PA-02: Message Digests & RSA Digital Signatures using OpenSSL

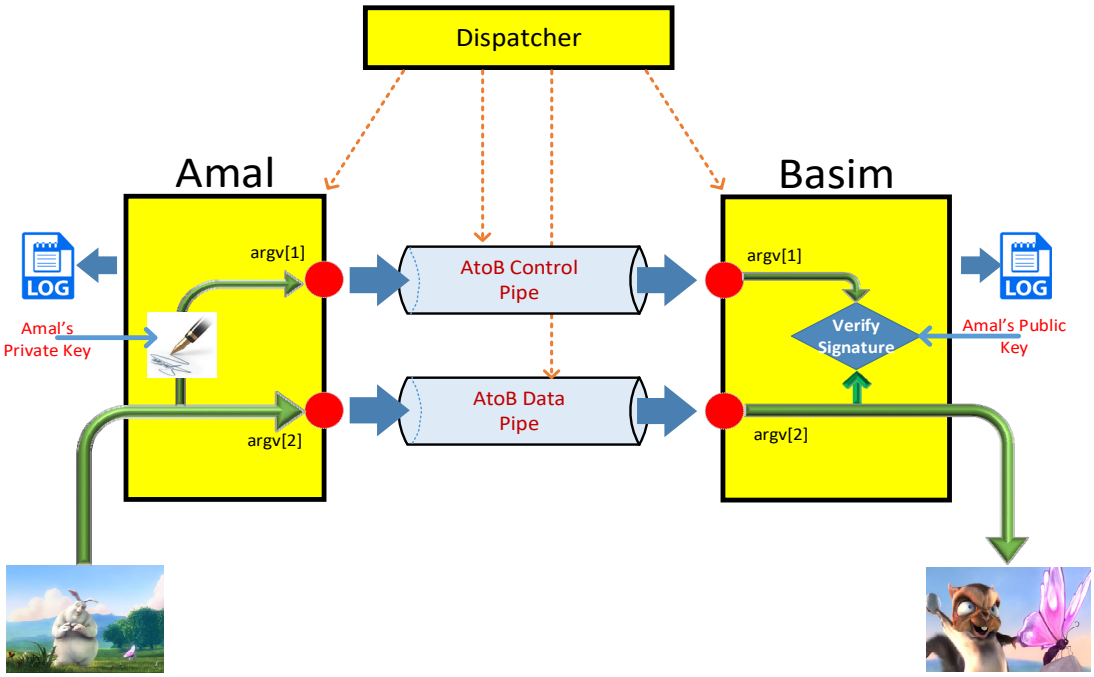
Learning Objectives:

By the end of this programming assignment, you will learn:

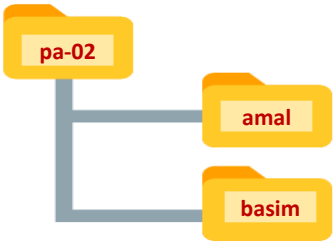
- ✓ calculate the digest of a large file using a secure hashing function,
- ✓ Digitally sign the file digest using the sender's private RSA key, and
- ✓ verify the integrity of the transferred file and the identity of its sender by validating the signature.

Project Requirements:

Following the example of the code in Programming Lab-02 on RSA encryption, your mission is to digitally sign a large file, too big to fit in memory, and then send the **unencrypted** file and the signature to a second party who will validate the signature.



Using the directory structure shown here:



1. Use the code in `dispatcher.c` that I provided to you in order to create the two pipes and launch the Amal (in `amal.c`) and Basim (in `basim.c`) child processes passing them (as command-line arguments) the end points of the two pipes representing the file descriptors they need to send / receive messages to each other.
   - The **AtoB Data** pipe is used to transfer the **unencrypted** video file `bunny.mp4` from Amal to Basim

- The **AtoB Control** pipe will carry Amal's RSA digital signature on the video file to Basim

2. Update the `myCrypto.c` source file so that you implement the following function:

```
size_t fileDigest( int fd_in , uint8_t *digest , int fd_out) ;
// Read all the incoming data stream from 'fd_in' file descriptor
// Compute the SHA256 hash value of this incoming data into the array 'digest'
// If the file descriptor 'fd_out is > 0, write a copy of the incoming data stream
// to 'fd_out'.  Returns actual size in bytes of the computed digest.
```

This function is similar to the `encryptFile()` function, except that it uses these openSSL library functions in this order:

- `EVP_MD_CTX_new()`
- `EVP_DigestInit()`
- While() { … `EVP_DigestUpdate()` … }
- `EVP_DigestFinal()`
- `EVP_MD_CTX_destroy()`

Amal will use this function as follows: (Basim has a similar, but not identical, behavior)

```
fd_ctrl = atoi( argv[1] ) ;
fd_data = atoi( argv[2] ) ;
fd_in   = open("bunny.mp4" , O_RDONLY , S_IRUSR | S_IWUSR ) ;
mdLen   = fileDigest( fd_in , fd_data , digest ) ;  // also dump file to Basim
```

3. Create the `genPkey.sh` shell script file (from the provided skeleton) to generate a pair of 2048-bit RSA private / public keys for Amal into the files: `amal_priv_key.pem` and `amal_pub_key.pem`, respectively. For debugging purpose, the script should also display the information of the just-created Amal's RSA key. This script needs to run ONLY once during the development of your project.
4. Create the `amal/amal.c` source file whose `main()` function:
   i. Gets the write-end file descriptors of both pipes from the command-line arguments.
   ii. Opens `bunny.mp4` and calls `fileDigest()` to compute the SHA256 hash value of the file while transmitting a copy of the file over the **AtoB Data** pipe.
   iii. Uses Amal's RSA private key to encrypt the hash value computed in the previous step. This is Amal's digital signature on this video file
   iv. Transmits Amal's digital signature to Basim over the **AtoB Control** pipe
5. Create the `basim/basim.c` source file whose `main()` function:
   i. Gets the read-end file descriptors of both pipes from the command-line arguments.
   ii. Calls `fileDigest()` to receive the incoming data file over **AtoB Data** pipe, and compute its SHA256 hash value, while saving a local copy of that file as `bunny.cpy` file in Basim's folder.
   iii. Receives Amal's digital signature on the video over the **AtoB Control** pipe.
   iv. Verify Amal's signature aided with her RSA public key.
6. All dynamically-allocated objects must be deallocated by calling the appropriate API / system calls.

7. Have both `amal.c` and `basim.c` log the main steps they are taking to text files named `logAmal.txt` and `logBasim.txt`, respectively.
8. Use the provided shell script file `pa-02.sh` that compiles and builds all programs, launches the dispatcher, then uses the "`diff -s`" command to compare the original `bunny.mp4` vs `bunny.cpy` files.

Before submission, clean the folders of all executables and move the `bunny.*` files away (so that it is NOT included in the submitted files). Submit the entire `tar`'ed `pa-02` folder side by side with any other items requested from you.

Inclusion of the video files in your submission results in a large download delay when grading, and may cost your grade to receive a significant penalty.