

Project

by Salman Ahmed Khan

Submission date: 10-Dec-2022 11:59PM (UTC+0500)

Submission ID: 1977440021

File name: anagement_implementing_Confidentiality_and_Integrity_Report.docx (1.35M)

Word count: 1524

Character count: 8163

Record Management implementing Confidentiality and Integrity

Information Security (IS)

Presented to:

Dr. Fahad Samad

By:

Salman Ahmed Khan 19K-1043

Dated: December 10, 2022

Table of Contents

Abstract.....	3
Introduction.....	3
Existing System	4
1.1 System Architecture.....	4
Related Work/System	5
Adversary Model	6
System Design	6
System Implementation	7
I. Asymmetric Keys.....	7
II. Certificate Request & Digital Certificate.....	9
III. Certificate Authority Keys.....	10
IV. Confidentiality Tokenization Authentication.....	11
V. Add Student Record.....	11
VI. Edit Student Record.....	12
VII. View Student Record	14
System Evaluation	14
Technology	14
Conclusion	14

Abstract

The record management system is a daily activity carried out in various organization including schools, college, universities, companies, factories etc. Due to large number of user's details which contains sensitive information, it is difficult to make it in a secure fashion so that no one other than the authorized user of the system should carry out the amendments in the system. In order to tackle the physical record management system, I developed an automated record management system of the students in a university that ensure the integrity and confidentiality in the system. The integrity is maintained by using digital certificate authorized and issued by Certificate Authority (CA) upon providing Digital Certificate Request (CSR), creating digital signature of the data that is required to be stored in the database and the confidentiality is maintained by authenticating the user, validating the user details and maintaining its token in the system.

Introduction

A user details is an essential element that needs to be validated to prevent attack, manipulating and breaching of the data. The physical record management system is a tedious job that is carried out daily in the schools, universities etc. To ensure the system works perfectly, I will perform confidentiality and integrity.

The system will allow users to register within the system through which we will generate its public and private keys at runtime.

The user will be able to register with the system, keys will be generated and a private key will be provided to the user in hashed form on its email to perform digital signature from the browser/client.

The confidentiality will be maintained through authenticating user, validating user id and tokenization through which the system will allow the operations right to the user. There will exist a Certificate Authority (CA) that will issue a certificate to the user through which the system will maintain the confidence on the user. When the user wants to Edit the Record, the user will create a digital signature, digital certificate and data to the application server. The application server will then verify the certificate, obtain the public key from it and then verify the digital signature. After doing this, it will update the student record.

This system is built to secure the data of the users of the organization and by maintaining the confidentiality and integrity in the system.

Existing System

1.1 System Architecture

The system is built using Model View Controller (MVC) architecture. The system consists of three main entities server that include Client Server, Application Server and CA Server. The below system diagram displays the communication between these entities.

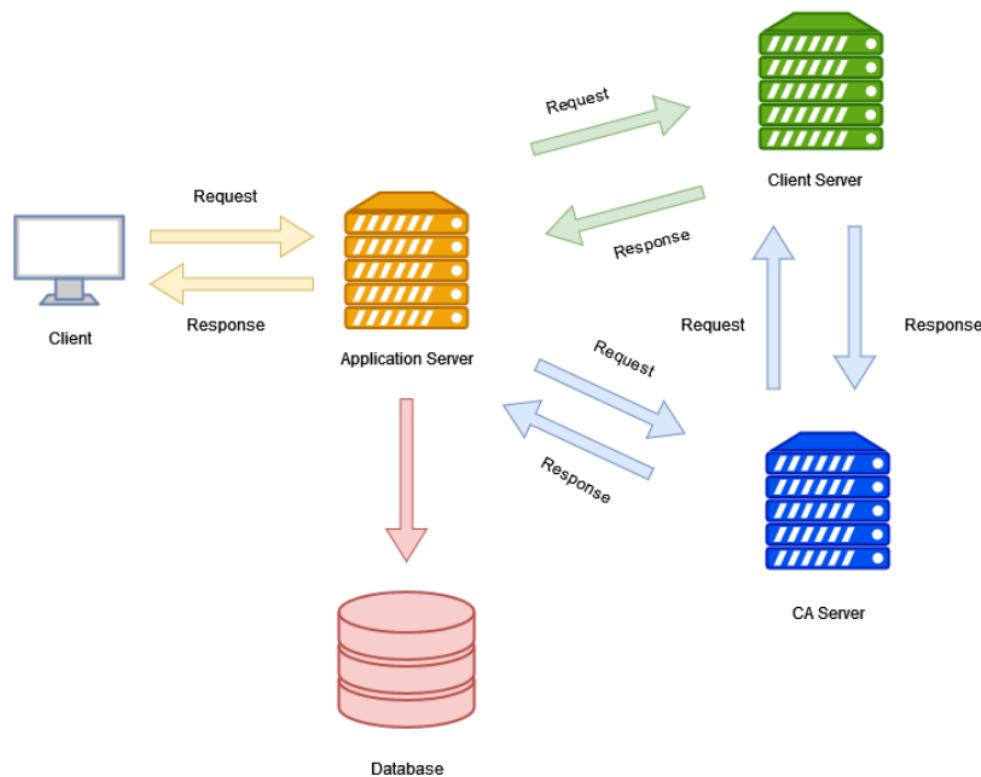


Figure 1 System Architecture

The responsibility of the Application Server is to handle all the request from the users. It also stores all the information in the database. The Application Server will pass the request of the creation of Asymmetric key to the Client Server. The Client Server is also responsible for creating the Certificate Request (CSR) and provide this request to the Certificate Authority (CA). The CA's Server will issue a Certificate to the Client Server. The Client Server will create a Digital Signature. It will provide the Digital Signature and Digital Certificate to the Application Server. The Application Server will verify the Digital Signature provided by the Client Server and upon verification, it will then obtain the public key and verify the Digital Signature. After verification, it will then update the record.

Related Work/System

The related record management does not implement confidentiality and integrity in the system. The client request is handled no security measures are taken to prevent the data access and data manipulation by third party. The data is being displayed to the user and all the data manipulation is carried out by just ensuring the login details of the user. Due to no security layer, if the user's credentials are breached or stolen then anyone with the credential can access the system and manipulate the data. Below is the attached architecture figure of such system.

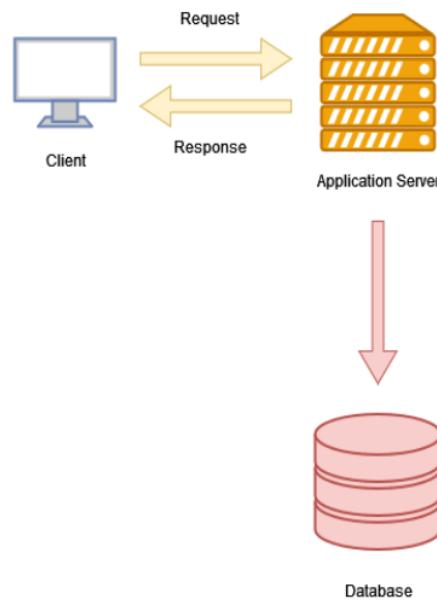


Figure 2 Related System

Adversary Model

The system model as describe in the section Existing System.

System Design

The System is designed using Model View Controller (MVC) pattern. Also, distributed system architecture is used to build the system. As per the security design is concerned, the user's password is hashed with generated salt to make secure password. The tokenization is done to maintain the user's session. The Public and Private keys are created to perform the digital signature. The Private key is then hashed and mailed to the user to perform certain actions. A Certificate Authority is used to issue the Digital Certificate to the user. Both the digital certificate and digital signature are used to maintain the integrity in the system.

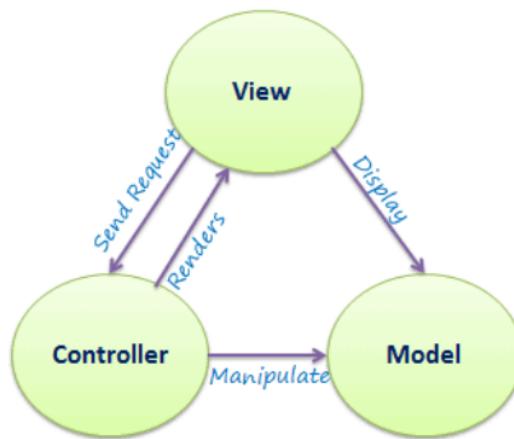


Figure 3 Software Architecture

Cookie Based Tokenization for Confidentiality contains the user's id as the data payload.

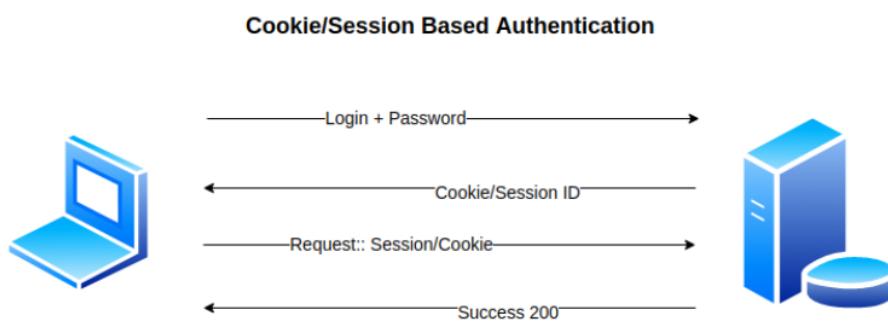


Figure 4 Tokenization Authentication

The user's password is stored in a hashed form in the database.

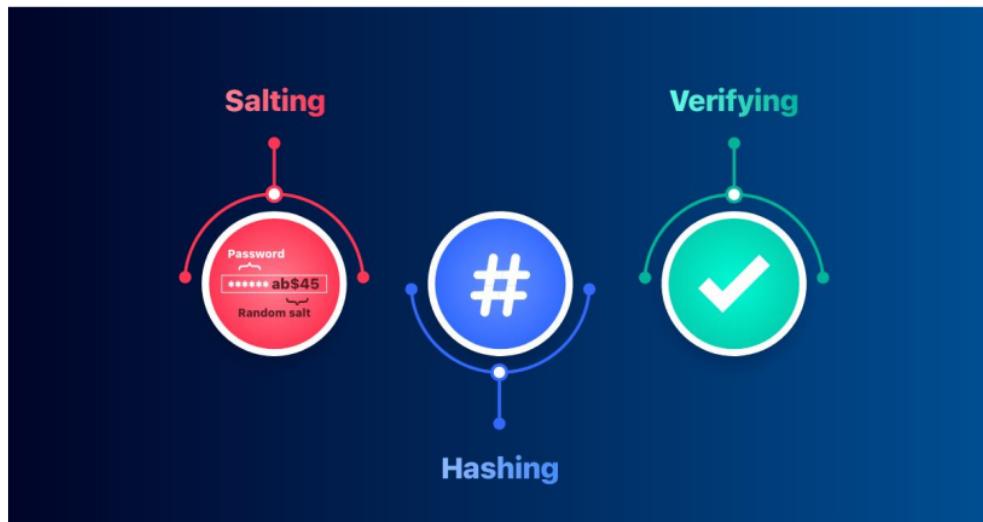


Figure 5 Salt Hashed Password

System Implementation

The features of the system are:

- Signup
- Login
- Add Student Details
- View Students Details
- Edit Student Record

The system focuses on a single entity named admin. The user will signup the system. After signing in the system, the Asymmetric Keys that are RSA Public and RSA Private Key is created on the Client Server.

I. Asymmetric Keys

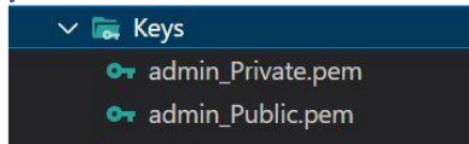


Figure 6 RSA Asymmetric Keys

The Public Key of the user.

```
Backend > Keys > admin_Public.pem
1 -----BEGIN RSA PUBLIC KEY-----
2 MIIBCgKCAQEAvwYON1GOBtdcAA71U43F14o/zvKom+4vW8/dIUh/Xexvg0LQA5n/
3 KH8qGpXbzVc8yDxjY1VIuFTn15I1PnsZwoV+wlospdrZupzC7gw1FBByxaKxUO6m+
4 EURxINM0udLvTuuhmh+6qJvTPy69khVZcrcCH4y1vdCpNiWTZuIjSA5Cnp0hZqC/
5 K2eFQAdf4JLKZ+5BeI+c4s+Q37MxFIttE64N6oFD88ARSoLJq7fsLLCPLrstbeYS
6 eMi1RJ001sBpq8dIRnnxw1X00E8e4l1lx+LR1IT2t+XOTXFaywlcBeLyBHFA/p1
7 NdZsuEhpLQgJjue8Ng7LOGBqc7f1d4tslwIDAQAB
8 -----END RSA PUBLIC KEY-----
9
```

Figure 7 User Public Key

The Private Key of the user.

```
Backend > Keys > admin_Private.pem
1 -----BEGIN RSA PRIVATE KEY-----
2 MIIEogIBAAKCAQEAvwYON1GOBtdcAA71U43F14o/zvKom+4vW8/dIUh/Xexvg0LQ
3 A5n/KH8qGpXbzVc8yDxjY1VIuFTn15I1PnsZwoV+wlospdrZupzC7gw1FBByxaKxU
4 O6m+EURxINM0udLvTuuhmh+6qJvTPy69khVZcrcCH4y1vdCpNiWTZuIjSA5Cnp0h
5 ZqC/K2eFQAdf4JLKZ+5BeI+c4s+Q37MxFIttE64N6oFD88ARSoLJq7fsLLCPLrst
6 beYSem1RJ001sBpq8dIRnnxw1X00E8e4l1lx+LR1IT2t+XOTXFaywlcBeLyBH
7 A/p1NdZsuEhpLQgJjue8Ng7LOGBqc7f1d4tslwIDAQABoIBACCv+hWcbgfIqWMP
8 KXnbPDEaxKF2j5ijMPJuHuHvpG/fihN1ht/e5bU50lwcmBZEZGPQR1cAzgFQ8hJT
9 QA4xhW5uZorVTs+QvAo9UFPl/mILKOaG+qD9Alu1EQCzc4XKzQ19r7kObep1Sjg5
10 GNJbGePR4iig70a0wq3Tzz3f03dUQPsc4W8sUZZ01qPC7hYFPz6b1ZBUq4B45huD
11 lgujylUscFsXtbVGmrJ4YKyyLNVYDKKf097XgvCz+I2hLHpJjqLV6toSnbbypNEM
12 wsJa3tQoQCqxBIKvUNq1+UTywLl1l2/EaTU6qtU1X5ipMyg40YvmLEFBueFjbHR+
13 NtZsVp0CgYEAwQYrWlxDu3qfN8LLXvKzaMcKpmwqWyVJyH6JKTdwley8ZqWQ+rT
14 yEzwrK6nu+eD1LDB1jH71G61Mx+sPlqHxB2Bo4R4xte4wkW4PdR8zfnUcBBdK85e
15 Jo8sTekqUgE4tfBkEPpHLouKdE2Y7ixwlxiYc6JJTCxeHdLb/CQEhi8CgYEAVjN
16 SayDang5QFKV8Q3TqOkYqhEtV7ICmf+tiZoan1lsm/l3lrxLvsjq4iicbpXCjam
17 aIK62D/EywHtVqcul8ggmf6uV7TmSHOBpHS1//sX2791EiEy7gRx/pCvPdx+JeGE
18 PkRkg5XrHgwXvbfbKRgAGBrjciB0pAOnOSYqHZUCgYBM83evE19RS4ZUBY1sMdeC
19 au98/1w3b/rk5e/CUglU213VW+qR8Gwv0TSVsx9dku0E1F8CVHTWbPhCBFt31EHR
20 1m2LUDGQfAnFMWRNCfwjZXjioDN0tlXki21/OfhMLv57LrTHa++ax7iPuyu/r+md
21 bPHCD62m930h6KEYuQcwsWkBgbosuaEYEL/IRQr79iVZNWmfiKATgH3InBxOdFv3
22 XeRark4gn3ygorhP6UCIr1fxq7R3bvvvPcSn2forac0BGtQ9rfibEoInQ3+kx4Mp
23 Nh1bZNykTEEiQvPKa8ikRCa9CB97zEDuqf/aV19VWoKUYYaX7Ad3vhNFeTqC0h0a
24 b1F9AoGARHko1/tvIXUnJvzBOX0U9aKn//wMRPeeZ1PC2eNyCaDX/N2ymElH1dw9
25 TUXNIhUYE4iHCvdVx5glPbgufHxvd51nv3+WC50jzhfUaWZiOHDZ7mPKXgKVsmsP
26 fghbWyIxvsr4o+A817LGFmyZE/U4V/KUE2Cm89nAZgrEly61RKU=
27 -----END RSA PRIVATE KEY-----
```

Figure 8 User Private Key

After the creation of Asymmetric Keys, an automated mail is done to the user email containing the user's hashed private key used in the operation of Edit Student Record to create the digital signature as shown below.

Private Key for Signature

 k191043@nu.edu.pk <k191043@nu.edu.pk>
20/11/2022 7:29 pm
To: salman.ahmed979@outlook.com

\$2b\$10\$e9dzwzsQENQS4uyNKXbcdeF204LFm7qUXvgOKZx7Z.gQGcklEUHe6

Figure 9 Email Hashed Private Key

II. Certificate Request & Digital Certificate

The Certificate Request (CSR) is generated after the creation of Asymmetric Key Pair. The CSR is used to obtain a Digital Certificate from the Certificate Authority (CA). The CSR is displayed below.

```
Backend > Certificate > admin_CSR.pem
1 -----BEGIN CERTIFICATE REQUEST-----
2 MIICrjCCAZYCAQAwTEVMBMGA1UEAxMMQ2xpZW50X2FkbWluMQswCQYDVQQGEwJQ
3 SzEOMAwGA1UECBMFU2luZGgxEzARBgNVBAcTCkJsYWNrc2J1cmcxDzANBgNVBAoT
4 BkNsawVudDENMAAsGA1UECxMEVGVzdDCCASIwDQYJKoZIhvcNAQEBBQADggEPADCC
5 AQoCggEBAL8GDjZRjgbXXAA05VONxZeKP87yqJvuL1vP3SFIf13sb4NC0AOZ/yh/
6 KhqV281XPMg8Y2JVSLhU55eSNT57GcKFFsFqLKXa2bqcwu4FpRQcsWisVDupvhFE
7 cSDTNLnS707roZofuqib0z8uvZIVWXK3Ah+Mtb3QqTY1k2biI0gOQp6dIWagvytn
8 hUAHX+CSymfuQXiPnOLPkN+zMSH7bROuDeqBQ/PAEUqCyau37Cywjy67LW3mEnjI
9 tUSTtNbAaavHSEZ58cNV9DhPHuJYpcfI0ZSE9rfhzk1xQmssFnAXi8gRxQP6ZTXW
10 bLhIaS0ICY7nvDYOyzhgan039XeLbFsCAwEAAsAAMA0GCSqGSIb3DQEBBQUAA4IB
11 AQCP5o/a10pxsrMP3tcqWiS96QpWRUx9ulmczG6+nMVT31mn6Ss7LGL+WL9hplo+
12 1VoW67mcRyKQuSt25FKnJ9bSTuIoMG70BK676kZARL2LOZm5x0K6znEieV5D3/wQ
13 buX0k1LaHwbmOUTPSGmyZ50J7wQ6CEs/3uuZ/V+eYIxCeLKa1s3qouVAZT0AAk7Y
14 pEPKojT1yWSVE6asORB9PHVQmlf4SSZHGAaG5QagMiKoeM9JcaEbTrStF7fy+m5m
15 UZB/+SasU/anad4Hu7x+QSrC07UyHWqsuZ13AvalQsMpzJ4xne7dSoQd3r3dQE+2
16 5QMXnJixbxnmmtmEVRAJ1gUr
17 -----END CERTIFICATE REQUEST-----
```

Figure 10 Certificate Request

The Certificate Request (CSR) is then passed to the Certificate Authority (CA)'s Server to obtain the Digital Certificate. The user's issued Digital Certificate is shown below.

```
Backend > Certificate >  admin_Certificate.pem
1 -----BEGIN CERTIFICATE-----
2 MIIDTjCCAjaBgIBAgJANBgkqhkiG9w0BAQUFADBsmQswCQYDVQQDEwJDQTEM
3 MAoGA1UEBhMDUEFLMQ4wDAYDVQQIEwVTaW5kaDEPMA0GA1UEBxMGQ0FBdXRoMR8w
4 HQYDVQQKExZDZXJ0aWZpY2F0aW9uQXV0aG9yaXR5Mj0wCwYDVQQLEwRUZXN0MB4X
5 DTIyMTEyMDE0MjkxN1oXDTIzMTEyMDE0MjkxN1owaTEVMBMGA1UEAxMMQ2xpZW50
6 X2FkbWluMQswCQYDVQQGEwJQSzEOMAwGA1UECBMFU2luZGgxExARBgNVBACtCkJs
7 YWNrc2J1cmcxDzANBgNVBAoTBkNsawVudDENMasGA1UECxMEVGVzdDCCASIwDQYJ
8 KoZIhvcNAQEBBQAQDggEPADCCAQcggEBAL8GDjZRjgbXXAA05VONxZeKP87yqJvu
9 L1vP3SFIf13sb4NC0AOZ/yh/KhqV281XPMg8Y2JVSLSu55eSNT57GcKFFsFqLKKxa
10 2bqcwu4FpRQcsWisVDupvhFeCSDTNLnS707roZofuqib0z8uvZIVWXK3Ah+Mtb3Q
11 qTY1k2biI0gOQp6dIWagvytnhUAHX+CSymfuQXiPnOLPkN+zMSH7bROuDeqBQ/PA
12 EUqCyau37Cywy67LW3mEnjItUSTtNbAavHSEZ58cVN9DhPhuJYpcfi0ZSE9rf1
13 zk1xQmssFnAXi8gRxQP6ZTXbLhIaS0ICY7nvDYOyzghan039XeLbFsCAwEAATAN
14 BgkqhkiG9w0BAQUFAAOCAQEASAXyAHVWaNMUAEp0NtCqD8FAsmN0PcLNP/hhgWP
15 JydoRnX2Pjy2TVxctc7CgVyeKWB8lr+JDuOZCCDExbZPvWB3L2ZgfWjF3QDyhnaX
16 jKrf/1Q7v9G1uN1HkxGzVKAzFs5tiUTW97Gt2puLhFLoQNs1e6SkLwnk1ahBKqqR
17 WJlnK76EMTj8K2ISbzIEoFLGt0u6ZE6LdF+8o3f8AqBsADnHg3U/JdITZTCOSWdT
18 7658GH1s0qUmGi2EpuE1W3jKZHTl1j5oODyRKwW3sfZfb9mqhWccSoh/e4hlqxXm
19 Eui4hVjMiwJOx+MEoUIXwmFU2M9/u8EULk2S7kdTVKdFZw==
20 -----END CERTIFICATE-----
```

Figure 11 Digital Certificate of User

III. Certificate Authority Keys

The Certificate Authority (CA) has already created its Asymmetric Key Pair on its Server.

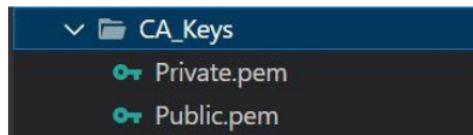
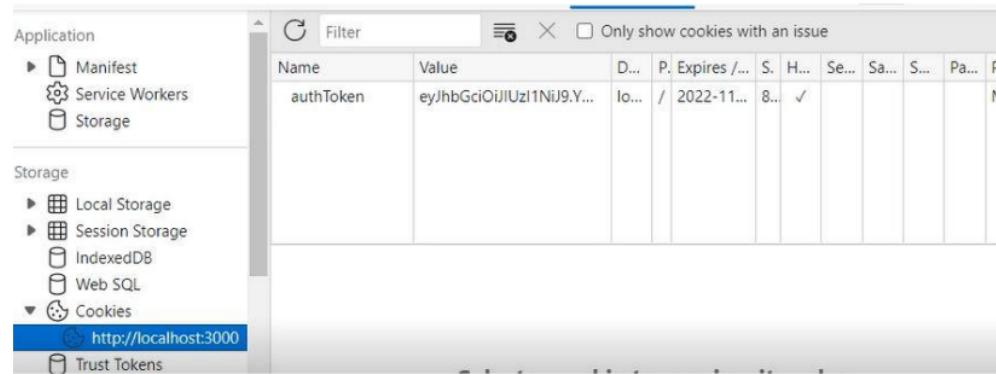


Figure 12 CA Asymmetric Key Pair

IV. Confidentiality Tokenization Authentication

After successful signup, the system redirects the user to the Login Page. The user provides its credentials and after the validation of its credentials, the authentication token is maintained in the cookie that provide system confidence on the user whenever a client request is generated, hence provide confidentiality.

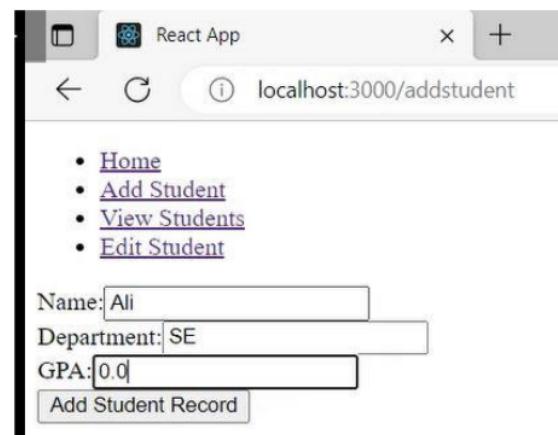


The screenshot shows the Chrome DevTools Network tab. On the left, there's a sidebar with sections for Application, Storage, and Cookies. Under Cookies, a single entry for 'authToken' is listed. The table has columns for Name, Value, D..., P..., Expires /..., S..., H..., Se..., Sa..., S..., Pa..., and F... . The value column shows a long JWT token: eyJhbGciOiJIUzI1NiJ9.Y... . The 'Expires /...' column shows '2022-11-11T10:42:00.000Z'. The 'S...' column has a checkmark. The 'F...' column shows 'N'. At the bottom of the sidebar, it says 'http://localhost:3000' and 'Trust Tokens'.

Figure 13 Authentication Token – Confidentiality

V. Add Student Record

Adding Student Record is shown below. The Application Server gets the student data and stored in the database.



The screenshot shows a web browser window titled 'React App' at 'localhost:3000/addstudent'. The address bar shows the URL. Below the address bar is a navigation bar with back, forward, and search buttons. The main content area contains a list of links: Home, Add Student, View Students, and Edit Student. Below this is a form with three input fields: 'Name:' containing 'Ali', 'Department:' containing 'SE', and 'GPA:' containing '0.0'. A button labeled 'Add Student Record' is at the bottom of the form.

Figure 14 Add Student Record

VI. Edit Student Record

The Edit Student Record require the hashed Private Key from the user in order to generate the Digital Signature which is carried out by the Client Server. If the Private Key mismatched with the original Private Key stored on the Client Server, it then provides the unauthorized message to the user and the further procedure is terminated. The Client Server provides the Digital Signature and Digital Certificate to the Application Server. The Application Server then obtain a Digital Certificate from the Certificate Authority (CA), creates a CA Store and verify the issued Digital Certificate of the client. After successful verification of the Digital Certificate, the Application Server then obtain the Public Key from the Digital Certificate of the user and verify the Digital Signature provided by the Client Server. Upon successful verification, the data is then updated in the database.

A screenshot of a web browser window titled "React App". The address bar shows "localhost:3000/editstudent". The page content includes a navigation menu with links to Home, Add Student, View Students, and Edit Student. Below the menu is a section titled "Edit Student Record". It contains three input fields: "SID: 1", "GPA: 2.2", and "Key:". At the bottom is a blue "Update Record" button.

Figure 15 Edit Student Record

A screenshot of a web browser window titled "React App". The address bar shows "localhost:3000/editstudent". The page content includes a navigation menu with links to Home, Add Student, View Students, and Edit Student. Below the menu is a section titled "Edit Student Record". It contains three input fields: "SID: sid...", "GPA: 4.0", and "Key: *****". At the bottom is a blue "Update Record" button. Below the form, a message "Recorded Updated Successfully" is displayed.

Figure 16 Edit Student Record Successful

```
const editStudentRecord = (req, res) => {
  // Digital Signature
  const { signature, certificate, ca_certificate, studentData } = req.body;

  const certificateVerify = forge.pki.certificateFromPem(certificate);

  let store = forge.pki.createCaStore([ca_certificate]);
  const verified = forge.pki.verifyCertificateChain(store, [certificateVerify]);

  if (!verified) {
    return res
      .status(401)
      .json({ message: "Certificate Verification Failed", status: "failed" });
  }
  console.log("Certificate Public Key\n", certificate.publicKey);
  console.log("\n\nCertificate Verify Public Key\n", forge.pki.publicKeyToPem( certificateVerify.publicKey));
  let publicKey = forge.pki.publicKeyToPem(certificateVerify.publicKey);
  console.log("PublicKey", publicKey);

  const data = Buffer.from(JSON.stringify(studentData));
  const verifySign = crypto.verify("SHA256", data, publicKey, Buffer.from(signature, "base64"));
  if (!verifySign) {
    return res
      .status(401)
      .json({
        message: "Digital Signature Verification Failed",
        status: "failed",
      });
  }
}

let sid = studentData.sid;
let gpa = studentData.gpa;

if (sid === "" || gpa === "") {
  return res
    .status(400)
    .json({ message: "Invalid Parameters", status: "failed" });

  let _sid = parseInt(sid);
  let _gpa = parseFloat(gpa);
  const sqlQuery = "UPDATE student_record SET gpa = ? WHERE sid = ?";
  db.query(sqlQuery, [_gpa, _sid], (err, result) => {
    if (err)
      return res
        .status(400)
        .json({ message: "Failed to add record", status: "failed" });
    if (result.affectedRows > 0)
      return res
        .status(201)
        .json({ message: "Recorded Updated Successfully", status: "success" });
  });
}
```

Figure 17 Edit Student Record - Digital Certificate & Digital Signature

VII. View Student Record

After the successful edit record, the data could be displayed in the view page.

The screenshot shows a web browser window titled "React App" with the URL "localhost:3000/viewstudents". The page displays a navigation menu with links to Home, Add Student, View Students, and Edit Student. Below the menu, the title "View Student Records" is centered. A button labeled "Click to Display Record" is present. A table with four columns (StuID, Name, Department, GPA) shows one row of data: StuID 1, Name Ali, Department SE, and GPA 2.2. A blue button labeled "View Records" is located to the left of the table, and a three-dot menu icon is to its right.

Figure 18 View Student Record

System Evaluation

The system is evaluated using API testing using Postman API Testing tool. Moreover, the after the running the system, we could evaluate the system performance.

Technology

- React JS (Frontend)
- Node JS + Express JS (Backend)
- Bcrypt (Hashing), Node-Forge (Digital Certificate + Certificate Request) + Crypto (Asymmetric Key Pair + Digital Signature)
- MySql (Database)

Conclusion

The secure record management is necessary in the organization that could maintain the integrity and confidentiality in the system and does not allow any illegal and unauthorized access of the system by any other party. Therefore, such a secure system is a must in an organization to protect their data from data breach and unauthorized data access.

Project

ORIGINALITY REPORT

2%
SIMILARITY INDEX

2%
INTERNET SOURCES

0%
PUBLICATIONS

2%
STUDENT PAPERS

PRIMARY SOURCES

1 digital.lib.usu.edu
Internet Source 2%

2 www.coursehero.com
Internet Source 1%

Exclude quotes Off

Exclude bibliography On

Exclude matches Off