# INTE2401/2402 Lab 8

Student ID: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

This week’s lab concentrates on establishing secure communication between two parties. In the first task, we utilise online AES tool to demonstrate the model of Kerberos network authentication protocol. In the second task, we implement a JavaScript program of Diffie-Hellman key exchange method to securely exchange cryptographic keys over a public channel. Detail will be introduced below.

## Task 1. Demo of Kerberos network authentication protocol

Kerberos is a network authentication protocol. It is designed to provide reliable authentication over open and insecure networks. It uses secret-key cryptography so that a client can prove its identity to a server (and vice versa) across an insecure network connection. After a client and server has used Kerberos to prove their identity, they can also encrypt all of their communications to assure privacy and data integrity as they go about their business.

Figure 1 illustrates the model of the Kerberos protocol. Besides the client and the server, the model involves trusted third parties: Key Distribution Centre (KDC), Ticket Granting Server (TGS). Among these parties, there are 3 phases: Phase 1 - client gets a Ticket-Granting Ticket (TGT) from the KDC; Phase 2 - client uses the TGT to get a Ticket for server; Phase 3 - client communicates with server. Following the completion of 3 phases, the client and server are able to establish a trustworthy connection, and encrypted communication if required.

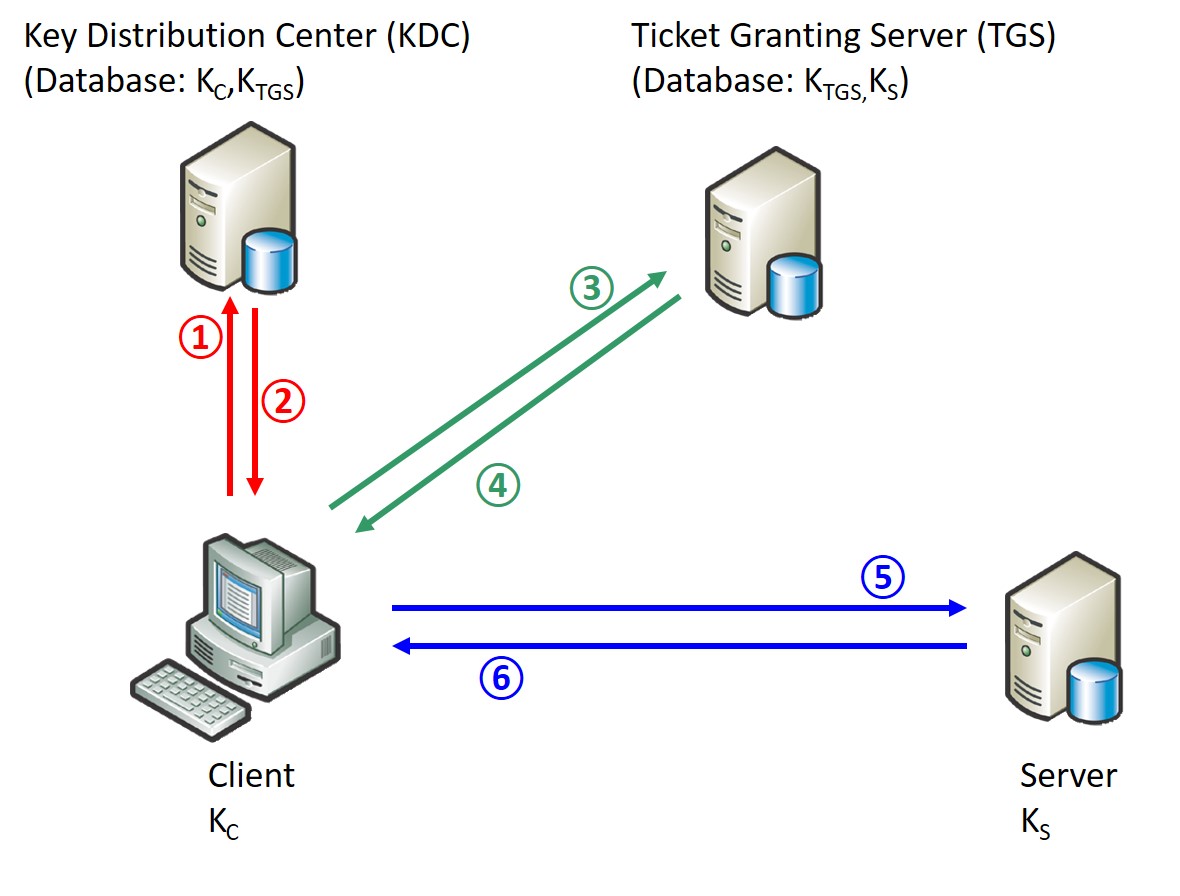


Figure . Model of Kerberos protocol

In this task, we demonstrate how authentication being provided in each phase. For brevity, we omit the “Lt” and “nC” in all phases. We use online AES-CBC tool to perform encryption. Noted that the encryption of multiple messages is performed over the concatenation of all messages. For example, {KC,TGS, TGS}KC is performed in the way {KC,TGS || TGS}KC. Besides, all secret keys are derived from the student no. Other required informations and example are provided in the table below. Noted that the encryption of multiple Task is emulated by filling the question table.

Online AES encryption and decryption: <http://rubbingalcoholic.github.io/cowcrypt/demos/aes.html>

Question table:

|  |  |
| --- | --- |
| Id of Client: | 0001 |
| Id of Server: | 0010 |
| Id of KDC: | 0011 |
| Id of TGS: | 0100 |
| KC:  (if student no. s3701234) | F3701234F3701234F3701234F3701234F3701234F3701234F3701234F3701234 |
| KTGS:  (if student no. s3701234) | 1234F3701234F3701234F3701234F3701234F3701234F3701234F3701234F370 |
| KS:  (if student no. s3701234) | 4321073F4321073F4321073F4321073F4321073F4321073F4321073F4321073F |
| Session key KC,TGS: | 0123456712345678234567893456789a0123456712345678234567893456789a |
| Session key KC,S: | a9876543987654328765432176543210a9876543987654328765432176543210 |
| Ticket TC,TGS = {KC,TGS, C}: | 0123456712345678234567893456789a0123456712345678234567893456789a0001 |
| Ticket TC,S = {KC,S, C}: | a9876543987654328765432176543210a98765439876543287654321765432100001 |
| Session key sk: | 0123456701234567012345670123456701234567012345670123456701234567 |
| Timestamp ts: | 1536755037 |
|  | |
| **Tag ①: Client sends a request to the KDC for a “ticket-granting ticket” (TGT)** | |
| Number of phase:  (Example) | Phase 1 |
| Message transferred:  (Example) | Plaintext {C, TGS} |
| Value of transferred message:  (Example) | {00010011} |
| **Tag ②: KDC responds to client** | |
| Number of phase: | Phase 1 |
| Message transferred:  (Example) | {KC,TGS, TGS}KC= {0123456712345678234567893456789a0123456712345678234567893456789a0100}KC  ,  {TC,TGS}KTGS={KC,TGS, C}KTGS ={0123456712345678234567893456789a0123456712345678234567893456789a0001 }KTGS |
| Value of transferred message:  (Example) | {xaIMFXLB9LIXpwaf2YRiSakN7J3zfUyZb/ejreYgnCJ8Gd16es1PybDnehWz190tv9ii5rLvjhw9wP7dlVkruitM1wzlvXUcgOOmMtrmbFk=}, {sxQTKJDt69fcv6qoWgsDmZ4feYIBBToxX4pvmpMr3KWFiDfA2WXJM2GnAVitYu8HoSlUuW8q5ZMFE54WmtBR38QcCzCvJB6nVBSBlx6Gth8=} |
| **Tag ③: Client requests a ticket to communicate with server from the ticket-granting service (TGS)** | |
| Number of phase: | Phase 2 |
| Message transferred: | {ts}KC,TGS , {TC,TGS}KTGS={ KC,TGS, C } KTGS , S |
| Value of transferred message: | {LtepvYiUPEeT3TNRJyNfBw== }  , {sxQTKJDt69fcv6qoWgsDmZ4feYIBBToxX4pvmpMr3KWFiDfA2WXJM2GnAVitYu8HoSlUuW8q5ZMFE54WmtBR38QcCzCvJB6nVBSBlx6Gth8=}  ,  {0010} |
| **Tag ④: TGS returns a ticket for client to talk to server** | |
| Number of phase: | Phase 2 |
| Message transferred: | {KC,S, S}KC,TGS, {TC,S}KS={KC,S, C}KS |
| Value of transferred message: | {oDPp3AW48JJPiC3l+pj8ghw2lMw7AYf2fTwRIylQV6SwEUoawNdxOZlL05E9R5p75shYG+nyp2b3pz3ZPDEV4qECZKDENeA8U8kA66oDX5M= }  ,  {RC6dkFyvjV8iPhpcLdzuqalxxq2qnZe1PAilJ4LtWH3ZqOBJPIHj41hprW/kSBUXJcUx7bKoeCIdkOqYIM8m5nQ0xpiSVIYqbPyE3mBqLU0= } |
| **Tag ⑤: Client sends the ticket to server along with an authenticator to establish a shared secret** | |
| Number of phase: | Phase 3 |
| Message transferred: | {C, ts, sk}KC,S, {TC,S}KS={KC,S, C}KS |
| Value of transferred message: | { YHmjr3W8r/rtIqv5oXKHKKG0/cWY33UoM9b2op3bGfvBeMqbfjYJehgunJhYLCVhPOJ91Id1iHoMY37PL80u/wZYNwvHcRqYD8lgIskPyDA=}  ,  {RC6dkFyvjV8iPhpcLdzuqalxxq2qnZe1PAilJ4LtWH3ZqOBJPIHj41hprW/kSBUXJcUx7bKoeCIdkOqYIM8m5nQ0xpiSVIYqbPyE3mBqLU0= } |
| T**ag ⑥: Server decrypts the ticket to obtain the KC,S and replies to client with proof of possession of the shared secret** | |
| Number of phase: | Phase 3 |
| Message transferred: | {ts, sk}KC,S |
| Value of transferred message: | { LZJovBDQvTRVmvpRRyCT1pXJDSJcvm3IExaRXGph6aGasTjEGt++OkXwXD4J5NwWyuFMG0ASr3Fl6tOkymhmzZtnRn0zoZTU3o6XeCEURXY= |

## Task 2. Implementation of Diffie-Hellman key exchange method

Diffie-Hellman key agreement, shown in Figure 2, is an exponential key agreement to perform real-time key exchange over an untrusted network. It allows two users to change a secret key without requiring prior secrets. In this task, we use JavaScript to implement a prototype of Diffie-Hellman key agreement, and we output the public values and shared secret .

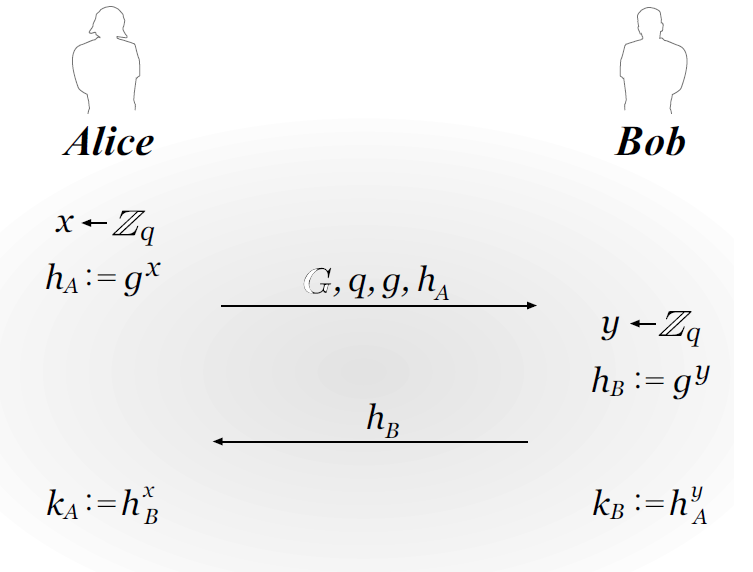


Figure 2. The Diffie-Hellman key-exchange protocol

Q2. Implement the prototype of Diffie-Hellman key exchange protocol and output the intermediate results and shared secret between Alice and Bob.

Example inputs: q=23, g=9, x=4, y=3;

Excepted output: secret=9, hA=4, hB=16.

Sample form:

