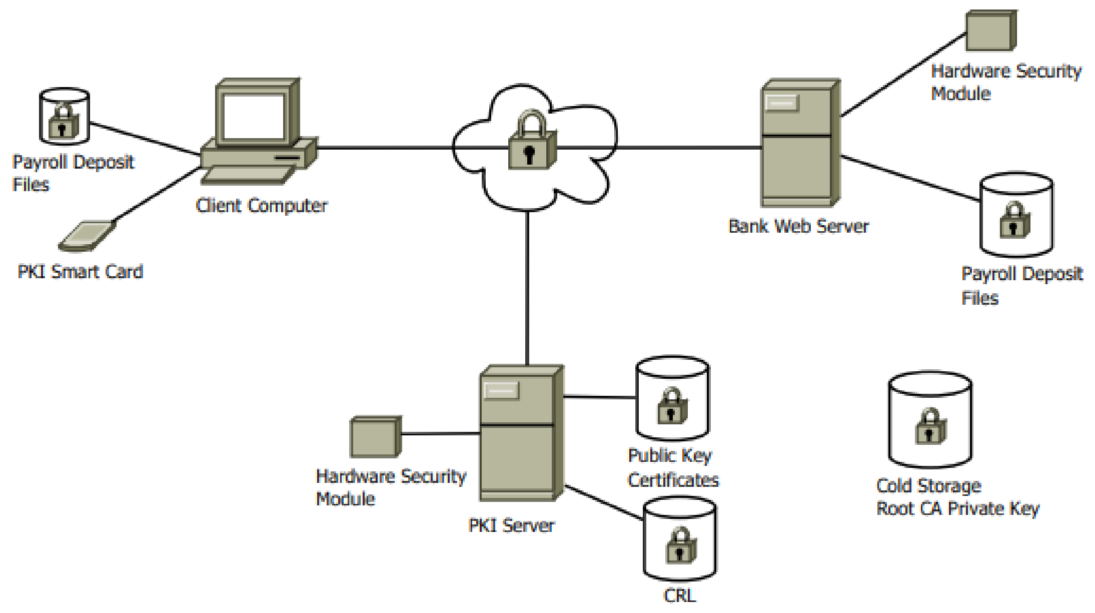
## Project Overview

This project is all about designing, documenting, building, and testing a secure payroll storage and transmission system that meets the up-to-date cyber security requirements of confidentiality, integrity, authenticity, and non-repudiation following approved NIST algorithms and standards. It involves creating a Root CA, allotment of CA certificates to clients and servers, secure transmission of documents and verification of signature and certificates, and confidentiality of documents at rest as well as in transit.

## Project Topology



CST8805 Project Topology

## Scenario

The client is a business organization that needs to send its payroll regularly to the bank. In order to accept the payroll document, the bank requires the client to be an authorized party certified by a trusted root authority which is also the same Root Authority for the bank's web server. On the other hand, for the client to trust the bank web server, the client wants the bank server to also be certified by the same trusted root authority. The client also requires that the bank web page ensures proper security of the data in transit. The certificates provided by the common Root CA should be valid when it is used for signature or verification, and it should contain the proper extension for the action executed by each certificate user.

## <u>Project Overview</u>

<u>System Components</u>

- **<u>Client Computer</u>**

  To ensure proper security, client's agent uses **PKI Smart Card** for secure signing/protection of public key/private key pair for signing Payroll Deposit files (**proposed but not implemented**). Smart Card is suitable for single user situation. Smart card technology contains a cryptographic module. This module facilitates multifactor authentication for the generation and security of public key infrastructure (PKI) keys and certificates that are used to authenticate operating systems and applications, sign documents, or encrypt data, such as files or emails. Users utilizing smart card technology insert a card or cryptographic USB token into a reader, and then enter the associated PIN. A key exchange then occurs between the operating system and an application to validate the certificate and associated keys. In our case it happens between the client and the bank web server application. The client's web browser is configured with the certificate authority's root certificate to ensure the client trusts the web server.

- **<u>Bank Web Server</u>**

  HTTPS has been implemented in the bank web server, with a certificate signed by the PKI server root certificate authority. The server forces an HTTPS connection to ensure no insecure communication can be conducted. TLS 1.3 is implemented using a cipher suite of NIST approved algorithms.

  HSM is implemented in the bank server for secure cryptographic operations and protection of web server. A hardware security module (HSM) is a dedicated crypto processor that is specifically designed for the protection of the crypto key lifecycle. Hardware security modules act as trust anchors that protect the cryptographic infrastructure of security-conscious organizations by securely managing, processing, and storing cryptographic keys inside a hardened, tamper-resistant device. Enterprises like PKI, financial institutes etc. buy hardware security modules to protect transactions, identities, and applications, as HSM excels at securing cryptographic keys and provisioning encryption, decryption, authentication, and digital signing services for a wide range of applications. HSM is suitable for multi-user situation like financial institute, PKI etc.

- **<u>PKI Server</u>**

  Because of the extreme security requirements, PKI Certificate and key preservation Servers are maintained as completely stand-alone isolated servers. The CA stores, issues, and signs digital certificates, and maintains and publishes a Certificate Revocation List (CRL). CRLs are stored in a separate encrypted storage. The PKI Server is also equipped with a HSM (Hardware Security Module).

Algorithms Used

- Standards used: **X.509 is an ITU standard used for certificates.**
- For private/public key generation/extraction: **RSA:2048bits approved by NIST SP800-175BR1.**
- Digital Signature: **Hash function SHA256 approved by NIST SP800-175BR1.**
- Encryption/Decryption of documents: **Proposed off the shelf solutions uses NIST algorithms.**
- Web page session encryption: **TLS 1.3**.
- Key exchange suites: **ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256**

## Project Implementation

The project tasks and desired requirements have been implemented through a series of steps which are described as follows:

1. **Creating a root certificate authority**

- **Generate a 2048-bit CA key with AES128CBC**

```
openssl genpkey -outform pem -algorithm rsa -pkeyopt rsa_keygen_bits:2048 -aes-128-cbc -pass
pass:cyber123 -out CAprivKey.key
```

- **Extract the Public Key**

```
openssl rsa -in CAprivKey.key -pubout -out CApubKey.key
```

- **Generate a self-signed certificate with a SHA256 signature**

```
openssl req -new -x509 -outform pem -sha256 -set_serial 0x100 -key CAprivKey.key -days 365 -out
CArootCert.cer
```

Root certificate information:

```
Country Name (2 letter code) [XX]:CA
State or Province Name (full name) []:Ontario
Locality Name (eg, city) [Default City]:Ottawa
Organization Name (eg, company) [Default Company Ltd]:SkyNet Corp
Organizational Unit Name (eg, section) []:Security
Common Name (eg, your name or your server's hostname) []:sara.skynet.ca
Email Address []:.
```

- **Distribute root certificate to users**

```
scp CArootCert.cer cyber_admin@10.10.1.11:/home/cyber_admin/
scp CArootCert.cer cyber_admin@10.10.1.10:/home/cyber_admin/
```

2. **Root CA creates a certificate revocation list**

- **Create the necessary files to make the revocation list**

  ```
  vi crlnumber -> enter "01"
  vi certserial -> enter "01"
  touch index.txt
  ```

- **Make an openssl config file and modify it**

  ```
  cp /etc/pki/tls/openssl.cnf /home/cyber_admin/Desktop/Certificates
  ```

  Modify the following lines:

  ```
  dir     = .
  certs   = $dir
  crl_dir = $dir
  ```

- **Generate CRL and format it in DER**

  ```
  openssl ca -config openssl.cnf -gencrl -keyfile CAprivKey.pem -cert CArootCert.cer -out
  RevokedCerts.crl.pem
  openssl crl -inform PEM -in RevokedCerts.crl.pem -outform DER -out RevokedCerts.crl
  ```

- **Publish CRL**

  ```
  cp RevokedCerts.crl /var/www/html/
  ```

3. **Client requests certificate for signing documents**

- **Generate client's 2048 bit signing key with AES128CBC**

  ```
  openssl genpkey -outform pem -algorithm rsa -pkeyopt rsa_keygen_bits:2048 -aes-128-cbc -pass
  pass:cyber123 -out CLprivKey.key
  ```

- **Extract the public key**

  ```
  openssl rsa -in CLprivKey.key -pubout -out CLpubKey.key
  ```

- **Create certificate signing request to send to CA**

  ```
  openssl req -new -outform pem -key CLprivKey.key -out CLsignREQ.csr
  ```

  Certificate information:

  ```
  Country Name (2 letter code) [XX]: CA
  State or Province Name (full name) []: Ontario
  Locality Name (eg, city) [Default City]: Ottawa
  Organization Name (eg, company) [Default Company Ltd]:SkyNet Corp
  Organizational Unit Name (eg, section) []:Security
  Common Name (eg, your name or your server's hostname) []:bank.skynet.ca
  Email Address []:.
  ```

A challenge password []:.
An optional company name []:.

- **Send signing request to CA**

```
scp CLsignREQ.csr cyber_admin@10.10.1.12:/home/cyber_admin/Desktop/Certificates/
```

- **Receive signed certificate from the CA and place it in the certificates folder**

```
mv CLcert.cer /etc/pki/tls/certs/
```

4. **<u>Bank requests Certificate from CA for Web Server Authentication</u>**

- **Generate a 2048-bit CA key with AES128CBC**

```
openssl genpkey -outform pem -algorithm rsa -pkeyopt rsa_keygen_bits:2048 -aes-128-cbc -pass
pass:cyber123 -out SRVprivKey.key
```

- **Extract the public key**

```
openssl rsa -in CLprivKey.key -pubout -out CLpubKey.key
```

- **Create certificate signing request to send to CA**

```
openssl req -new -outform pem -key SRVprivKey.key -out SRVsignREQ.csr
```

<u>Certificate information:</u>

```
Country Name (2 letter code) [XX]:CA
State or Province Name (full name) []:Ontario
Locality Name (eg, city) [Default City]:Ottawa
Organization Name (eg, company) [Default Company Ltd]:SkyNet Corp
Organizational Unit Name (eg, section) []:Security
Common Name (eg, your name or your server's hostname) []:bank.skynet.ca
Email Address []:.
A challenge password []:.
An optional company name []:.
```

- **Send signing request to CA**

```
scp SRVsignREQ.csr cyber_admin@10.10.1.12:/home/cyber_admin/Desktop/Certificates/
```

5. **<u>Root CA signs the certificates sent by the client and server, then distributes the signed certificates</u>**

- **Sign client certificate with proper extensions**

<u>Make extension parameters text file with following information:</u>

```
keyUsage=digitalSignature, nonRepudiation
crlDistributionPoints=URI:http://pki.skynet.ca/RevokedCerts.crl
```

Sign request from client

```
openssl x509 -req -in CLsignREQ.csr -CA CArootCert.cer -set_serial 0x300 -sha256 -CAkey CAprivKey.key -days 365 -extfile CLcertExtensions.txt -out CLcert.cer
```

Send signed certificate to client

```
scp CLcert.cer cyber_admin@10.10.1.10:/home/cyber_admin/
```

This Certificate along with the self-signed root CA certificate is sent to the client for signing documents.

- **Sign server certificate with proper extensions**

  Make extension parameters text file with following information:

  ```
  keyUsage=digitalSignature
  extendedKeyUsage=serverAuth
  crlDistributionPoints=URI:http://pki.skynet.ca/RevokedCerts.crl
  subjectAltName=DNS:www.bank.skynet.ca
  ```

  Sign request from server

  ```
  openssl x509 -req -in SRVsignREQ.csr -CA CArootCert.cer -set_serial 0x200 -sha256 -CAkey CAprivKey.key -days 365 -extfile SRVcertExtensions.txt -out SRVcert.cer
  ```

  Send signed certificate to server

  ```
  scp SRVcert.cer cyber_admin@10.10.1.11:/home/cyber_admin/
  ```

  This certificate along with the self-signed root CA certificate is sent to the bank for using with the server for auth.

6. **Server ensures secure configuration for file transmission with perfect forward secrecy**

Security of data in transit has been ensured by encrypting the session of the bank webpage with TLS 1.3 in accordance with NIST.SP.800-52r2. It has also been ensured that the server certificate is trusted by the client. The implementations are done as follows:

- **Modify server configuration**

  Copy updated upload.html and upload.php files to server

  Change file permissions so apache can read them

  ```
  chown root:root upload.*
  chmod 644 upload.*
  ```

  Make server serve the html file by default

  ```
  mv upload.html index.html
  ```

  After receiving signed certificate, put it in the correct location

  ```
  mv SRVcert.cer /etc/pki/tls/certs/
  ```

In apache's ssl.conf file, configure server to always serve over HTTPS for transit security

```
NameVirtualHost *:80
<VirtualHost *:80>
  ServerName bank.skynet.ca
  DocumentRoot "/var/www/html"
  Redirect / https://bank.skynet.ca
</VirtualHost>
```

In apache's ssl.conf file, configure the certificate that the server uses

```
SSLCertificateFile /etc/pki/tls/certs/SRVcert.cer
SSLCertificateKeyFile /etc/pki/tls/certs/SRVprivKey.key
SSLCACertificateFile /etc/pki/tls/certs/CArootCert.cer
```

In apache's ssl.conf file, configure the security protocols and methods the server uses

```
SSLProtocol +TLSv1.3
SSLCipherSuite ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-
ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-
SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256
```

Restart apache server

```
apachectl stop
apachectl start
```

Add this to hosts file

```
10.10.1.12 pki.skynet.ca
```

7. **Client imports root CA certificate into its environment and signs payroll file to ensure server is trusted**

- **Client receives root ca certificate**

- **Import CA into browser's (Firefox ) trusted CA list**

  menu -> preferences -> privacy & security -> view certificates -> authorities -> import -> click root cert -> ok

- **Add hosts file entries to resolve names**

  ```
  10.10.1.11 bank.skynet.ca
  10.10.1.12 pki.skynet.ca
  ```

- **Sign the payroll document**

  ```
  openssl dgst -sha256 -sign CLprivKey.key -out PayrollSig.bin Payroll.xlsx
  ```

Client then sends payroll documents to the bank with digital signature using the CA authorised key to ensure authenticity, integrity, and non-repudiation.

8. <u>**Server verifies the payroll file's digital signature and signing certificate and stores the files**</u>

- **Verify the signature and certificate**

  <u>Alter the upload.php file by inserting the following code</u>

  ```
  $cmd = "openssl verify -crl_check -crl_download -trusted /etc/pki/tls/certs/CArootCert.cer
  $target_fileX509";
  $valRes = shell_exec($cmd);
  echo "<p style='color:blue; font-size:1.5em;'> Certificate Validation Results</p> &nbsp &nbsp &nbsp
  $valRes";

  $cmd = "openssl x509 -noout -ext keyUsage -in $target_fileX509";
  $valRes = shell_exec($cmd);
  echo "<p style='color:blue; font-size:1.5em;'> Certificate Extension Results</p> &nbsp &nbsp &nbsp
  $valRes";
  if(strpos($valRes, "Digital Signature") == false || strpos($valRes, "Non Repudiation") == false){
      echo "<p> &nbsp &nbsp &nbsp Bad extensions.</p>";
  }
  ```

  <u>Configure server to let apache files communicate with the internet</u>

  ```
  setsebool -P httpd_can_network_connect 1
  ```

- **Encrypt and store the file on the server (proposed)**

  Bank preserves the document in the cloud storage in encrypted format to ensure confidentiality. The encryption solution used is FPE as approved by NIST 800-38G and FIPS 140-2. **Format-preserving encryption** (FPE) is designed for data that is not necessarily binary. FPE assures that it would be following almost all regulations and standards which would be enough to satisfy regulatory requirements of **HIPAA, PCI DSS** etc. FPE includes MFA (multi factor authentication) DLP (data loss prevention). It is suitable for data on use.

## <u>Conclusion</u>

This project has been implemented to ensure the data transmission, preservation, storage, etc. are processed with the highest possible cyber security so that data confidentiality, integrity, authenticity, and non-repudiation are maintained in every sphere. All algorithms used are NIST approved. System worked excellently. Due to technical limitations some of the planned features could not be practically implemented, however, those are shown as proposed so that it may be understood that those issues are not overlooked.