List of Software and Packages with versions:

1) Python:
    a) Device 1: Python 3.11.9

4) Libraries:
    - socket
        - Allows for communication between devices via endpoints known as sockets
        - Used to create network sockets for digital and physical twins
    - random
        - Allows for random number generation
        - Used to generate random computational values
    - time
        - Allows for work with dates, times, and timestamps
        - Used to take timestamps
    - json
        - Allows for functionality with JavaScript Object Notation data (JSON)
        - Used to convert dictionaries into strings and send data in a lightweight format
    - ecdsa
        - Allows us to bring in the PointJacobi class for help in converting points on curve into dictionaries and back for sake of sending and receiving messages in the protocol
        - i Used in importing the curve used in the protocol
    - sympy
        - Allows us to import the mod_inverse function from the sympy library
    - hashlib
        - Allows for use of hashing algorithms
        - Used to implement the SHA-256 and SHA-512 algorithms
    - os
        - Allows for interaction with operating systems
        - Used to automate the authentication process

Installation of Software and Packages:

a) Windows
    i) Step 1: Install IDE (IDE used: Visual Studio Code)
        1) Download Visual Studio Code link
    ii) Step 2: Install Python
        1) Download Python link
        2) pip install <insert library> (socket, threading, random, time, json, tinyec, secrets, hashlib, Crypto, os)

GitHub link:
https://github.com/jwwyeth/Secure-Lightweight-Proxy-Re-encrypted-Data-Relay-in-IoT-enabled-Smart-Interconnected-Factories.git

Description of files in GitHub:

TA.py:

TA.py serves as the trusted authority in our protocol that hands out all values to be used for the protocol's computations. It defines the curve and give out values such as the generator point P, prime order of the curve q, and the key pairs for all entities.

ECC.py:

ECC.py serves as a class of the elliptic curve used in the algorithm and serves as a modularized means by which TA.py can create the curve

POCDT1.py:

POCDT1.py emulates the entity, PoC-DT1 in the protocol. It is given values from TA and generates the original encrypted message to then send to the edge server.

ES.py:

ES.py emulates the edge server in the protocol. It is given the re-encryption key from the TA and re-encrypts the encrypted message given to it by PoC-DT1 and sends it to PoC-D2.

POCDT2.py:

POCDT2.py emulates the entity, PoC-DT2 in the protocol. It is given values from TA and generates the decrypts the re-encrypted message given to it by the edge server.

Automation.py:

Automation.py serves as a means to easily test the protocol by running all previously mentioned files in successive order.

Execution Steps:

Preamble:

The github possesses all files needed to run the protocol in a windows environment. The only alteration that needs to be changed is within Automation.py where the file path for each file in the protocol must be added after the protocol is downloaded into whatever space on one's computer and prepared for testing. These spaces that require the addition of the file path are denoted as "YOUR FILE PATH HERE" in the file. Another specification of note, is that should this protocol be tested in a multi device environment then the server functionality implemented across each file will need to be changed. This protocol was tested on one device and as such ip address was not considered in the code. If this protocol were to be tested in a multi-device setting then this matter should be addressed.

Manual Testing execution:

- Download the repository and store the protocol in a preferred space
- Go into automation.py and where dictated as "YOUR FILE PATH HERE" input the location of each file's location on one's device relative to the variable name given ie. POCDT1_path = "file_path to POCDT1.py"
- Should there be a need or interest in a multi-device testing of the algorithm then alterations to POCDT1.py, POCDT2.py, and ES.py are needed in terms of their server connections.
- The socket programming connections are as follows:
  - All entities temporarily bind to port 5050 with TA
  - PoC-DT1 binds to ES on port 8080
  - ES binds to PoC-DT2 on port 7070
- Run the automation file in either visual studio or on your computer's terminal
- Various terminal windows will appear showcasing each entity in the protocol and its execution