Student ID:22101071d

Student Name: Zhu Jin Shun

Question 1: Functional Dependency and Normalization

a)

Top FD:

PlantNo → RecorderPoint, QtyOnHand

ItemNo, PlantNo → ItemDesc

OrderNo→ ShipAddr,OrderDate,CustNo

ItemNo, OrderNo→LineNo, QtyOrdered, QtyOustanding

CustNo → CustBal, CustDiscount

Bottom FD:

LineNo, OrderNo→ ItemNo, QtyOrdered, QtyOustanding

ItemNo, OrderNo → LineNo

b)

Underline→Primary Key

OrderNo	ItemNo	PlantNo	LineNo	Order date	ShipAddr	QtyOrdered	QtyOutstanding	CustNo	ItemDesc	QtyOnHand	ReorderPoint	CustBal	CustDiscount
O001	I001	P001	L001	2024/10/5	PQ306	10	5	C001	IA	50	RP01	1000	5%
O001	I002	P001	L002	2024/10/5	PQ306	5	2	C001	IB	30	RP01	1000	5%
O002	I001	P002	L001	2024/10/6	PQ307	20	10	C002	IA	60	RP02	1500	10%
O002	I002	P002	L002	2024/10/6	PQ307	15	15	C002	IC	40	RP02	1500	10%

c)

2NF Table:

OrderNo → ShipAddr, OrderDate, CustNo

OrderNo	ShipAddr	OrderDate	CustNo

ItemNo, OrderNo→LineNo, QtyOrdered, QtyOustanding

ItemNo	OrderNo	LineNo	QtyOrdered	QtyOustanding

ItemNo → ItemDesc

ItemNo	ItemDesc

LineNo, OrderNo→ ItemNo, QtyOrdered, QtyOustanding

LineNo	OrderNo	ItemNo	QtyOrdered	QtyOustanding

PlantNo → RecorderPoint， QtyOnHand

PlantNo	RecorderPoint	QtyOnHand

CustNo → CustBal, CustDiscount

CustNo	CustBal	CustDiscount

d)

3NF Table

OrderNo → ShipAddr, OrderDate, CustNo，CustBal, CustDiscount

OrderNo	ShipAddr	OrderDate	CustNo	CustBal	CustDiscount

OrderNo, ItemNo, LineNo → QtyOrdered, QtyOutstanding

OrderNo	ItemNo	LineNo	QtyOrdered	QtyOutstanding

ItemNo → ItemDesc

ItemNo	ItemDesc

PlantNo → RecorderPoint, QtyOnHand

PlantNo	RecorderPoint	QtyOnHand

e)

Possible BCNF Tables:

OrderNo → ShipAddr, OrderDate, CustNo， CustBal， CustDiscount

OrderNo	ShipAddr	OrderDate	CustNo	CustBal	CustDiscount

OrderNo, ItemNo , LineNo → QtyOrdered, QtyOutstanding

OrderNo	ItemNo	LineNo	QtyOrdered	QtyOutstanding

ItemNo → ItemDesc

ItemNo	ItemDesc

PlantNo → RecorderPoint, QtyOnHand

PlantNo	RecorderPoint	QtyOnHand

f)

ShipAddr → CustNo， CustBal,CustDiscount

PlantNo → RecorderPoint, QtyOnHand

ItemNo, PlantNo → ItemDesc

OrderNo→ ShipAddr,OrderDate

ItemNo, OrderNo→LineNo, QtyOrdered, QtyOustanding

Bottom FD:

LineNo, OrderNo→ ItemNo, QtyOrdered, QtyOustanding

I won't say it reasonable, as ShipAddr won't be unique for every customer, customer from same family will have same ShipAddr but different CustNo. If ShipAddr determines CustNo, customer from the same family will all have same CustNo which won't be accurate and unique for the relationship design process. Also, as OrderNo defines CustNo previously, changing to let ShipAddr defining CustNo won't change the original relationship as OrderNo still determines ShipAddr, which means OrderNo can still determine CustNo. So having ShipAddr to determine CustNo I would say isn't reasonable.

Question 2: Securing Application Passwords

a)

Symmetric encryption can be to store passwords through the two-tiered key based architecture. The encryption key is responsible for encrypting user passwords before they are stored in the database, ensuring that only encrypted versions are saved, and plaintext passwords are never exposed. The master key is stored and managed by an external security module and is used to encrypt the encryption key itself. So even the attacker can have access with the encryption key, the can't unlock it because they don't have the master key, meanwhile, the users will still have access to the master key and could unlock the encryption key with their own passwords.

The storage of Relevant Keys:

The TDE tablespace encryption Key is encrypted and can only be encrypted by the TDE master encryption key.

The TDE master encryption key's information and encrypted values of column encryption will be obtained in an external security module like the TPM or HSM.

b)

This method can be immune to dictionary attacks.

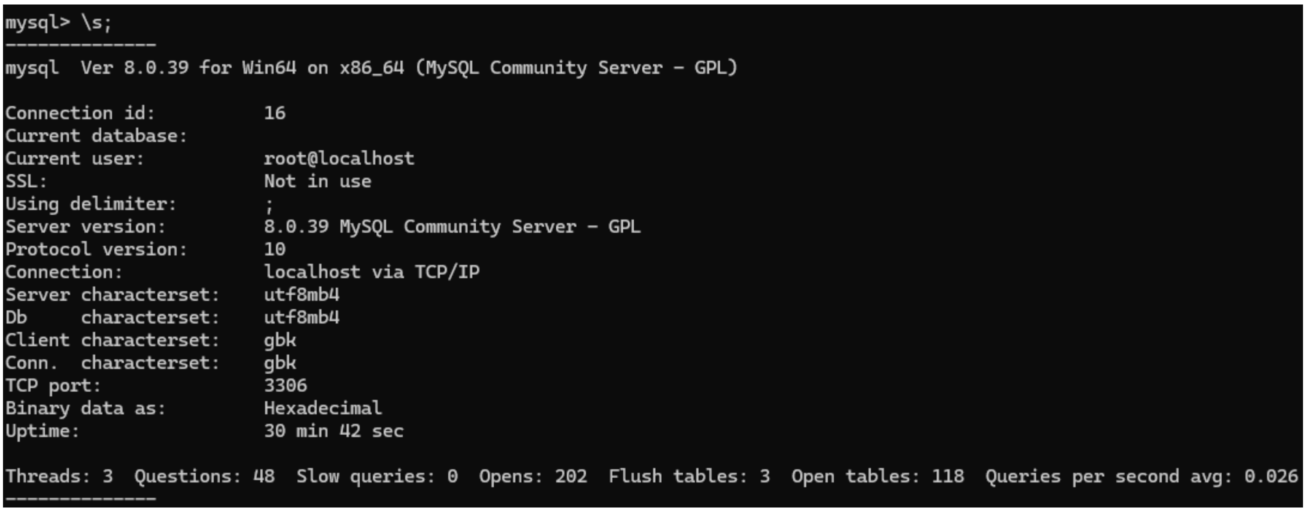
The reason is because this method ensures the passwords are encrypted on client side before sending to database, so the server never sees the plaintext data. Only the client application has access to encryption keys which can encrypt or decrypt data, so when attackers perform dictionary attack, they can only see encrypted values instead of accessing plaintext passwords or passwords hashes.

Question 3: Securing Data-in-motion in MySQL

a)

SSL: Not in use

Server version: 8.0.39 MySQL Community Server - GPL

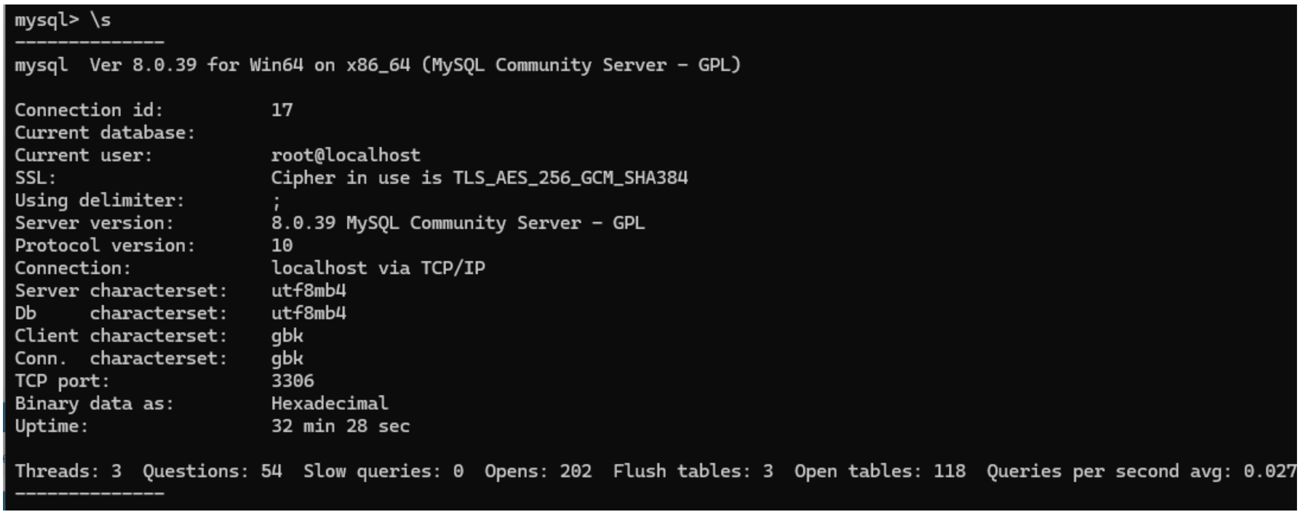


```
mysql> \s;
--------------
mysql  Ver 8.0.39 for Win64 on x86_64 (MySQL Community Server - GPL)

Connection id:          16
Current database:
Current user:           root@localhost
SSL:                    Not in use
Using delimiter:        ;
Server version:         8.0.39 MySQL Community Server - GPL
Protocol version:       10
Connection:             localhost via TCP/IP
Server characterset:    utf8mb4
Db     characterset:    utf8mb4
Client characterset:    gbk
Conn.  characterset:    gbk
TCP port:               3306
Binary data as:         Hexadecimal
Uptime:                 30 min 42 sec

Threads: 3  Questions: 48  Slow queries: 0  Opens: 202  Flush tables: 3  Open tables: 118  Queries per second avg: 0.026
--------------
```

b)

I chose mode REQUIRED

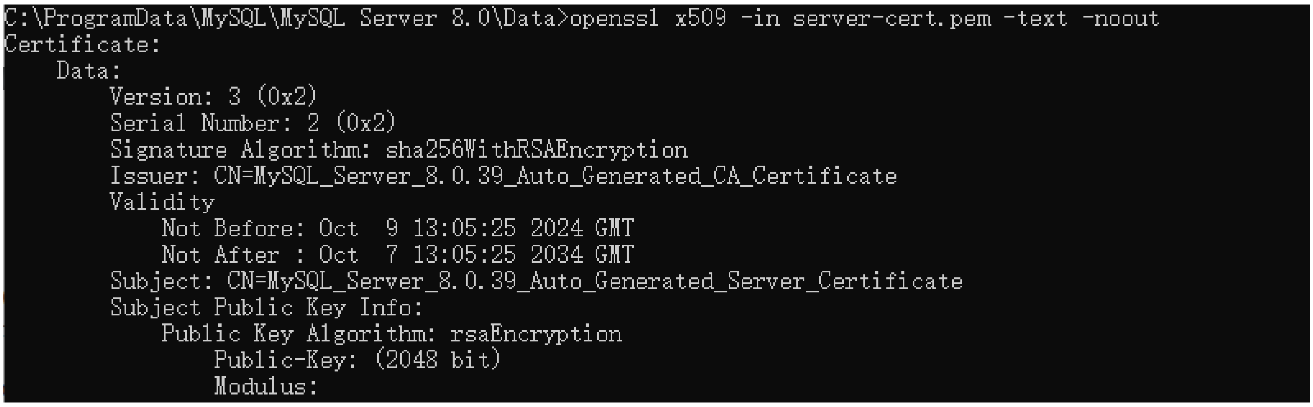
SSL: Cipher in use is TLS_AES_256_GCM_SHA384



```
mysql> \s
--------------
mysql  Ver 8.0.39 for Win64 on x86_64 (MySQL Community Server - GPL)

Connection id:          17
Current database:
Current user:           root@localhost
SSL:                    Cipher in use is TLS_AES_256_GCM_SHA384
Using delimiter:        ;
Server version:         8.0.39 MySQL Community Server - GPL
Protocol version:       10
Connection:             localhost via TCP/IP
Server characterset:    utf8mb4
Db     characterset:    utf8mb4
Client characterset:    gbk
Conn.  characterset:    gbk
TCP port:               3306
Binary data as:         Hexadecimal
Uptime:                 32 min 28 sec

Threads: 3  Questions: 54  Slow queries: 0  Opens: 202  Flush tables: 3  Open tables: 118  Queries per second avg: 0.027
--------------
```

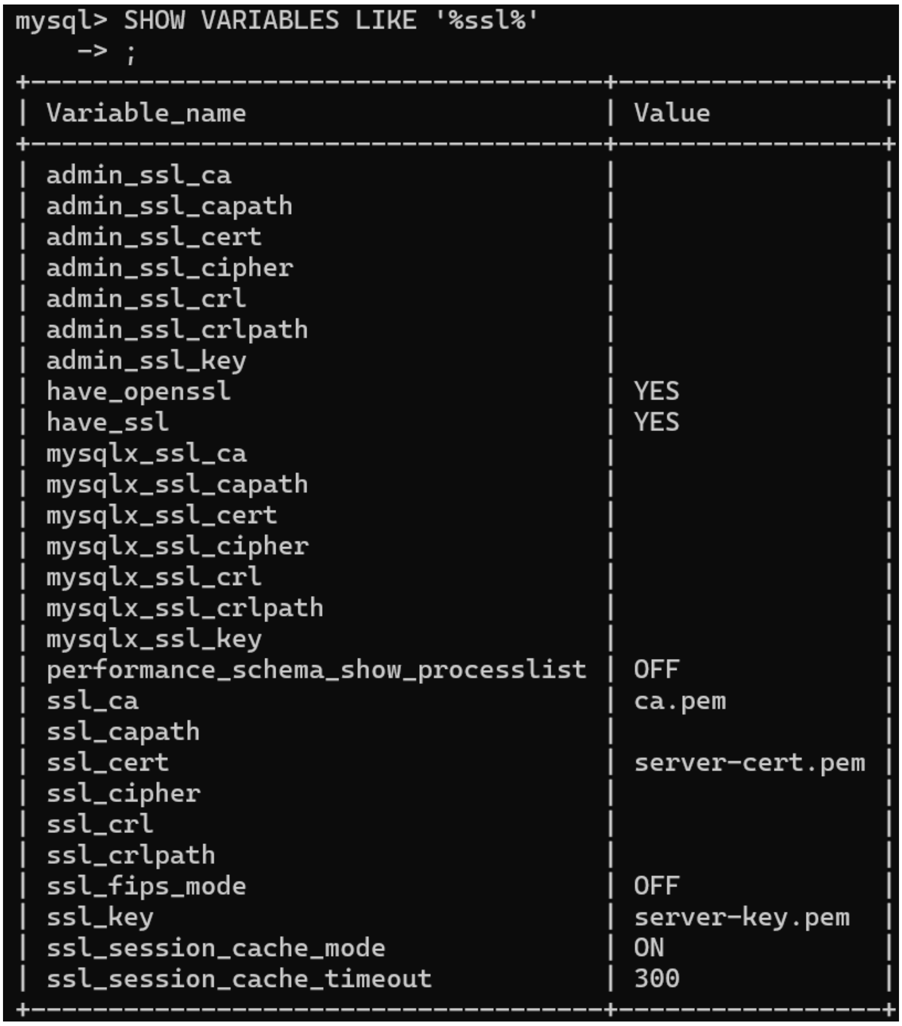
c)

Issuer: CN=MySQL_Server_8.0.39_Auto_Generated_CA_Certificate



```
C:\ProgramData\MySQL\MySQL Server 8.0\Data>openssl x509 -in server-cert.pem -text -noout
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 2 (0x2)
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: CN=MySQL_Server_8.0.39_Auto_Generated_CA_Certificate
        Validity
            Not Before: Oct  9 13:05:25 2024 GMT
            Not After : Oct  7 13:05:25 2034 GMT
        Subject: CN=MySQL_Server_8.0.39_Auto_Generated_Server_Certificate
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
```

Certificate Screenshot



```
mysql> SHOW VARIABLES LIKE '%ssl%'
    -> ;
+-------------------------------------+-----------------+
| Variable_name                       | Value           |
+-------------------------------------+-----------------+
| admin_ssl_ca                        |                 |
| admin_ssl_capath                    |                 |
| admin_ssl_cert                      |                 |
| admin_ssl_cipher                    |                 |
| admin_ssl_crl                       |                 |
| admin_ssl_crlpath                   |                 |
| admin_ssl_key                       |                 |
| have_openssl                        | YES             |
| have_ssl                            | YES             |
| mysqlx_ssl_ca                       |                 |
| mysqlx_ssl_capath                   |                 |
| mysqlx_ssl_cert                     |                 |
| mysqlx_ssl_cipher                   |                 |
| mysqlx_ssl_crl                      |                 |
| mysqlx_ssl_crlpath                  |                 |
| mysqlx_ssl_key                      |                 |
| performance_schema_show_processlist | OFF             |
| ssl_ca                              | ca.pem          |
| ssl_capath                          |                 |
| ssl_cert                            | server-cert.pem |
| ssl_cipher                          |                 |
| ssl_crl                             |                 |
| ssl_crlpath                         |                 |
| ssl_fips_mode                       | OFF             |
| ssl_key                             | server-key.pem  |
| ssl_session_cache_mode              | ON              |
| ssl_session_cache_timeout           | 300             |
+-------------------------------------+-----------------+
```

SHOW VARIABLES LIKE '%ssl%' screenshot

d)

Command 1:

(CA):

```
openssl ecparam -out C:\myCA\private\ca_private_key.pem -name prime256v1 -genkey
```

(Server 1: Localhost):

```
openssl ecparam -out C:\myCA\private\localhost_private_key.pem -name prime256v1 -genkey
```

(Server 2: IAmAHacker.com):

```
openssl ecparam -out C:\myCA\private\IAmAHacker.com_private_key.pem -name prime256v1 -genkey
```

Command 2

(Server 1: Localhost):

```
openssl req -new -key C:\myCA\private\localhost_private_key.pem -out C:\myCA\localhost.csr -config C:\myCA\private\openssl.cnf
```

(Server 2: IAmAHacker.com):

```
openssl req -new -key C:\myCA\private\IAmAHacker.com_private_key.pem -out C:\myCA\IAmAHacker.csr -config C:\myCA\private\openssl.cnf
```

Command 3:

(CA certificate): openssl req -x509 -new -key C:\myCA\private\ca_private_key.pem -out C:\myCA\ca_certificate.pem -days 365 -config C:\myCA\private\openssl.cnf

Command 4:

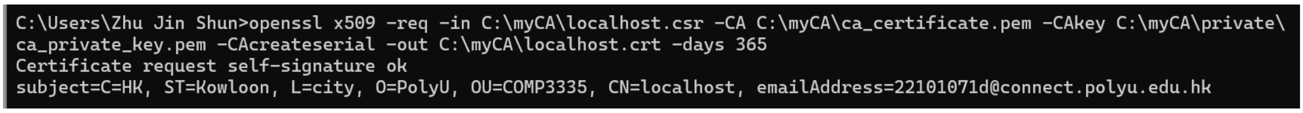
(Server 1: Localhost):

openssl x509 -req -in C:\myCA\localhost.csr -CA C:\myCA\ca_certificate.pem -CAkey C:\myCA\private\ca_private_key.pem -CAcreateserial -out C:\myCA\localhost.crt -days 365

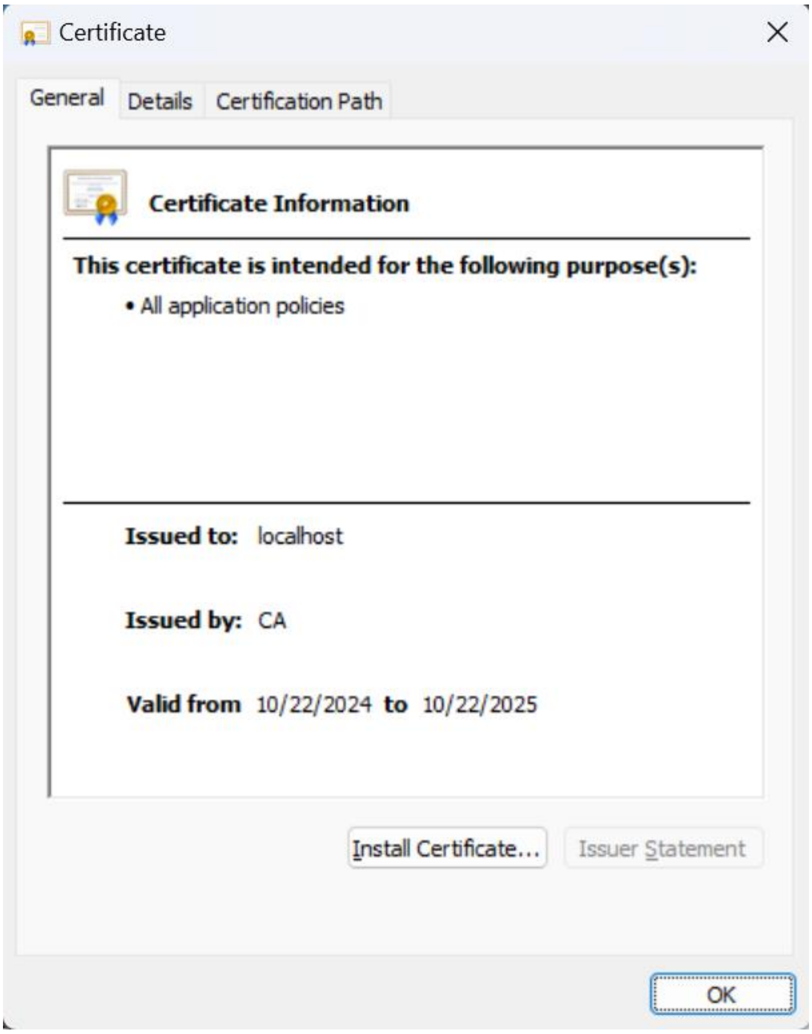
(Server 2: IAmAHacker.com):

openssl x509 -req -in C:\myCA\IAmAHacker.csr -CA C:\myCA\ca_certificate.pem -CAkey C:\myCA\private\ca_private_key.pem -CAcreateserial -out C:\myCA\IAmAHacker.crt -days 365

Certificate 1:



```
C:\Users\Zhu Jin Shun>openssl x509 -req -in C:\myCA\localhost.csr -CA C:\myCA\ca_certificate.pem -CAkey C:\myCA\private\
ca_private_key.pem -CAcreateserial -out C:\myCA\localhost.crt -days 365
Certificate request self-signature ok
subject=C=HK, ST=Kowloon, L=city, O=PolyU, OU=COMP3335, CN=localhost, emailAddress=22101071d@connect.polyu.edu.hk
```



Certificate Information

This certificate is intended for the following purpose(s):

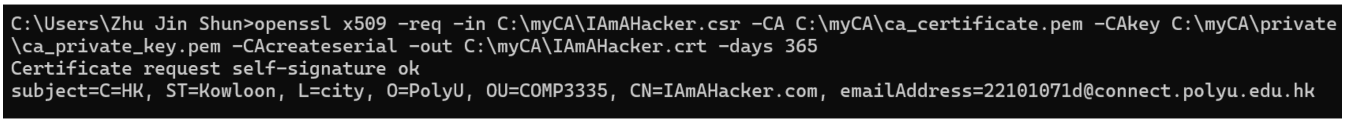
- All application policies

Issued to: localhost

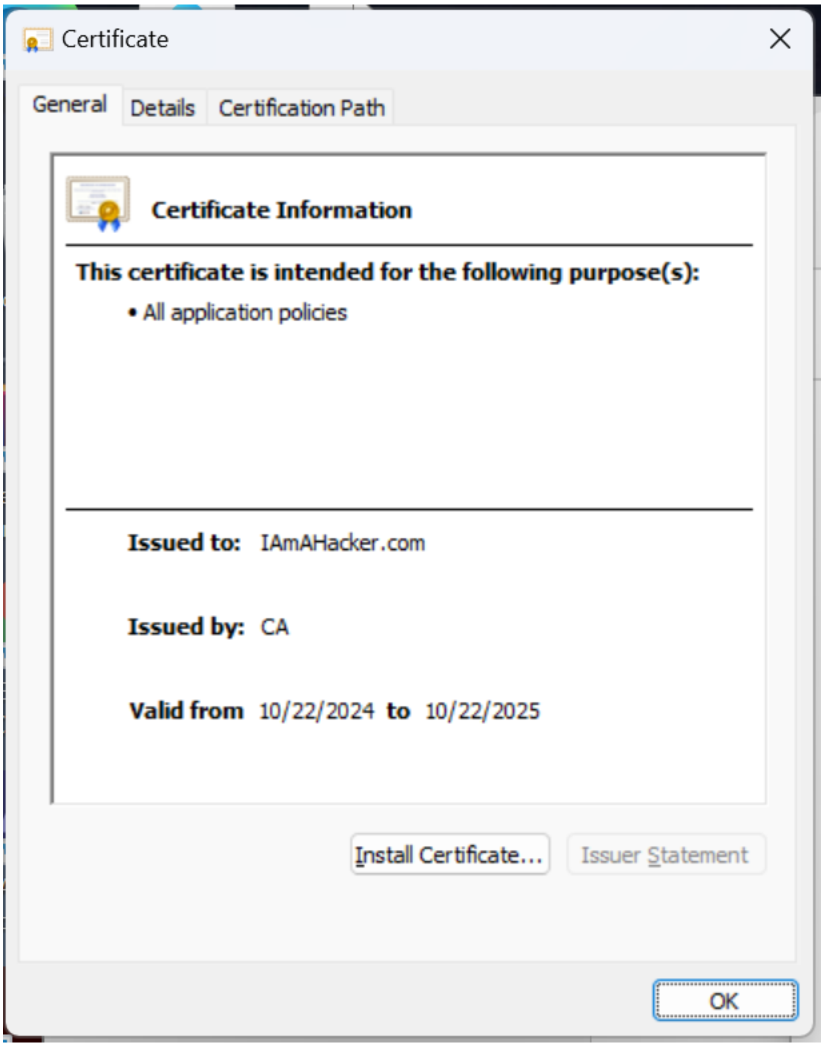
Issued by: CA

Valid from 10/22/2024 **to** 10/22/2025

Certificate 2:



```
C:\Users\Zhu Jin Shun>openssl x509 -req -in C:\myCA\IAmAHacker.csr -CA C:\myCA\ca_certificate.pem -CAkey C:\myCA\private
\ca_private_key.pem -CAcreateserial -out C:\myCA\IAmAHacker.crt -days 365
Certificate request self-signature ok
subject=C=HK, ST=Kowloon, L=city, O=PolyU, OU=COMP3335, CN=IAmAHacker.com, emailAddress=22101071d@connect.polyu.edu.hk
```



Certificate

Certificate Information

This certificate is intended for the following purpose(s):

- All application policies

Issued to: IAmAHacker.com

Issued by: CA

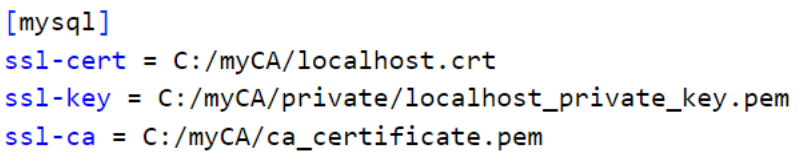
Valid from 10/22/2024 to 10/22/2025

e)

Lines Added to mysqld:

```
[mysqld]
ssl_cert = C:/myCA/localhost.crt
ssl_key = C:/myCA/private/localhost_private_key.pem
ssl_ca = C:/myCA/ca_certificate.pem
require_secure_transport = ON
```

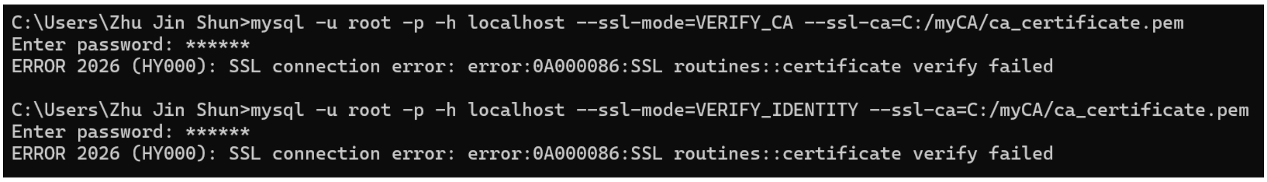
Lines Added to mysql:



```
[mysql]
ssl-cert = C:/myCA/localhost.crt
ssl-key = C:/myCA/private/localhost_private_key.pem
ssl-ca = C:/myCA/ca_certificate.pem
```

f)

All two connections failed due to ERROR 2026 (HY000): SSL connection error: error:0A000086:SSL routines::certificate verify failed.



```
C:\Users\Zhu Jin Shun>mysql -u root -p -h localhost --ssl-mode=VERIFY_CA --ssl-ca=C:/myCA/ca_certificate.pem
Enter password: ******
ERROR 2026 (HY000): SSL connection error: error:0A000086:SSL routines::certificate verify failed

C:\Users\Zhu Jin Shun>mysql -u root -p -h localhost --ssl-mode=VERIFY_IDENTITY --ssl-ca=C:/myCA/ca_certificate.pem
Enter password: ******
ERROR 2026 (HY000): SSL connection error: error:0A000086:SSL routines::certificate verify failed
```

The --ssl-mode setting plays an important role in encrypted connections in MySQL servers, it determines the level of verification on the SSL certificate, which is improving the security level of data transmission.

The effects caused on the verification can be different due to the mode chosen by the user. Stricter mode like the VERIFY_IDENTITY and VERIFY_CA provides higher security level of protection than DISABLED mode. Higher security mode can ensure the verification of the certificate through a recognized CA and ensuring that the hostname matches with the certificate details to prevent attacks.