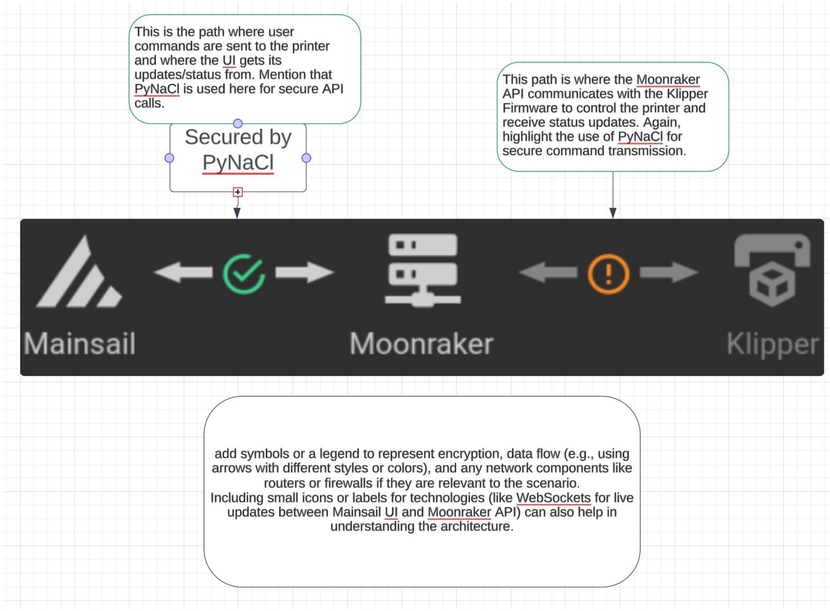
PyNaCl “Salt” Encryption and Authentication for Networking to 3D Printer

Justin Hartberg

Mentors: Clemente Izurieta, Yvette Hastings

Introduction:

In remote networking with 3D printers, ensuring secure and encrypted connections is crucial for maintaining confidentiality and integrity. To solve this issue, I am testing encryption and decryption capabilities using the PyNaCl library to look at the information passed between a computer system and a 3D printer. The use of the PyNaCl library to encrypt the information being sent between a computer and a 3D printer has several implications for cybersecurity. Implementing security measures ensures the confidentiality, integrity, authenticity, and non-repudiation of data exchanged between the computer and the printer. Moreover, these measures enhance the security, privacy, trust, reliability, and availability of the 3D printing system. The inclusion of a robust security library restricts unauthorized access, use, modification, and disclosure of data transmitted between the computer and the printer. This proactive approach mitigates the risk of malicious commands being sent to the printer, safeguarding it and its environment from potential harm in both digital and physical domains. Encrypting the data and authenticating the user credentials improves the safety of the 3D printing setup. Additionally, it ensures that the information originates from the correct source and hasn't been modified by unauthorized parties.

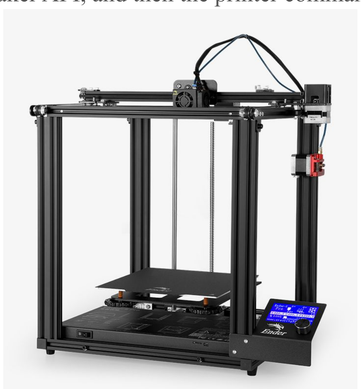


Methods: (Steps for networking to printer from anywhere)

- Initialize printer and software (explain?)
- Modify Mainsail User Interface to send requests to my PyNaCl Flask server
  - This is necessary to send the requests to my python script instead of sending the request directly from the User Interface to the Moonraker API (which provides a bridge between the user interface and the Klipper 3D printer firmware).
  - This is done by modifying the javascript code that sends requests to the Moonraker and change it to send the information to the python script server.

- Then when the user sends a command to the printer, the data of the command is sent to the python script.
- The data is encrypted in the python script, and then the encrypted data is sent to the Moonraker API
- The data is then decrypted on the printer in the Moonraker API, and then the printer command is given to the Klipper firmware

The printer reads the command and runs.



Future direction/ Future works

Pseudo Code:

Import all necessary libraries:

- Flask, threading, socket, nacl* requests

Define ‘generate_key_pair’ function to generate a public/private key pair for signing and encryption for the Mainsail UI

- A new ‘SigningKey’ object is created by calling the ‘generate’ method.
- The ‘generate’ method generates a new private key and returns a ‘SigningKey’ object that contains the private key
- The public key is then made by accessing the ‘verify_key’ attribute of the ‘SigningKey’ object. This returns a ‘VerifyKey’ object that contains a public key
- Then the function encodes both keys as Base64-encoded strings by calling the ‘encode’ method
- Then the public and privates keys are returned as separate values

Define ‘generate_dh_key_pair’ function to generate public/private key pair for Diffie-Hellman key exchange

- Creates new DiffieHellman object and calls ‘generate’ method on it
- The ‘generate’ method returns a tuple containing a private and public key as separate values
- The private key is used to compute a shared secret with a public key from the other party.
- The public key can be shared with other parties to allow them to compute a shared secret with the private key

Define ‘encrypt_message’ function to encrypt a data using the public key

- Create ‘Box’ object for asymmetric encryption
- The private key is kept secret and used for the decryption while the public key can be shared openly and used for encryption.
- ‘Encrypt’ method is used to encrypt message and is passed as byte string and this is returned.

Define ‘decrypt_message’ function to decrypt data using private key

- Create ‘Box’ object
- Both the private key and the public key are passed as Base64-encoded strings that are decoded
- The message is then decoded using the ‘decrypt’ method, converting it from a byte string to a Unicode string

Define ‘sign_message’ function to authenticate the data that will be sent out

- ‘SigningKey’ object is created which is used for digital signatures. This involves using private key to sign message and public key to verify the signature
- The ‘sign’ method is called to sign the data, which returns the signature as a byte string

Define ‘verify_signature function to verify the signature of the data

- ‘VerifyKey’ object is created, which involves using a private key to sign the data and a public key to verify the signature
- The ‘verify’ method is called to verify the signature, which takes two arguments: the signature and the data which are both passed as byte strings
- The ‘verify’ method returns a boolean value indicating whether the signature is valid or not. If the signature is valid, the method returns ‘True’, otherwise it returns ‘False’

verify the signature of a message using the public key

Generate all keys:

- generate a public/private key pair for the Mainsail UI
- generate a the diffie-hellman key pair for the Mainsail UI
- Load the printer's public key

Set up the Flask app

- Define a route to handle incoming requests from the Mainsail UI

Define ‘handle_mainsail_request’ function that handles incoming requests from the Mainsail UI and then sends the data to the Moonraker API

- Function gets data from Mainsail UI using JSON request that returns the data as a python dictionary
- Then data is encrypted using the printer’s public key by creating a secret box and passing in the data, the printer’s public key, and a random nonce (a unique value used to ensure the security of the encryption).
- The secret box object returns the encrypted data as a byte string
- Then a shared secret using the Diffie-Hellman key exchange which returns the shared secret as a byte string
- Then the encrypted data is signed using the Mainsail private key
- Then the encrypted data and signature are sent to the Moonraker API as a string.

PyNaCl “Salt” Encryption and Authentication for Networking to 3D Printer

Justin Hartberg

Mentors: Dr. Clemente Izurieta, Yvette Hastings

Introduction

In remote networking with 3D printers, ensuring secure and encrypted connections is crucial for maintaining confidentiality and integrity. To solve this issue, I am testing encryption and decryption capabilities using the PyNaCl library to look at the information passed between a computer system and a 3D printer (Figure 1). This proactive approach mitigates the risk of malicious commands being sent to the printer, safeguarding it and its environment from potential harm in both digital and physical domains. Encrypting the data and authenticating the user credentials improves the safety of the 3D printing setup and ensures that the information originates from the correct source and hasn't been modified by unauthorized parties.

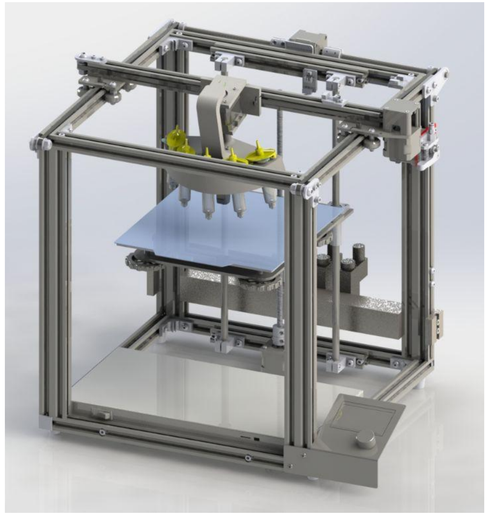


Figure 1: CAD Render of our 3D printer system

Methods

Steps (Figure 2):

- Initialize printer and software
- Modify Mainsail User Interface to send requests to my PyNaCl Flask server
  - This is done by modifying the javascript code that sends requests to the Moonraker and change it to send the information to the python script server.
- The data is encrypted in the python script, and then the encrypted data is sent to the Moonraker API
- The data is then decrypted on the printer in the Moonraker API, and then the printer command is given to the Klipper firmware.
- The printer reads the command and runs.

Pseudo Code:

- Import all necessary libraries (Flask, threading, socket, nacl*, requests)
- Generate public/private key pair for encryption for data from Mainsail UI
- Generate public/private key pair for Diffie-Hellman key exchange (prevent middle-man attacks)
- Create function to encrypt data using public key
- Create function to decrypt data using private key
- Create function to authenticate encrypted data by signature
- Load printer’s public key
- Set up Flask Server
- Handle incoming requests from Mainsail User Interface
- Encrypt data and sign secret box using private key
- Send encrypted data and signature to Moonraker API

Acknowledgments

This work is supported by funding for the VICEROY Northwest Institute for Cybersecurity Education and Research (CySER) provided by The Office of the Undersecretary of Defense for Research and Engineering, in collaboration with the Air Force Research Laboratory and Griffiss Institute.

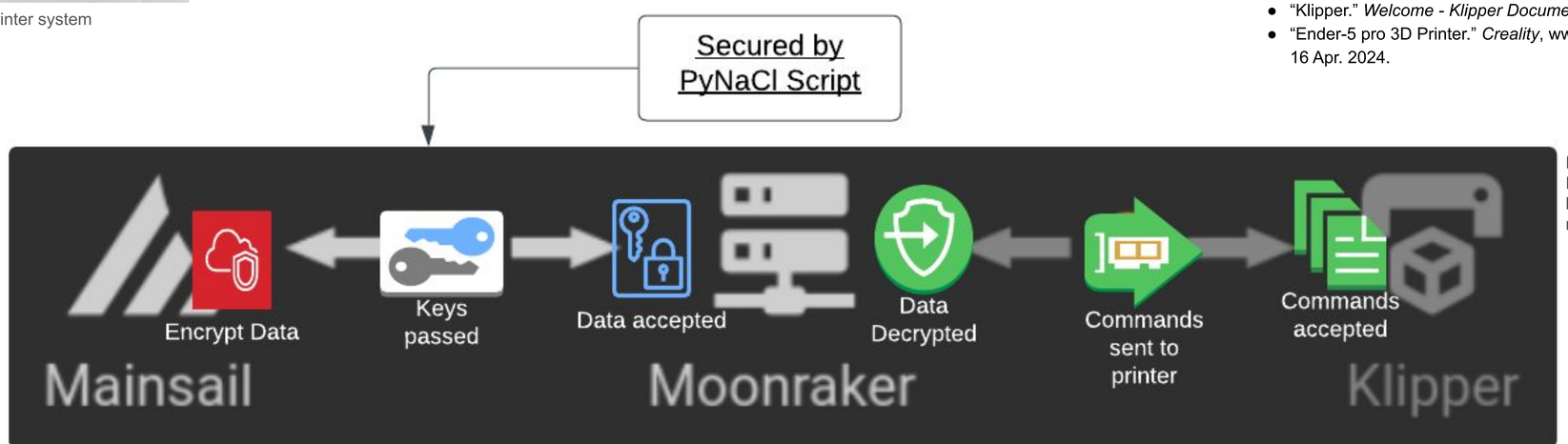


Figure 2: Methods process diagram
Figure from: https://www.reddit.com/r/klippers/comments/vm3y5e/mainsail_moonraker/

Future Steps

Once all the hardware is in place, I will test the printer with the PyNaCl script. I will test to ensure the encryption and signature process work correctly. I will send the encrypted data to the Moonraker API to see if it is properly decrypted. Lastly, I will ensure the firmware encodes the commands correctly

Conclusion

Using PyNaCl in Python to encrypt commands from MainSail UI to Moonraker API on a printer with Klipper firmware enhances security. Encrypting commands prevents unauthorized access or tampering, securing device and user data. PyNaCl encryption shields sensitive information, bolstering system security. This ensures secure communication, lowering risks of interference or breaches. Integrating PyNaCl encryption fortifies printer defense, assuring users the safety of their data and machine.

References

- Pyca, Contributors. “Pyca/Pynacl: Python Binding to the Networking and Cryptography (NaCl) Library.” *GitHub*, 21 Feb. 2013, github.com/pyca/pynacl.
- Contributors. “Python Binding to the Libsodium Library.” *PyNaCl*, Donald Stufft, 2013, pynacl.readthedocs.io/en/latest/.
- Mainsail-Crew. “Mainsail-Crew/Mainsail: Mainsail Is the Popular Web Interface for Managing and Controlling 3D Printers with Klipper.” *GitHub*, 7 Mar. 2020, github.com/mainsail-crew/mainsail.
- *Mainsail*, docs.mainsail.xyz/. Accessed 16 Apr. 2024.
- Arksine. “Arksine/Moonraker: Web API Server for Klipper.” *GitHub*, github.com/Arksine/moonraker. Accessed 16 Apr. 2024.
- Contributors. *Moonraker*, moonraker.readthedocs.io/en/latest/web_api/. Accessed 16 Apr. 2024.
- Klipper3d, Contributors. “Klipper3d/Klipper: Klipper Is a 3D-Printer Firmware.” *GitHub*, github.com/Klipper3d/klipper. Accessed 16 Apr. 2024.
- “Klipper.” *Welcome - Klipper Documentation*, www.klipper3d.org/. Accessed 16 Apr. 2024.
- “Ender-5 pro 3D Printer.” *Creality*, www.creality.com/products/ender-5-pro-3d-printer. Accessed 16 Apr. 2024.

PyNaCl “Salt” Encryption and Authentication for Networking to 3D Printer

Justin Hartberg

Mentors: Dr. Clemente Izurieta, Yvette Hastings

Introduction

In remote networking with 3D printers, ensuring secure and encrypted connections is crucial for maintaining confidentiality and integrity. To solve this issue, I am testing encryption and decryption capabilities using the PyNaCl library to look at the information passed between a computer system and a 3D printer (Figure 1). This proactive approach mitigates the risk of malicious commands being sent to the printer, safeguarding it and its environment from potential harm in both digital and physical domains. Encrypting the data and authenticating the user credentials improves the safety of the 3D printing setup and ensures that the information originates from the correct source and hasn't been modified by unauthorized parties.

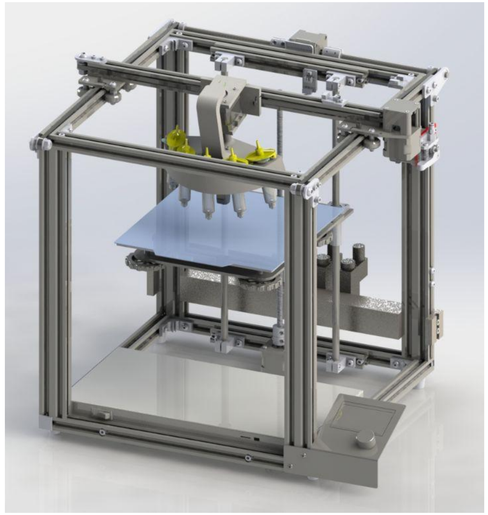


Figure 1: CAD Render of our 3D printer system

Methods

Steps (Figure 2):

- Initialize printer and software
- Modify Mainsail User Interface to send requests to my PyNaCl Flask server
  - This is done by modifying the javascript code that sends requests to the Moonraker and change it to send the information to the python script server.
- The data is encrypted in the python script, and then the encrypted data is sent to the Moonraker API
- The data is then decrypted on the printer in the Moonraker API, and then the printer command is given to the Klipper firmware.
- The printer reads the command and runs.

Pseudo Code:

- Import all necessary libraries (Flask, threading, socket, nacl*, requests)
- Generate public/private key pair for encryption for data from Mainsail UI
- Generate public/private key pair for Diffie-Hellman key exchange (prevent middle-man attacks)
- Create function to encrypt data using public key
- Create function to decrypt data using private key
- Create function to authenticate encrypted data by signature
- Load printer’s public key
- Set up Flask Server
- Handle incoming requests from Mainsail User Interface
- Encrypt data and sign secret box using private key
- Send encrypted data and signature to Moonraker API

Future Steps

Once all the hardware is in place, I will test the printer with the PyNaCl script. I will test to ensure the encryption and signature process work correctly. I will send the encrypted data to the Moonraker API to see if it is properly decrypted. Lastly, I will ensure the firmware encodes the commands correctly.

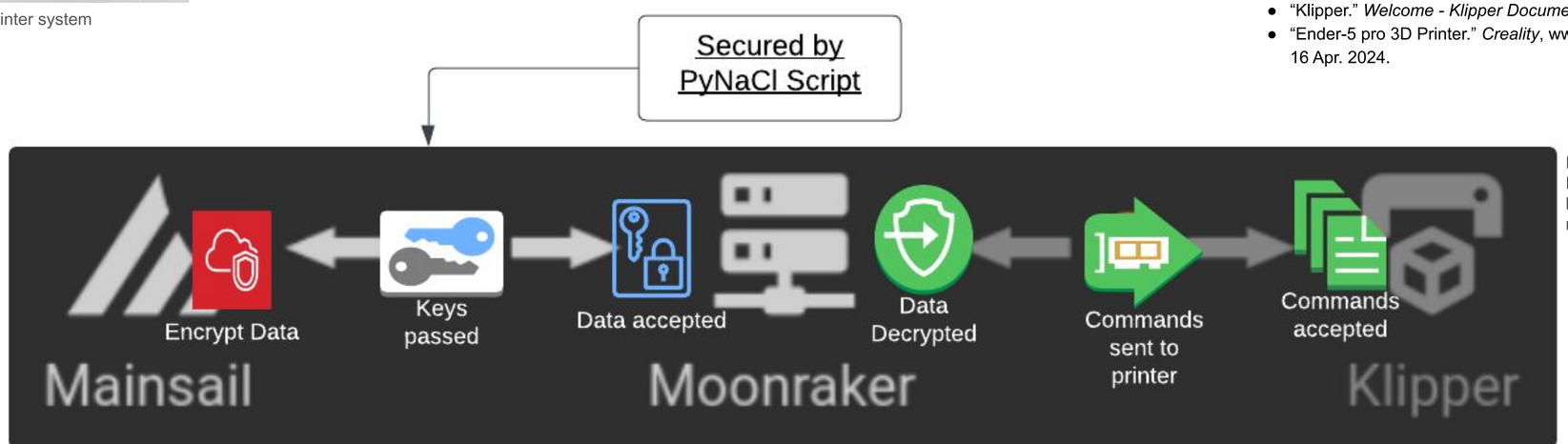


Figure 2: Methods process diagram
Figure from: https://www.reddit.com/r/klippers/comments/vm3y5e/mainsail_moonraker/

Conclusion

Using PyNaCl in Python to encrypt commands from MainSail UI to Moonraker API on a printer with Klipper firmware enhances security. Encrypting commands prevents unauthorized access or tampering, securing device and user data. PyNaCl encryption shields sensitive information, bolstering system security. This ensures secure communication, lowering risks of interference or breaches. Integrating PyNaCl encryption fortifies printer defense, assuring users the safety of their data and machine.

Acknowledgments

This work is supported by funding for the VICEROY Northwest Institute for Cybersecurity Education and Research (CySER) provided by The Office of the Undersecretary of Defense for Research and Engineering, in collaboration with the Air Force Research Laboratory and Griffiss Institute.

References

- Pyca, Contributors. “Pyca/Pynacl: Python Binding to the Networking and Cryptography (NaCl) Library.” *GitHub*, 21 Feb. 2013, github.com/pyca/pynacl.
- Contributors. “Python Binding to the Libsodium Library.” *PyNaCl*, Donald Stufft, 2013, pynacl.readthedocs.io/en/latest/.
- Mainsail-Crew. “Mainsail-Crew/Mainsail: Mainsail Is the Popular Web Interface for Managing and Controlling 3D Printers with Klipper.” *GitHub*, 7 Mar. 2020, github.com/mainsail-crew/mainsail.
- *Mainsail*, docs.mainsail.xyz/. Accessed 16 Apr. 2024.
- Arksine. “Arksine/Moonraker: Web API Server for Klipper.” *GitHub*, github.com/Arksine/moonraker. Accessed 16 Apr. 2024.
- Contributors. *Moonraker*, moonraker.readthedocs.io/en/latest/web_api/. Accessed 16 Apr. 2024.
- Klipper3d, Contributors. “Klipper3d/Klipper: Klipper Is a 3D-Printer Firmware.” *GitHub*, github.com/Klipper3d/klipper. Accessed 16 Apr. 2024.
- “Klipper.” *Welcome - Klipper Documentation*, www.klipper3d.org/. Accessed 16 Apr. 2024.
- “Ender-5 pro 3D Printer.” *Creality*, www.creality.com/products/ender-5-pro-3d-printer. Accessed 16 Apr. 2024.