The Text 2 field is meant to be used for confidential data

T_A denotes a time-stamp generated by A

Discussion I

Note that this protocol involves signing enciphered data – generally accepted to be bad practice. In general, it is always better to sign data prior to encipherment

Suppose A wants to make an enquiry of public database B (the database is public but query is private)

A puts query in data string Text2.

Discussion II

C intercepts message and constructs new message with same *Text1* and *eP_B(Text2)* but with new signature

B responds to *C*, and *C* can deduce A’s confidential query from the response!

Problem is that property 5 is not provided

Keying requirements

Protocols based on symmetric cryptography (enchipherment or data integrity mechanism) need a shared secret between A and B.

Digital signature based protocols need A and B to have a trusted copy of each other’s verification key.

Look at these different keying requirements in turn.

Symmetric cryptography protocols

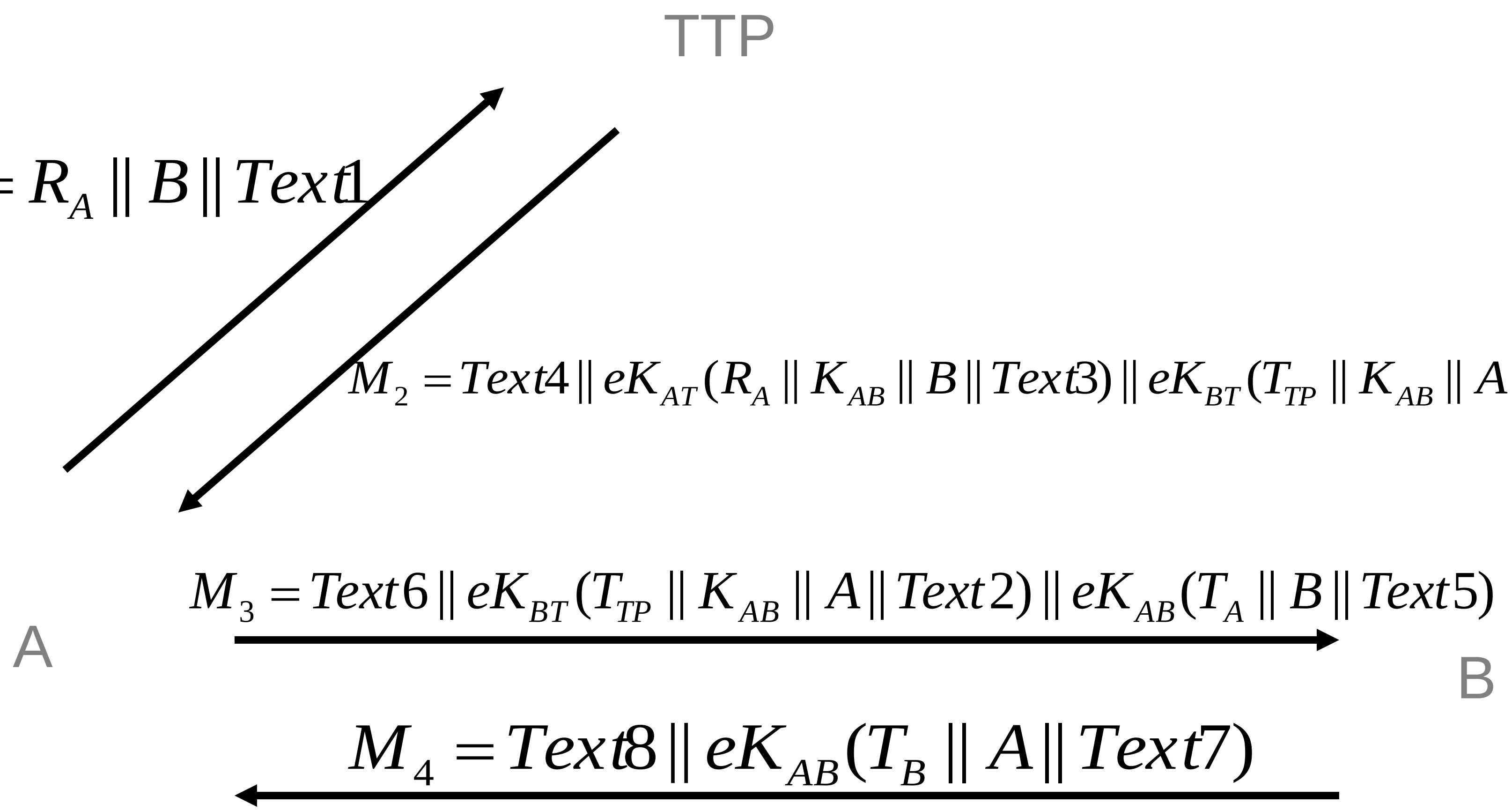
Assume there is a trusted third party with whom both A and B share a secret.

The (on-line) trusted third party (TTP) co-operates to enable A and B to authenticate one another.

Requires more elaborate protocols.

Example 7 (time stamp & encipherment)

This example can be found in clause 6.1 of ISO/IEC 9798-2



$$M_1 = R_A \,||\, B \,||\, Text1$$

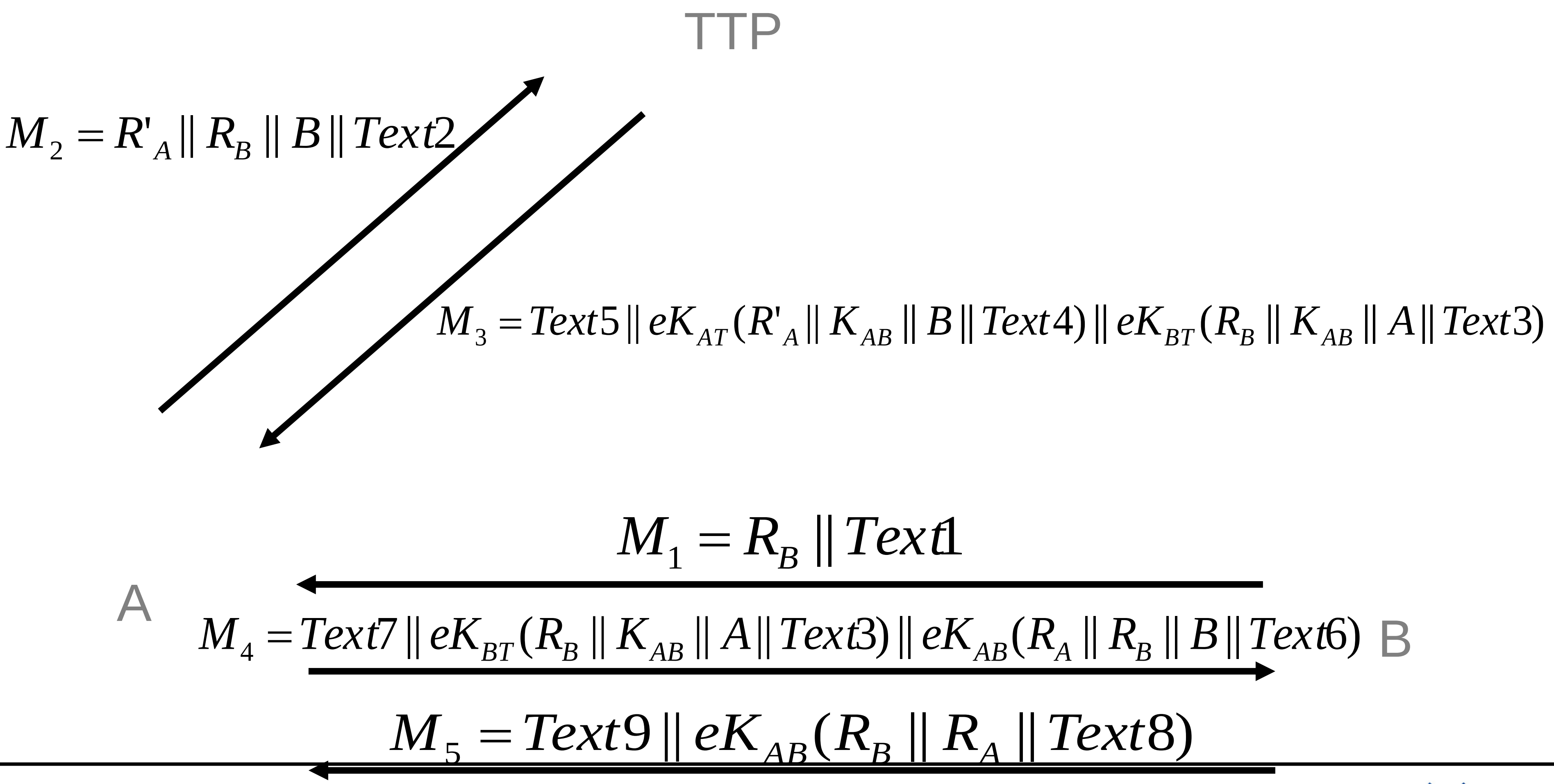
$$M_2 = Text4 \,||\, eK_{AT}(R_A \,||\, K_{AB} \,||\, B \,||\, Text3) \,||\, eK_{BT}(T_{TP} \,||\, K_{AB} \,||\, A \,||\, Text2)$$

$$M_3 = Text6 \,||\, eK_{BT}(T_{TP} \,||\, K_{AB} \,||\, A \,||\, Text2) \,||\, eK_{AB}(T_A \,||\, B \,||\, Text5)$$

$$M_4 = Text8 \,||\, eK_{AB}(T_B \,||\, A \,||\, Text7)$$

Example 8 (nonce & encipherment)

This example can be found in clause 6.2 of ISO/IEC 9798-2



$$M_2 = R'_A \| R_B \| B \| Text2$$

$$M_3 = Text5 \| eK_{AT}(R'_A \| K_{AB} \| B \| Text4) \| eK_{BT}(R_B \| K_{AB} \| A \| Text3)$$

$$M_1 = R_B \| Text1$$

$$M_4 = Text7 \| eK_{BT}(R_B \| K_{AB} \| A \| Text3) \| eK_{AB}(R_A \| R_B \| B \| Text6)$$

$$M_5 = Text9 \| eK_{AB}(R_B \| R_A \| Text8)$$

Kerberos - Introduction

Kerberos is a TTP-aided authentication protocol, closely related to Example 7.

Devised as part of Project Athena at MIT.

Designed to provide means to authenticate workstation users (clients) to servers (and vice versa)

Uses symmetric encipherment and a Manipulation Detection Code (MDC).

Kerberos is widely integrated into versions of the Unix OS, and code implementing the protocol is available on the Internet.

Kerberos - TTPs

Kerberos makes use of two types of TTP:

- An authentication server (AS), and
- A ticket-granting server (TGS).

User has a long-term shared secret key with the AS, which then sets up a short term shared secret key with the TGS.

The TGS is then involved in setting up shared session keys between client and server

Kerberos – motivation

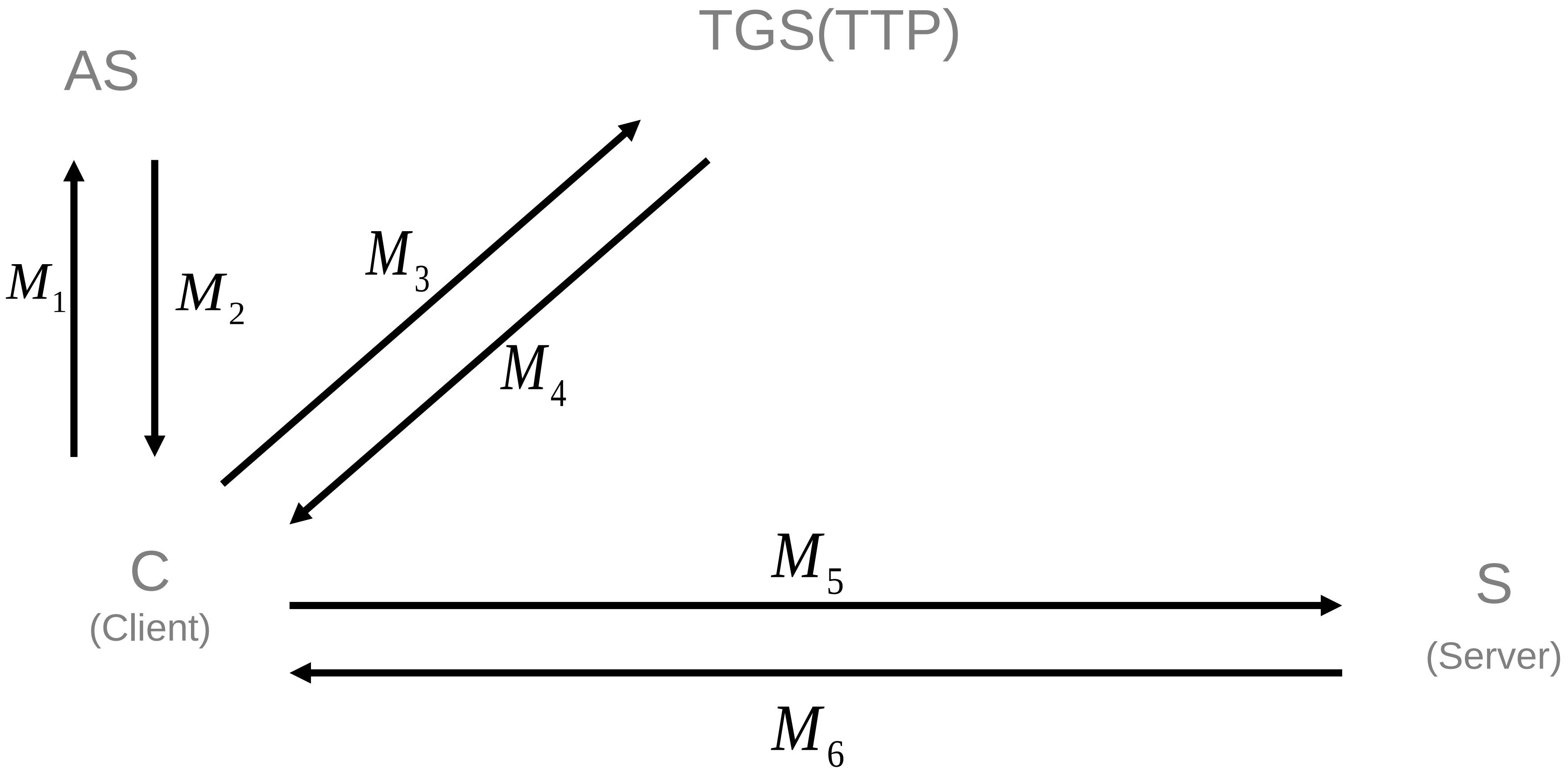
Idea of having two TTPs is that a user only needs load his/her long-term secret key into the work-station for the minimum time.

Once the short-term secret key is established (with TGS) the long-term secret key can be erased from the workstation.

This minimises the risk of exposure of the long-term secret key

Example 9 (Kerberos protocol)

This example can be found in clause 6.1 of ISO/IEC 9798-2



Kerberos – message formats

$$M_1(C-AS) = C \,||\, TG \,||\, times \,||\, N_C$$

$$M_2(AS-C) = C \,||\, eK_{AST}(K_{CT} \,||\, C \,||\, times) \,||\, eK_{ASC}(K_{CT} \,||\, times \,||\, N_C \,||\, TG)$$

$$M_3(C-TGS) = S \,||\, times \,||\, N'_C \,||\, eK_{AST}(K_{CT} \,||\, C \,||\, times) \,||\, eK_{CT}(C \,||\, T_1)$$

$$M_4(TGS-C) = C \,||\, eK_{TS}(K_{CS} \,||\, C \,||\, times) \,||\, eK_{CT}(K_{CS} \,||\, times \,||\, N'_C \,||\, S)$$

$$M_5(C-S) = eK_{TS}(K_{CS} \,||\, C \,||\, times) \,||\, eK_{CS}(C \,||\, T_2)$$

$$M_6(S-C) = eK_{CS}(T_2)$$

K_{AST} is a secret key shared by the AS and TGS

K_{ASC} is a secret key shared by the AS and C

K_{TS} is a secret key shared by the TGS and S

K_{CT} is a secret key shared by the C and TGS

K_{CS} is a secret key shared by the C and S

$times$ denotes a specified time interval (start and end time) – it is used to limit the validity of a key

Digital signature protocols

Need for trust instead of shared secret

Public verification keys can be certified by applying the digital signature of a Trusted Third Party (TTP).

Result (public key + entity name + expiry date + TTP signature on three items) called a certificate.

Using certificates

To check a certificate signed by a TTP requires a trusted copy of the TTP’s public verification key.

If two entities have certificates signed by different TTPs, then a cross-certificate is needed (i.e. one TTP’s public verification key signed by the other TTP).

Leads to notion of certification paths, i.e. sequence of cross-certificates with the subject of one certificate being the signer of the next certificate in the sequence.

Key Distribution

One very important application of authentication protocol is during connection establishment.

Can be used to set up session key(s) to protect data transferred during connection lifetime.

Keys can be transferred by inclusion in data string elements of protocol messages.

Note that, in some applications, property 5 is needed, i.e. the recipient of an enciphered key wishes to be sure that the sender actually knows the plaintext key, since the key is to be used subsequently to guarantee the origin of data.

Example 10

This example can be found in clause 5.3 of ISO/IEC 11770-2

$$A \xrightarrow{M_1 = Text2 \,||\, eK_{AB}(T_A \,||\, B \,||\, SK_{AB})} B$$

This protocol has been derived in a very simple way from the protocol of Example 1. The Text1 field has been replaced with a session key SK_{AB}, to be used by A and B in subsequent secure communications.

Note that protocol property 5 is guaranteed since A Knows the key K_{AB}

Example 11

This example can be found in clause 5.6 of ISO/IEC 11770-2

$$\begin{tikzcd}[column sep=huge, row sep=large]
A \arrow[r, shift left=6, leftarrow, "M_1 = R_B"] \arrow[r, "M_2 = eK_{AB}(R_A \| R_B \| B \| F_A)" description] \arrow[r, shift right=6, leftarrow, "M_3 = eK_{AB}(R_B \| R_A \| F_B)"'] & B
\end{tikzcd}$$

This protocol is based on Example 3. Text1, Text3 and Text5 are not used. Text2 and Text4 have been replaced by keying material fields F_A and F_B, which can be combined to form a session key for use by A and B

Note that protocol property 5 is guaranteed since A knows the key K_{AB}.

Summary

There are four basic properties and one extra property for the protocol requirements. A /(B) wants to be sure that”

M2, M4 /(M1, M3), … were all sent by B / (A) (as received),

M2, M4 /(M1, M3),… are “fresh”, i.e. not replays of old messages

M2, M4 /(M1, M3),… were intended for A / (B), not for any other entity, and

M2, M4 /(M3, M5),… were only generated by B / (A) after M1, M3 /(M2, M4),… (respectively) were received correctly.

After A/(B) receives Mi then A/(B) can be sure that B/(A) knows the plaintext version of any enciphered data included in Mi.