Web Based Secure Purchase Order

Computer Engineering Department, College of Engineering
San Jose State University, San Jose, CA 95192

Chaitra Satyanarayana
chaitra.satyanarayana@sjsu.edu

Rikitha Manjunath
rikitha.manjunath@ sjsu.edu

Nischala Raja Shekar
nischala.rajashekar@ sjsu.edu

Annesha Mukherjee
annesha.mukherjee@ sjsu.edu

Abstract:*Web based secure purchased order is implemented using the scenario of Student purchasing centre at SJSU where people can sign in/login and buy Laptop, Book, Pen and Bag.2048 bits of public and private keys are generated for Public Key Cryptography. All the signatures involves MD5 for ensuring data integrity. Private key is used to sign, for the sake of authentication. The contents are then encrypted using public key for data confidentiality.All the communications between parties are through secure email.The public and private key is stored in database.*

1. INTRODUCTION

A web based secure purchase order system allows the user to buy products on the internet and routes the purchase order by sending a secure email to supervisor for signature then to the purchasing department.

The whole project is divided into separate parts:

- Purchase order website
- Security
- Database

We have done the purchase order for a scenario where students can sign in and login using a email ID and purchase items such as Laptop, Bag, Book and Pen.A public and private key is generated and assigned to a user and is used to sign and encrypt the Purchase Order(PO). When the user confirms the order, encrypted-signed PO is sent to supervisor and order department. Supervisor verifies the signature and send the encrypted and signed PO is sent to Order department. Order Department verifies the sign of both user and supervisor and matches the PO sent by both the parties. If the authentication and data integrity is ensured, the order department sends confirmation of the order through email to the user.

2. PROJECT REQUIREMENTS

Implement a secure purchase order system that allows the user to enter a purchase request and routes it (by secure email) to a supervisor for signature and then to the purchasing department

- All user interactions will be Web
- All connections between parties will be preceded by public-key mutual authentication.
- The signatures of both the purchaser and the supervisor will be public key based, and will be performed on a hash of the purchase order. The signature of the purchaser will be sent to both the supervisor and the orders department. If an order is approved by the supervisor, the orders department can across-check the digest signed by the supervisor with the digest signed by the purchaser. The signature is obviously important in preventing repudiation. I am purposely ignoring the possibility that a user will "publish" their key to back up a repudiation. Ideally, the user 's key will not be easily accessible and,since the whole process takes place in one organization, the possible means of revealing a key are very limited. The

biggest threat is a user using another user's

3. CRYPTOGRAPHIC ALGORITHMS

3.1 RSA Algorithm

RSA is the most popular algorithm for public key encryption. RSA scheme is a cipher in which the plain text and the ciphertext are integers between 0 and n-1, for any value of n. Generally, the size of n is 1024 bits or n is less than 2^{1024} .

Description of the algorithm:

RSA uses an expression which has an exponential component. The message is encrypted in blocks, where each block has a binary value less than n, i.e. $n \leq \log_2(n)+1$.

For a message M and cipher text C, the following encryptions and decryptions are performed by RSA:

$$C = M^e \bmod n$$

$$M = C^d \bmod n$$

Where, $n = p*q;\ p \neq q$ (p, q – random prime integers)

$e = \int ;\ 1 < e < \phi(n)$ (public key exponent)

$d \equiv e^{-1}(mod\ \phi(n))$ (private key exponent)

h public key being [e, n] and private key being [d, n].

To obtain the desired security in encryption, certain requirements must be met:

- It is possible to find the values of e, d, n such that $M^{ed} \bmod n = M\{\forall M < n\}$
- It is computationally easy to calculate $M^e \bmod n\{\forall M < n\}$ and $C^d \bmod n\{\forall M < n\}$
- It is mathematically infeasible to calculate d, given e and n.

3.2 Digital Signatures

Digital Signature is a process that guarantees that the contents of a message have not been altered in transit. When you, the server, digitally sign a document, you add a one-way hash of the message content using your public and private key pair.Your client can still read it, but the process creates a "signature" that only the server's public key can decrypt. The client, using the server's public key, can then validate the sender as well as the integrity of message contents.

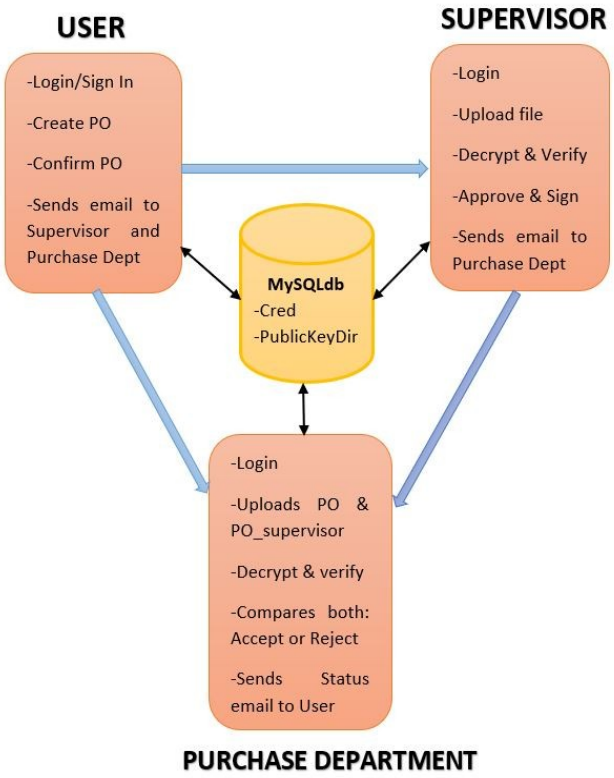
The digital signature can be considered as a numerical value that is represented as a sequence of characters. The creation of a digital signature is a complex mathematical process that can only be created by a computer.

Consider a scenario where Alice has to digitally sign a file or an email and send it to Bob.

- Alice selects the file to be digitally signed or clicks on 'sign' in her email application
- The hash value of the file content or the message is calculated by Alice's computer
- This hash value is encrypted with Alice's Signing Key (which is a Private Key) to create the Digital Signature.
- Now, the original file or email message along with its Digital Signature are sent to Bob.
- After Bob receives the signed message, the associated application (such as email application) identifies that the message has been signed. Bob's computer then proceeds to:
    - Decrypt the Digital Signature using Alice's Public Key
    - Calculate the hash of the original message
    - Compare the (a) hash it has computed from the received message with the (b) decrypted hash received with Alice's message.
- Any difference in the hash values would reveal tampering of the message

4. TECHNICAL DETAILS

4.1 Project Architecture



SJSU Student Purchase Centre has three main components: User, Supervisor and Order Department.When a user signins, a public and a private key is generated and stored in the database. Public key is stored in public key directory which is accessible to all the components. User authentication is checked during the login using the email ID and password of the user. When the user selects and confirms the purchase, purchase order is signed by the user . PO,along with signature is encrypted and sent to both Supervisor and Order Department. the supervisor downloads the attachment, decrypts, verifies the signature and data with hash. The PO is then signed ,encrypted and sends and email to Order Department.The Order Department now downloads both the file, verifies both the signatures and compares the hashes of both the received PO.Once verified, it will then send a confirmation email to the user.

Supervisor receives the encrypted PO via email and downloads the file. Later the same file is uploaded to the website, decrypted and checked for the contents of PO. Once the supervisor has verified the signature of user, he/she later approves the PO by encrypting and signing the PO with purchase department public key and his private key. Supervisor signed PO sent to order department via email for further approval.

Once the user has purchased the items and the Supervisor has verified and approved the Purchase Order, the purchase Order department can cross check the digest signed by the supervisor with the digest signed by the purchaser and if it matches, the order department can approve the Purchase Order, a confirmation mail will be sent to the user with the approved status, following the processing and delivery of purchased items. If the authentication fails, then, the Purchase Order will be rejected, and approval status of authentication failure is sent to the user.

4.2 Technologies used

- RSA-2048: Encryption Algorithm
- MD5 : Hashing Algorithm
- Python : Programming language
- Flask : Flask is the web framework used to implement REST APIs of the web server of the purchase centre
- MySQLdb : The database used for Public key directory and private key/credentials .
- HTML/CSS: Front-end technology

4.3 Modules used

- rsa : For encryption algorithms. Includes functions such as encrypt, decrypt, sign, verify and nexwkeys.
- MySQLdb : MySQLdb connector module. To connect to public key directory and credentials.
- Pickle : Serializing and de-serializing public and private key objects
- werkzeug : It is a utility library for WSGI.
- os : For forking processes as the server is multi process server.
- smtplib : to send Emails

5. IMPLEMENTATION DETAILS

Representational State Transfer(REST) is the architecture style used to implement the SJSU student purchase center website.This is accomplished by using FLASK web framework of Python. The front-end is implemented using HTML/CSS.

MySQL database is used to store public and private keys , email ID and password.

The database ‘spo’ is created with two tables

- PublicKeyDir - Table acts has public key directory where public key of a user can be accessed with the EmailID.

EmailID (varchar 255)	PublicKey (varchar 5000)

- Cred - This is a table where private key is stored safely.Private key can be accessed only when the user is authenticated himself with the right combination of email and password.

EmailID (varchar255)	Password (varchar255)	PrivateKey (varchar 5000)

Each module implementation details are as given below

5.1 User

The user application is implemented for user login and purchase activities.The REST API “/” is used to access the application.The application does the below activities:

When the user inputs the email ID and password, the application will search for the email Id , if it does not match then it will create a profile for the email Id. It will create a public and private key and stores it in the database.If the email ID is existing, it will verify if the password matches.If the password is verified, the conformation for the purchase items os obtained from the user. the purchase order is hashed with MD5 and signed with the private key of the user.

The signature is concatenated with the PO, broken into chunks of 245 bytes. Encrypted with the public key of supervisor and order department one by one. The contents are written to a file PO. txt.

The file is attached to the email and send to order department and supervisor according to the public key used to encrypt.

5.2 Supervisor

Similar to user module, Supervisor module is also built using python flask as backend. All transactions performed are web based. An encrypted file of the PO is received at the supervisor’s email. Supervisor logs into his email and downloads the encrypted text file and then performs the following steps:

1. Run the Supervisor python script which links to the localhost and gives the address of localhost as http://127.0.0.1:5000/
2. Open the browser and run the address of localhost, it displays login page of the supervisor.
3. Enter the login credentials, if the email id and password are enter correctly, it logins into the webpage else displays error message for wrong credentials and login again.
4. The next page is to upload the encrypted file. once uploaded, it save in the local directory of the script.
5. Next, the supervisor can choose to display the contents of PO. At the backend, rsa.decrypt function is applied to the encrypted PO. Private key of the supervisor is retrieved from the database created using init_db. The encrypted format for PO from the user is,

Encrypt((sign(hash(PO))+PO), recepient_public).

For the purpose of our project and to decrypt with less problems the final PO file is concatenated

with “SSSS” as explained in user module.
6. For decryption, the PO file is split based on the delimiter “SSSS”. We get encrypt(PO) and encrypt(sign(hash(PO))). Apply rsa.decrypt (encrypt(PO),private key of supervisor)), we get the contents of PO. Apply rsa.verify(sign(hash(PO)), public key of user) to check for authentication. This message and the user email is displayed, then the supervisor should ‘Verify & sign’ the PO to approve.
7. To approve PO of the user, supervisor needs to verify and sign the PO with his private key. The encrypt is done using rsa.encrypt with purchase department public key extracted from the database. Authentication is provided by rsa.sign which signs the PO with supervisor’s private key and MD5 hash. This encrypted text is saved in ‘PO_supervisor.txt’ file and emailed to order department for further approval.

5.3. Order Department

Once the Order Department receives the Supervisor and the User Purchase Order, he/she logs into his gmail account and download the Purchase Order from both user and supervisor.
1.Order Department can login to his system and these encrypted files are uploaded. All submitted form data can be forged, and filenames can be dangerous. For this purpose,we have used a function secure_filename in flask to secure a filename before storing it directly on the filesystem. If the one or both files are not uploaded then it it prompts to go back and upload again.
2.Upon uploading the user encrypted file decrypt function will decrypt the encrypted text using the built in function rsa.decrypt, the plaintext and the signature of the user is obtained.
3.In a similar fashion, the plaintext and the signature of the Supervisor is obtained
4.The PO of the User and the Supervisor are hashed and checked if both match. If both of the digest matches then the Verified status or if doesn’t match then Authentication failure status is displayed to the Order Department..
Depending on the status, the Order Department can approve or reject the PO.
5. Once approved or rejected, the approval status mail is sent to the user.

6 PROJECT SETUP

6.1 Initial Setup

- Download ‘rsa’ module for various cryptographic functions
  `sudo pip install rsa`
- Install ‘flask’ module
  `sudo pip install Flask`
- Install ‘pickle’ module
  `sudo pip install flask`
- Install ‘mysql’
  `sudo apt-get install mysql-server`
- Run the init_db.py to create the necessary ,database, tables and insert email, password and public-private keys of supervisor,purchase department and order department.
  `python init_db.py`

6.2 Steps to run SJSU purchase center application

- export flask user application and run the purchase centre script
  `export FLASK_APP = purchase_center.py`
  `flask run`
- The application runs with port 5000.Open the browser, Browse http://localhost:5000 Login and make the required purchases.Close the application

- export flask supervisor application and run the supervisor application
  export FLASK_APP =supsv.py
  flask run
- Open the supervisor email, download the PO text file sent by the purchase center, login and upload the file ,verify signature and view the contents. sign and encrypt the PO, forward it to Order department
- export flask order department application and run the supervisor application
  export FLASK_APP =orderdpt.py
  flask run.

- login , upload both the files PO obtained from supervisor and purchase department.Verify, both the signatures , if verified confirm the purchase order.

5. RESULTS AND CONCLUSION

We successfully implemented Web based Secure Purchase Order System. All transactions are sent using a secure medium based on RSA algorithm. Security application of RSA algorithm is learnt and understood.

Link to github:https://github.com/ChaitraSatyanarayana/Web-based-secure-purchase-order/blob/master/README.md

References

- [Cryptography and Network Security: 1Principles and practice (6th Edition)
- http://naelshiab.com/tutorial-send-email-python/
- https://www.pythonsheets.com/notes/python-crypto.html
- http://flask.pocoo.org/docs/1.0/tutorial/templates/
- https://pypi.org/project/rsa/
- http://flask.pocoo.org/docs/0.12/tutorial/
- https://www.w3schools.com/html/
- https://www.instantssl.com/https-tutorials/digital-signature.html