Introduction to Information Security (IY2760/DC3760): Introduction to key establishment

Dr Siaw-Lynn Ng

November 7, 2023

Contents

1 Introduction

Symmetric cryptosystems and key establishment

- *Symmetric* cryptosystems (*private key* cryptosystems):
  - Encryption and decryption keys are identical.
  - Parties must agree a key before communication.
  - Compromise of key compromises system.
- Properties of keys:
  - Large enough for security.
  - Easy enough to handle.
- Major issue in symmetric cryptography.

Public key cryptosystems and key establishment

- PKC: no need of secure channel for key exchange.
- Issue 1: authenticity of Bob's public key in the presence of active adversary.
- Issue 2: Most PKC are slower than symmetric key systems.

Long-term and short-term keys

- Users (or pairs of users) may have *long-term keys*:
  - Precomputed and stored securely;
  - Or computed from securely stored secret information (key pre-distribution).
  - Often used to transmit session keys.
- *Session keys*:
  - Short-term key for a particular session only.
  - Can be updated frequently to limit amount of ciphertext (encrypted with one key) available to cryptanalyst, and to limit exposure if session key is compromised.

2 Security protocols

Security protocols

- A protocol is a set of rules for exchanging messages between two or more principals/participants over a network.
- Rules cover:
  - Message formats.
  - How to handle the messages on receipt.
  - How messages are interpreted.
- Here we will see how basic cryptographic primitives (encryption, MACs, signatures) can be used to provide security services (authenticated key establishment) over insecure networks.
- Build on ISO 7498 OSI generic security architecture.

A secure protocol

In a secure protocol:

- When acting honestly, principals/participants achieve the stated *aim* of the protocol.
  - Example 1: A authenticates B.
  - Example 2: A and B sets up a fresh session key (with certain assurances).
- Neither a passive eavesdropper nor an active adversary can defeat this objective.
  - In Example 1: Oscar cannot successfully impersonate B to A.
  - In Example 2: Oscar cannot persuade A and B to reuse an old session key.

The legitimate principals/participants

- Legitimate participants: conventionally called Alice, Bob, Carol, *etc.*
- In more complex protocols there may be a Trusted Third Party (TTP) who is trusted by the legitimate participants.
  - Depending on the application they may be called Trusted Third Party, Trusted Authority, Trusted Server, Certification Authority, *etc.*
  - There may be varying levels of trust, for example, trusted to relay messages correctly, trusted to generate keys, trusted to verify identities, *etc*

The adversaries: Eve, Mallory, Oscar

There are two kinds of adversaries:

- *Eve*, a passive adversary, an eavesdropper.
  - Eve can only read sent messages.
- *Mallory/Oscar*, an active adversary, who can:
  - view, alter, delete, replay message.
  - inject messages into the network.
  - initiate protocol runs.
  - impersonate a principal in a protocol run.

The adversaries: What Mallory/Oscar can't do (I)

We assume that the underlying cryptographic primitives are secure:

- The (pseudo)random number generation is secure:
  - Mallory cannot guess a random number chosen by another principal if it is selected from a sufficiently large space.
- The hash function is secure:
  - Mallory cannot easily find preimages or collisions.

The adversaries: What Mallory/Oscar can't do (II)

We assume that the underlying cryptographic primitives are secure:

- The encryption algorithm is secure.
- For example,
  - In a symmetric key cryptosystem, Mallory cannot deduce the key from observing plaintext-ciphertext pairs.
  - In a public key cryptosystem, Mallory cannot deduce the private key from a public key.
- The signature scheme is secure: Mallory cannot deduce the signing key from the public verification key and from observing message-signature pairs.

Summary of assumptions

- We equip legitimate participants with idealised cryptographic mechanisms.
- Legitimate participants exchange messages over an untrusted communication network.
- How then do we use these cryptographic mechanisms to design secure protocols?

3 Authenticated key establishment

Key establishment

From Handbook of Applied Cryptography:

- Key establishment: process by which a shared secret key becomes available to two or more parties for subsequent cryptographic use.
  - Key distribution: one party chooses a key and transmits it securely to others.
  - Key agreement: the secret key is derived by all parties as a function of inputs by all parties.

Key management

From Handbook of Applied Cryptography:

- Key management: set of processes and mechanisms which support key establishment and the maintenance of ongoing keying relationships between parties.
  - Eg. key generation, distribution, storage, update, destruction *etc.*

Entity authentication and key establishment

- Entity authentication can only be achieved for an instant in time.
  - Typically this is established at the start of a connection/session.
- If we want security (confidentiality/integrity) for a whole session we need to establish a session key.
- A session key can be agreed as part of an authentication protocol.
  - The session key can be bound to that protocol run.
  - This can be done in an authenticated key establishment protocol.

Participants in a key establishment protocol

- Legitimate participants: Alice, Bob, Carol, *etc.*
- Trusted parties: Trusted Third Party, Trusted Authority, Trusted Server, Certification Authority, *etc.*
  - TTP can be online or offline.
  - Can be certification authority, vouching for authenticity of public keys or key generator or key escrow agent *etc.*
  - Varying levels of trust.
- Adversaries:
  - Can be passive or active, outsider or insider.

Key establishment protocols

- Many different scenarios and methods and models:
  - *Key pre-distribution*: TTP distributes keying information ahead of time securely. Pairs of users can derive secret keys later on.
  - *Key transport/distribution*: One party creates and transfers the key to other parties.
  - *Key agreement*: Users agree on session key using interactive protocols, maybe based on symmetric-key or public-key schemes. Usually do not require on-line TTP.

Security goals of key establishment protocols

- Implicit key authentication:
  - No one other than specified party may gain access to a key
- Key confirmation:
  - Assurance that second party (possibly unspecified) has actual possession of a key
- Explicit key authentication: both implicit key authentication and key confirmation
- Entity authentication:
  - Assurance of identity and liveness of communicating party

Other assurances and considerations (I)

- Other assurances:
  - Key freshness: guarantee that new key is used.
  - Key control: neither party can control/predict key value.
- Other considerations:
  - Efficiency: number of passes and bandwidth, complexity of computations.
  - TTP requirement: on-line, off-line, or none; degree of trust.

Other assurances and considerations (II)

- Security under different attack models:
  - Security if a session key is known?
  - Security if long-term key is known?
- *Perfect forward secrecy*: compromise of long-term keys does not affect security of short-term keys before the compromise.

Adversaries

- Oscar may be a passive adversary and restrict action to eavesdropping.
- Oscar may be an active adversary, and can
  - alter messages, replay recorded messages, masquerade as other users.
- Oscar's objectives?

Example: Session keys

Two parties, Alice and Bob, share a secret *long-term key* k.
Session key establishment:

- Alice sends Bob k' in the clear.
- *Session key* is $k \oplus k'$.

Weaknesses?

Example: Key hierarchies

Two parites, Alice and Bob, share a master key k_M.
Session key establishment:

- Alice sends Bob session key as $e_{k_M}(k_S)$.
- Bob decrypts and obtain session key k_S.
- Many level hierarchies are possible.
- Weaknesses?

Example: Using TTP

TTP is an agency trusted by all parties.

- TTP can generate and convey keys.
- Each user has secret key agreed with TTP.
  - Alice shares secret key k_A with TTP.
  - Bob shares secret key k_B with TTP.
- When Alice and Bob wish to communicate:
  - TTP generates session key k_S.
  - TTP sends Alice $e_{k_A}(k_S)$ and Bob $e_{k_B}(k_S)$.
- Weaknesses?

ISO/IEC 9798 standards (NON-EXAMINABLE)

ISO/IEC 9798, a multi-part standard, specifies a variety of authentication protocol and related key distribution protocols:

- ISO/IEC 9798-1: 1997 (2nd edition) - General.
- ISO/IEC 9798-2: 1999 (2nd edition) - Mechanisms using symmetric encipherment algorithms.
- ISO/IEC 9798-3: 1998 (2nd edition) - Mechanisms using digital signature techniques.

- ISO/IEC 9798-4: 1999 (2nd edition) - Mechanisms using a cryptographic check function.
- ISO/IEC 9798-5: 1999 - Mechanisms using using zero knowledge techniques.

3.1 Key pre-distribution

Key pre-distribution

- TTP distribute keying information securely ahead of time.
  - Preload keys/keying material on to devices in controlled environment before deployment.
- Pairs of users later determine key from keying information.
- Evaluation criteria:
  - How much information to be transmitted securely,
  - How much information to be stored securely,
  - Others: how much information to be published or broadcast, how much computation to be performed by TTP and users.

Examples of key pre-distribution

- A trivial example: for each pair of users U, V, TTP chooses random key K_{UV} and transmit it securely to U and V.
  - Strength and weaknesses?
- Example: Preloaded keys in SIMs on mobile phones.
- Example: Set top boxes for digital TV services.
- Issues:
  - Keeping track of device ownerships.
  - Post deployment key management.

3.2 Key distribution

Key distribution protocols

- One party (could be TTP) chooses a session key and securely transfers it to the others.
- Many different scenario possible:
  - Using symmetric key cryptosystems only, or PKC only, or a hybrid.
  - Different levels of involvement of TTP and trust in TTP.
  - Different levels of input to the session key.

Authenticated key establishment: symmetric key techniques

- Simple example using symmetric key cryptosystems and time-stamps:
  - A and B share a long-term key K.
  - $A \to B$: $e_K(t||i_B||K_s)$
  - t is a time stamp, i_B is an identifier for B, and K_s is a session key.
  - A is authenticated to B and they now share a secret session key K_s.
- Implicit key authentication: No one other than B (and A) may gain access to K_s.

Authenticated key establishment: public key techniques

- A checks the authenticity of B's public key PK_B.
- $A \to B$: $e_{PK_B}(K_s)$.
- Subsequent messages are encrypted or authenticated using K_s (or keys derived from K_s):

$$B \to A: \text{ data, } MAC_{K_s}(\text{data})$$

- Assurances?
  - A is not authenticated to B.
  - B is authenticated to A if subsequent messages are correctly encrypted/authenticated using K_s.
  - Explicit key authentication if subsequent messages are correct: only B (and A) could have K_s and B does actually have K_s.

Key transport protocol using TTP

- The "wide-mouthed frog protocol":
- Alice shares a key K_{AT} with TTP.
- Bob shares a key K_{BT} with TTP.
- If Alice and Bob wish to communicate, then Alice chooses session key K_{AB} and TTP transfers it to Bob securely.

1	Alice $\to$ TTP	$e_{K_{AT}}(t_A\|ID_B\|K_{AB})$
2	TTP $\to$ Bob	$e_{K_{BT}}(t_T\|ID_A\|K_{AB})$

- Security?

3.3 Keberos

Kerberos

- Kerberos is a TTP-aided authentication protocol.
  - Can achieve mutual authentication and key establishment.
- The name also refers to software implementing that protocol, currently Kerberos V5 Release 1.2.
- Also the name of a project at MIT which devised the protocols (properly called Project Athena).
- Standardised in RFC 1510 – Kerberos V5 (1992).
- Version of Kerberos incorporated in Windows and used in many versions of Unix

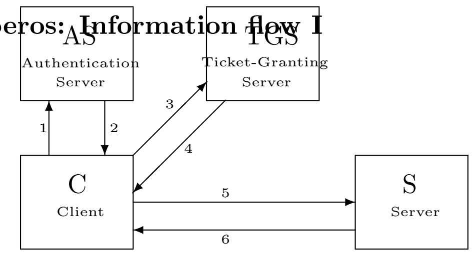
Kerberos: Principals

- Authentication of Client (C) to Server (S) done as a two-stage process.
- Authentication Server (AS)
  - Mutual authentication with Client at login based on a shared long-term secret.
  - Gives client *ticket granting ticket* and a *short-term key* for use between Ticket Granting Server and Client.
- Ticket Granting Server (TGS)
  - Performs mutual authentication with Client based on the short-term key and ticket granting ticket.
  - The TGS then issues tickets giving Client access to further Servers that demand authentication.

Kerberos: Motivation

- Two TTPs: Authentication Server (AS) and Ticket Granting Server (TGS)
  - A user only needs to load their long-term secret key (shared with AS) into the client host for the minimum time.
  - Once the short-term key is established (with TGS) this long-term secret key can be erased from the client host.
- All further client interactions are with TGS and servers.
- This minimises the risk of exposure of the long-term secret key.

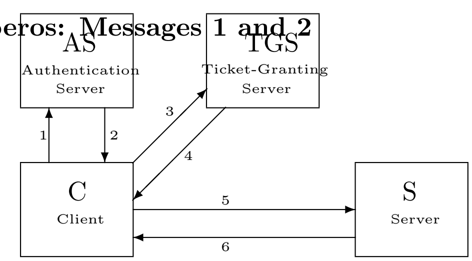
Kerberos: Information flow I



Kerberos: Information flow II

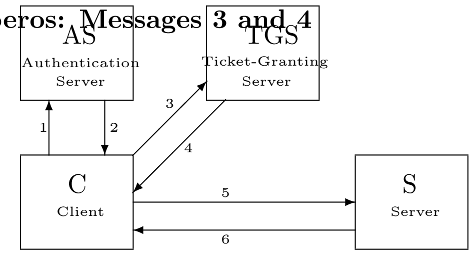
- Messages 1 and 2 are exchanged between the client C and the authentication server AS.
  - They derive a short term key to use with the TGS in messages 3 and 4.
- Messages 3 and 4 are exchanged between the client C and the ticket-granting server TGS (using the short-term key provided by the AS).
  - They derive a short term key to use with the server in messages 5 and 6.
  - This can be repeated without repeating messages 1 and 2.
- Messages 5 and 6 are exchanged between the client C and server S (using a key provided by the TGS).
  - This can be repeated without repeating messages 3 and 4.

Kerberos: Messages 1 and 2



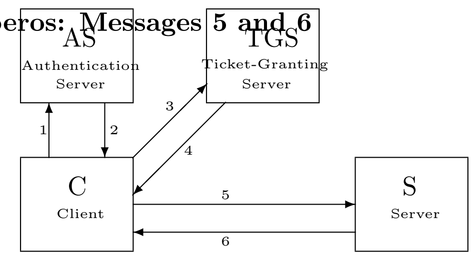
- C and AS share long-term key $K_{AS,C}$ derived from C's password.
- C and AS use $K_{AS,C}$ to mutually authenticate one another.
- C and AS derive a short-term key $K_{C,TGS}$ and a ticket-granting ticket to be used with TGS in messages 3, 4.

Kerberos: Messages 3 and 4



- C presents the ticket-granting ticket from AS to TGS.
- They mutually authenticate each other using $K_{C,TGS}$.
- They derive a session key $K_{C,S}$ and a (session-granting) ticket to be used with the Server S in messages 5, 6.

Kerberos: Messages 5 and 6



- C presents the session granting ticket to S.
- S authenticates C using $K_{C,S}$.
- Optionally, S can send C another message to authenticate itself to C.

Kerberos: Notation

i_X	Identifier of principal X.
N_C, N'_C	Nonces generated by the client C.
$K_{AS,C}$	Long term secret key shared by AS and C.
$K_{AS,TGS}$	Long term secret key shared by AS and TGS.
$K_{TGS,S}$	Long term secret key shared by TGS and S.
$K_{C,TGS}$	Short term secret key shared by C and TGS.
$K_{C,S}$	Short term secret key shared by the C and S.
T_1, T_2	Time-stamps.
L, L'	Life time, specifying validity period of a key.
$e_K()$	Encryption using symmetric cryptosystem with key K.

Kerberos simplified message format: Messages 1, 2

1. C $\rightarrow$ AS: $i_C||i_{TGS}||L||N_C$
2. AS $\rightarrow$ C: $i_C||\underbrace{e_{K_{AS,TGS}}(K_{C,TGS}||i_C||L)}_{\text{Ticket-granting ticket}}||e_{K_{AS,C}}(K_{C,TGS}||N_C||L||i_{TGS})$

- C and AS use $K_{AS,C}$ derived from client password to authenticate one another.
- They derive a short-term key $K_{C,TGS}$, and a ticket-granting ticket to allow C to talk to TGS in messages 3 and 4.
- The ticket includes $K_{C,TGS}$ and ticket lifetime L encrypted under longterm key $K_{AS,TGS}$.

Kerberos simplified message format: Messages 3, 4

3. C $\rightarrow$ TGS: $i_S||L||N'_C||e_{K_{AS,TGS}}(K_{C,TGS}||i_C||L)||e_{K_{C,TGS}}(i_C||T_1)$
4. TGS $\rightarrow$ C: $i_C||\underbrace{e_{K_{TGS,S}}(K_{C,S}||i_C||L')}_{\text{Session-granting ticket for S}}||e_{K_{C,TGS}}(K_{C,S}||N'_C||L'||i_S)$

- C presents request for access to server S along with ticket granting ticket and a message authenticating C to TGS (Message 3).
- TGS checks validity and lifetime of ticket granting ticket and extracts $K_{C,TGS}$. TGS can now authenticate the Client.
- If all OK, TGS issues session key $K_{C,S}$ and session-granting ticket to C. (Default validity is 5 minutes.)
- TGS also authenticates itself to C (Message 4).

Kerberos simplified message format: Messages 5, 6

5. C $\rightarrow$ S: $e_{K_{TGS,S}}(K_{C,S}||i_C||L')||e_{K_{C,S}}(i_C||T_2)$
6. S $\rightarrow$ C: $e_{K_{C,S}}(T_2)$

- C presents session-granting ticket along with a message authenticating C to S (Message 5).
- S checks validity and lifetime of session-granting ticket and extracts session key $K_{C,S}$. S can now authenticate C.
- If all OK, S grants access to C.
- Optionally, S sends C a message authenticating S to C (Message 6).

Kerberos: Use of cryptography

- Kerberos uses symmetric encryption and Manipulation Detection Codes (MDC).
- The MDC is computed on the data to be encrypted, and then the concatenation of the MDC with the data is encrypted.
- Specifically, Kerberos version 5 (as originally in RFC 1510) uses DES and MD4 or MD5.
- Release 1.2 of Kerberos Version 5 implements triple DES (3DES).

Kerberos issues I

- Revocation: ticket granting tickets valid until they expire, typically 10 hours.
- Within realms (domains): long-term keys need to be established between AS and TGS, TGS and Servers and AS and clients.
- Synchronous clocks are needed, and must be protected against attacks.
- Cache of recent messages to protect against replay.

Kerberos issues II

- AS and TGS must be trusted by clients not to eavesdrop.
  - Can be extended to include keying material to establish additional secret not chosen by AS or TGS.
- Client-AS long-term key often still based on password entry – vulnerable to guessing.
- Short-term keys and ticket granting tickets located on largely unprotected client hosts.
- Denial of service possible? E.g. on the clock service or on the TGS.

Kerberos and Windows network authentication

- Microsoft has adopted and extended Kerberos to provide network authentication in Windows.
- One extension: support for public key encryption to protect client/AS messages (rather than password-based long-term key).
- Second extension: use Kerberos (normally empty) data authorisation field to transmit access privileges.
- Message formats proprietary to Microsoft.
- Non-standard extension to Kerberos makes it hard to interoperate Microsoft & non-Microsoft implementations.

4 Key agreement

Key agreement protocols

- Key agreement: secret key derived by all parties as a function of inputs by all parties.
  - May or may not involve a TTP.
- Diffie-Hellman key exchange:
  - Allow two parties who have not met in advance or shared keying material to establish shared secret by public exchange of message.
  - First practical solution.

4.1 Diffie-Hellman key exchange

Diffie-Hellman key exchange

- Public information: prime p and primitive element α.
- Alice chooses secret x_A at random $(0 \leq x_A \leq p-2)$.
  - Alice sends $y_A = \alpha^{x_A} \bmod p$ to Bob.
- Bob chooses secret x_B at random $(0 \leq x_B \leq p-2)$.
  - Bob sends $y_B = \alpha^{x_B} \bmod p$ to Alice.
- Alice calculates $k = y_B^{x_A} = \alpha^{x_A x_B} \bmod p$.
- Bob calculates $k = y_A^{x_B} = \alpha^{x_A x_B} \bmod p$.

Example: Diffie-Hellman key exchange

Public information: prime $p = 59$, primitive element $\alpha = 2$.

Alice chooses secret key $x_A = 13$		Bob chooses secret key $x_B = 41$
Alice calculates y_A : $y_A = 2^{13} = 50 \bmod 59$		Bob calculates y_B: $y_B = 2^{41} = 34 \bmod 59$
	$\xrightarrow{y_A}$ $\xleftarrow{y_B}$	
Alice calculates $y_B^{x_A}$ $y_B^{x_A} = 34^{13} = 42 \bmod 59$		Bob calculates $y_A^{x_B}$ $y_A^{x_B} = 50^{41} = 42 \bmod 59$

Agreed secret value $k = 42$.

The Diffie-Hellman problem

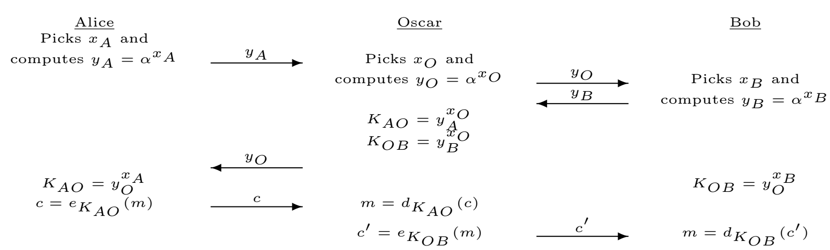
- Outsider knows p and α, $y_A = \alpha^{X_A} \bmod p$, $y_B = \alpha^{X_B} \bmod p$.
  - Determine $k = \alpha^{X_A X_B} \bmod p$ from this information.
- *Diffie-Hellman problem*: Given prime p, primitive element α:
  - Given $\alpha^{x_A} \bmod p$ and $\alpha^{x_B} \bmod p$.
  - Find $\alpha^{x_A x_B} \bmod p$.
- Example: Prime $p = 59$, primitive element $\alpha = 2$
  - Given $2^{x_A} = 47 \bmod 59$, $2^{x_B} = 33 \bmod 59$, find $2^{x_A x_B} \bmod 59$
- Diffie-Hellman problem believed to be hard.

The Diffie-Hellman problem and the discrete log problem

- The *discrete log problem*:
  - Prime p and primitive element α.
  - Given $\beta \in \mathbb{Z}_p$, find integer a $(0 \le a \le p-2)$ such that $\beta = \alpha^a \bmod p$.
- Can try to calculate x_A from $\alpha^{x_A} \bmod p$ - the discrete log problem.
  - Solution to discrete logarithm gives solution to Diffie-Hellman problem.

Authentication in Diffie-Hellman key exchange

- Diffie-Hellman key exchange does not provide entity or key authentication.
  - subject to intruder-in-the-middle attacks:



4.2 STS protocol

Station-to-station (STS) protocol

- Public information: prime p, primitive element α.
- User U: signature generation function S_U, signature verification algorithm V_U certified by TTP.

1	Alice $\to$ Bob	$y_A = \alpha^{x_A} \bmod p$
2	Bob $\to$ Alice	$y_B = \alpha^{x_B} \bmod p$, $e_K(S_A(y_B \| y_A))$
3	Alice $\to$ Bob	$e_K(S_B(y_A \| y_B))$

- $K = \alpha^{X_A X_B} \bmod p$ can be calculated by Bob after the first message, and Alice after the second message.
- Achieves key agreement, mutual entity authentication, explicit key authentication.