Lab 2: Cryptosystem Implementation

- Due Oct 16 by 8:59pm
- Points 100
- Available Sep 30 at 1pm - Dec 28 at 8:59pm

Lab 2: Central Ledger to preserve chain of custody inalterability

Objective

In this lab assignment, you will implement a cryptographic scheme intended to provide the most important protections for preserving chain of custody of a digital evidence. Your scheme must provide integrity and non-repudiation among the custodians involved, and also the communication with a central ledger, which implements a hash chain of the custodian signatures preventing the modification of the records.

The ledger is maintained in a remote machine in the internet and custodians interact with it through an API. Every signature recorded will be hash-chained with the previous hash generated and that will result in last hash of the chain.

```
LastHash="5a2fc4c4e22d0d2e88cfeb179d46e5fa119a8a87784e54849e37c7fe3c43fd51"

-- New signature record arrives to the ledger --

{"user": "Client43", "comment": "Custodian2 Signature over evidence", "message": "NjdNRjk2WVcvZ29waFJwUm5lUHRPVX
VOV1dwUytBbjNzdlhmRmR4NkY5dmE0Yk51S1ZRdDdRVEpCQnYyMjJvcEFLNXJWS0Y0ekd2Q1VvWlRWR3g4SXBJS2dWSnZMQ0tCcFZqcEJxdjJvdl
VpN3hMVnN4N2RjNHkydE5mcWJKamdxRXVkRnN3PT0k"}

-- New last hash calculation --

Signature="NjdNRjk2WVcvZ29waFJwUm5lUHRPVXV0V1dwUytBbjNzdlhmRmR4NkY5dmE0YkJ1S1ZRdDdRVEpCQnYyMjJvcEFLNXJWS0Y0ekd2Q
1VvWlRWR3g4SXBJS2dWSnZMQ0tCcFZqcEJxdjJvdlVpN3hMVnN4N2RjNHkydE5mcWJKamdxRXVkRnN3PT0k"

LastHash=$(echo "$LastHash$Signature"|sha256sum)
    5a6f1397dfdfab6c74e95bb01f3979eca5518c16808d8f03203c525160d0bc29

-- Record at the Ledger website --

Date                Name        Hash
Comment
2024-09-24 09:11:45    Client43    5a6f1397dfdfab6c74e95bb01f3979eca5518c16808d8f03203c525160d0bc29
  Custodian2 Signature over evidence
```


Setup

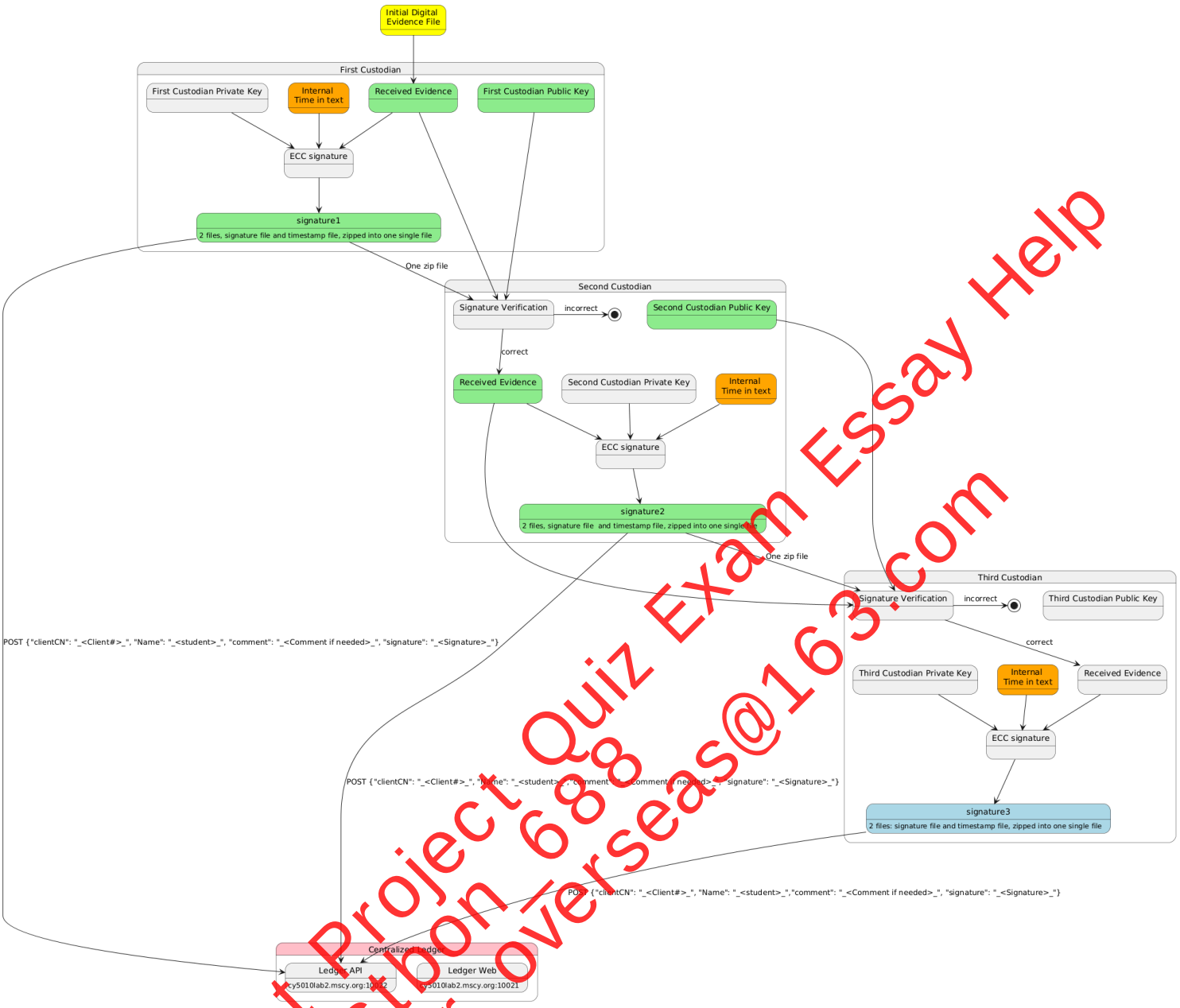
You will be using the **OpenSSL** toolkit for this lab. OpenSSL is a general purpose cryptographic toolkit that is widely used in both non-commercial and commercial applications. Read more about OpenSSL here Links to an external site. Also read the man page of OpenSSL by typing man openssl in a terminal. For those new to **bash scripting**, reading this tutorial **Bash Scripting (https://www.codecademy.com/learn/bash-scripting)** may help you.

Instructions

Implement a **BASH script** cryptographic scheme that provides integrity and non repudiation for digital evidence chain of custody involving N custodians (2<N<9) and the corresponding protected hands-off. An implementation of this scheme must receive a file representing the digital evidence (do not hardcode the filename, it needs to be provided as an *argument to the script*), and the N key pairs (private and public) for the N custodians involved. The output of the script should provide the requested intermediate hands-off files created by an specific custodian.

Write a **bash script** that implements the cryptographic scheme using openssl. Assume that all public-private keys use **ECDSA** to perform digital signatures and hash functions used are SHA3 and belong to the `keccak` family.

There are many ways to perform a chain of custody, but in this case, we want you to follow the design described in the following diagram, which includes some additional elements that need to be included. Make sure that your implementation follows exactly what is in the diagram, any miss-alignment will penalized in grading. Further explanations will be provided during weekly sessions.



Your script must output just one file for both encryption (Hint: use the zip/tar command to zip multiple files together) and decryption, and after every time a signature is performed, it needs to be recorded into a centralized Leger using an API (POST) according to the following example:

```
curl -X POST https://cy5010lab2.mscy.org:10022/api/v1/hash -H "Content-Type: application/json" -d
'{"client": "client40", "user": "jose", "comment": "custodian 1 signature", "signature": "5c864e7e8f
844131a8444c4832e2e9f1a0a26ba36865df29596aa068510d3b16"}'
```

Also, the API is protected with public key authentication, so you need to adapt the previous command because only with a valid certificate you would be able to record a signature into the ledger. You need to adapt the API call to successfully provide client authentication and record each custodian signature. Your public key certificate and keys can be found **here (https://northeastern.instructure.com/courses/192842/pages/certificates-and-keys-for-lab-2)** .

- Once your cryptosystem is working, you can verify that your are recoding into the ledger connecting to https://cy5010.ccs.neu.edu:10021/hash_chain

The bash script MUST run as follows, make sure that arguments files are provided in this exact order:

The script will receive a argument indicating which custodian is acting

First Custodian (lab2.sh -1)

```
./lab2.sh -1 <evidence_file> <custo1_priv_key> <*custo1_output.zip*>
```

Custodian >1 (lab2.sh -n)

```
./lab2.sh -n <evidence_file> <custo_n-1_output.zip> <custo_n-1_pub_key> <custo_n_priv_key>
<*custo_n_output.zip*>
```

Guidelines for the script:

- The code should be readable with appropriate comments where necessary.
- Use algorithms and key sizes appropriate for security today.
- Do not hardcode the name of key-pair files, input files, or output files. These should be accepted as arguments when executing the script and may change.
- Do not hardcode input or output files, all these files names will be provided as arguments to the script
- Do not use subfolders for any input and output files. All should be stored into the same directory where the script is executed.
- You will surely use temporary and intermediate files in your process, make sure that all of them are deleted after any execution of your script.
- The script should contain basic level of error handling and provide appropriate messages to the user. Error messages should be printed to stderr:
  - eg. Incorrect arguments. Expected input in the format ./lab2.sh ...
  - eg. if signature verification fails, the message should indicate which one was and in which part of the process"
- Error messages must always contain the string ERROR username where username is your labtest username (the first part of your email address last.f@northeastern.edu).

Deliverables

Once you have completed your script you need to upload it to the CY5010 Master Server

```
cy5010.mscy.org at port 17001
```

This is not your individual Virtual Machine, but the master server. So, you cannot work on it to complete the assignment, just to submit your final script. Once, you upload it there, you can test it to verify that is working well.

For this lab, each student must submit only your script (which must include comments explaining your implementation details) to submit it you need to:

- Create a folder into your home directory named lab2 `~/lab2/`. Store there your lab2.sh script with proper comments and error handling. Include there also the key pair(s) and evidence file that you used for testing your own script. TAs will test intensively your script with different key pair(s) and evidence files.
- We will also track your records at the central ledger to verify that your cryptosystem in working at https://cy5010lab2.mscy.org:10021/hash_chain

Grading Rubric

Task	Weight
Script completes all functions with the test files that you provided, according to the assignment functionality and implementation details	30%
Script completes all functions working with other test files according to the assignment functionality and implementation details.	40%
Script successfully record signature at the centralized ledger	20%
Script manages error scenarios according to what is required in the assignment	10%