List of Software and Packages with versions:

1) Python:
    a) Device 1: Python 3.11.9
    b) Device 2: Python 3.12.3
2) Wireshark:
    Device 1: 4.0.8
    Device 2: 4.2.5
3) AVISPA:
    a) Device 2: 1.1
4) Libraries:
    a) socket
        i) Allows for communication between devices via endpoints known as sockets
        ii) Used to create network sockets for digital and physical twins
    b) threading
        i) Allows for thread management
        ii) Used to handle connections between devices
    c) random
        i) Allows for random number generation
        ii) Used to generate random computational values
    d) time
        i) Allows for work with dates, times, and timestamps
        ii) Used to take timestamps
    e) json
        i) Allows for functionality with JavaScript Object Notation data
        ii) Used to convert dictionaries into strings and send data in a lightweight format
    f) tinyec
        i) Allows for elliptic curve cryptography
        ii) Used to generate keys
    g) Secrets
        i) Allows for cryptographically strong random numbers
        ii) Used to select a cryptographically strong number on the elliptic curve
    h) hashlib
        i) Allows for use of hashing algorithms
        ii) Used to implement the SHA-256 and SHA-512 algorithms
    i) Crypto
        i) Allows for cryptographic primitives use
        ii) Used to implement AES encryption/decryption and random prime number generation
    j) os
        i) Allows for interaction with operating systems
        ii) Used to automate the authentication process

Installation of Software and Packages:

a) Windows
   Step 1:
   Step 2:
b) MAC
   Step 1:
   Step 2:
c) Linux
   Step 1:
   Step 2:

GitHub link:
Execution video (screencast) link:

Description of files in GitHub:

RE.py:

RE.py serves as the registry enroller for the protocol and allocates to AMDT1, AMDT2, AM1, and AM2 their starting set of values to then be used in DTSMAKA-1 and DTSMAKA-2. These values given are the pseudo-identity (RID), temporal identity (TID), g, p, q, private key, and public key. The g, p, and q represent random computational values. After the value set is given the socket between RE and any of the other files is broken and the file is then ready to begin authentication.

AMDT1.py:

AMDT1.py emulates AMDT1, a digital twin in the protocol. It is given its starting values from the registry enroller and initiates DSTMAKA-1 by authentication with AMDT2.py to produce a shared session key. Afterwards, AMDT1.py starts DSTMAKA-2 by authenticating AM1.py to produce their separately distinct session key. The .gcode and .ngc files are then read by AMDT1.py and sent to AMDT2.py and AM1.py

AMDT2.py:

AMDT2.py emulates AMDT2, a digital twin in the protocol. It is given its starting values from the registry enroller and initiates DSTMAKA-1 by authentication with AMDT1.py to produce a shared session key. Afterwards, AMDT2.py starts its own DSTMAKA-2 by authenticating AM2.py to produce a separately distinct session key. AMDT2.py then receives the .gcode and .ngc files from AMDT1.py and sends these files to AM2.py.

AM1.py:

AM1.py emulates AM1, a physical machine in the protocol. It is given its starting values from the registry enroller and initiates DSTMAKA-2 by authentication with AMDT1.py to produce a shared session key. After it has been authenticated it then receives the .gcode and .ngc files from AMDT1.py

AM2.py:

AM2.py emulates AM2, a physical machine in the protocol. It is given its starting values from the registry enroller and initiates DSTMAKA-2 by authentication with AMDT2.py to produce a shared session key. After it has been authenticated it then receives the .gcode and .ngc files from AMDT2.py

Automation.py:

Automation.py serves as a file which when run automates the process of testing the protocol via running RE.py, AMDT1.py, AMDT2.py, AM1.py and AM2.py in that order. This negates human delay in individually running each file separately. In order to properly run you must insert the proper file path of each file in RE_path, amdt1_path, amdt2_path, am1_path, and am2_path respectively. When started Automation.py will initiate each file in a new terminal window and carry out the protocol.

Explanation of the Execution cycle of each phase:

DTSMAKA-1 Phase:

- AMDT1.py binds to AMDT2 on socket port 8080
- AMDT1 and AMDT2 execute steps ADT1-ADT4 to authenticate AMDT2
- AMDT1 and AMDT2 execute steps ADT5-ADT7 to authenticate AMDT1

DTSMAKA-2 Phase:

-

Design Decisions:

1) Why green flag? How was the coordination issue resolved?
2)

Execution Steps:

Step 1:

Step 2:

Step 3: