QRS 2024
The 24th International Conference on
Software Quality, Reliability, and Security
July 1-5, 2024 • Cambridge, United Kingdom

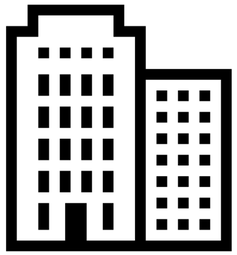
COLORADO STATE UNIVERSITY

The PIT-Cerberus Framework: Preventing Device Tampering During Transit

Rakesh Podder (Colorado State University, Ray's Lab)
Jack Sovereign (Colorado State University, Ray's Lab)
Dr. Indrajit Ray (Colorado State University, Ray's Lab)
Madhan B. Santharam (AMI US Holdings Inc.)
Stefano Righi (AMI US Holdings Inc.)

Motivation

DEVICES



COMPANY

(Point of origin)

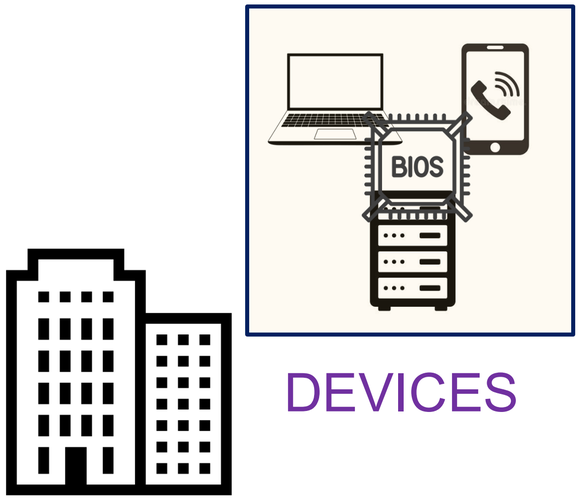
IN TRANSIT..

USER

(Point of use)

As a device (laptop, workstation, smartphone etc.) is shipped from the point of origin (Vendor) to point of use (Owner of device), there is a potential for tampering with the device and installation of malware.

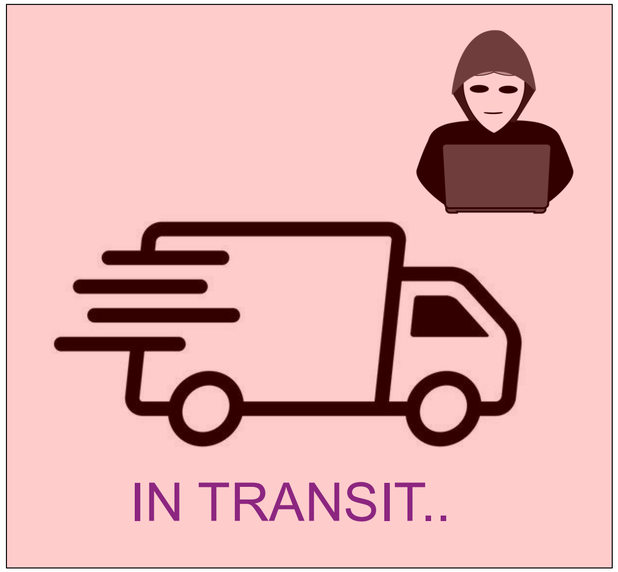
Problem Statement



DEVICES

COMPANY

(Point of origin)



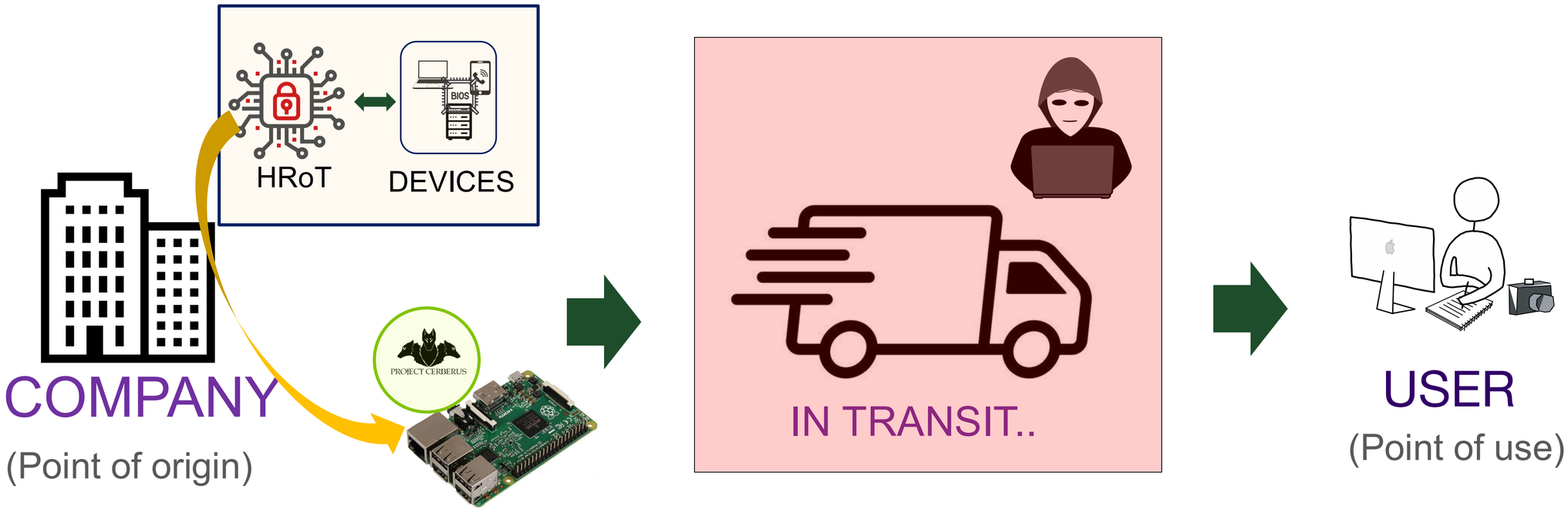
IN TRANSIT..

USER

(Point of use)

Prevent device tampering during transit so as to prevent malware installation

Solution



Lock device via BIOS/BMC during transit and allow to unlock only by an authorized user after proper authentication.

High-Level Description of PIT-Cerberus Protocol

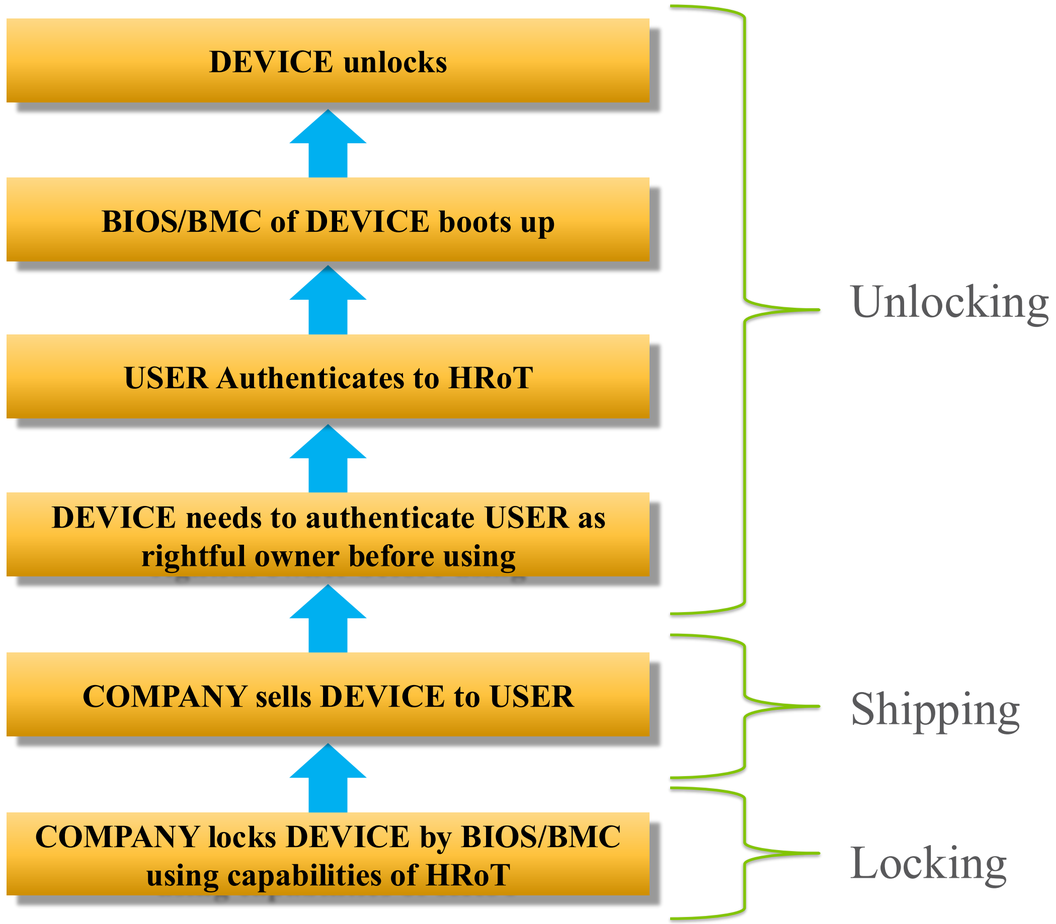


Figure: High-Level Workflow of PIT-Cerberus Protocol

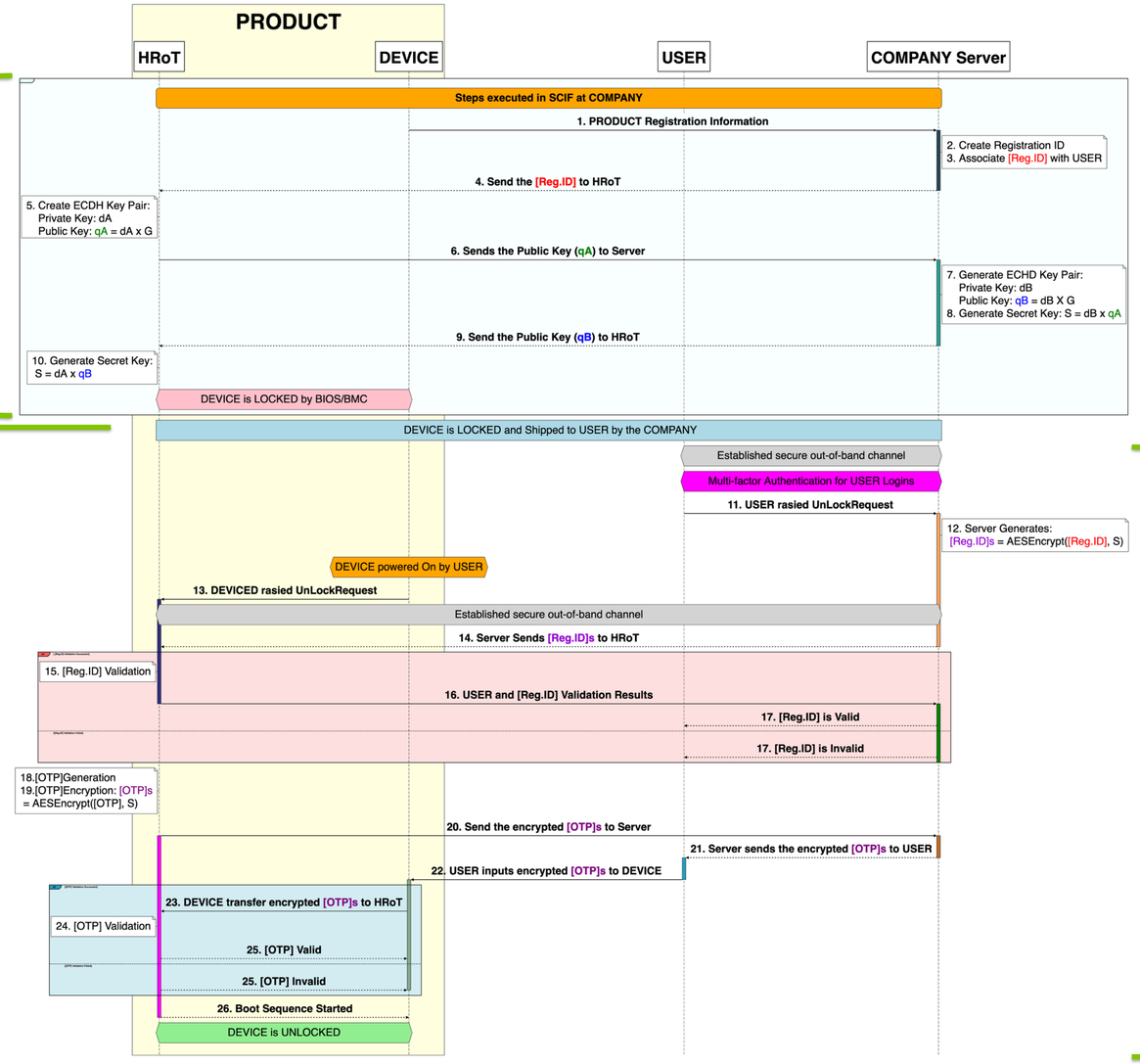
Our Contributions

1. We developed a protocol to prevent device tampering during transit.
2. The proposed PIT-Cerberus framework (PIT = Protection In Transit) leverages strong cryptographic techniques and has been implemented within a trusted microcontroller framework (Project Cerberus).
3. We enhance Microsoft's Cerberus Framework capabilities to carve out a more resilient security protocol.
4. We will contribute the PIT-Cerberus framework's libraries to Project Cerberus, an open-source project that offers a security platform for server hardware.
5. We have done extensive security testing to showcase the protocol's exceptions and failures handling capability in various attack scenarios.

Proposed Protocol

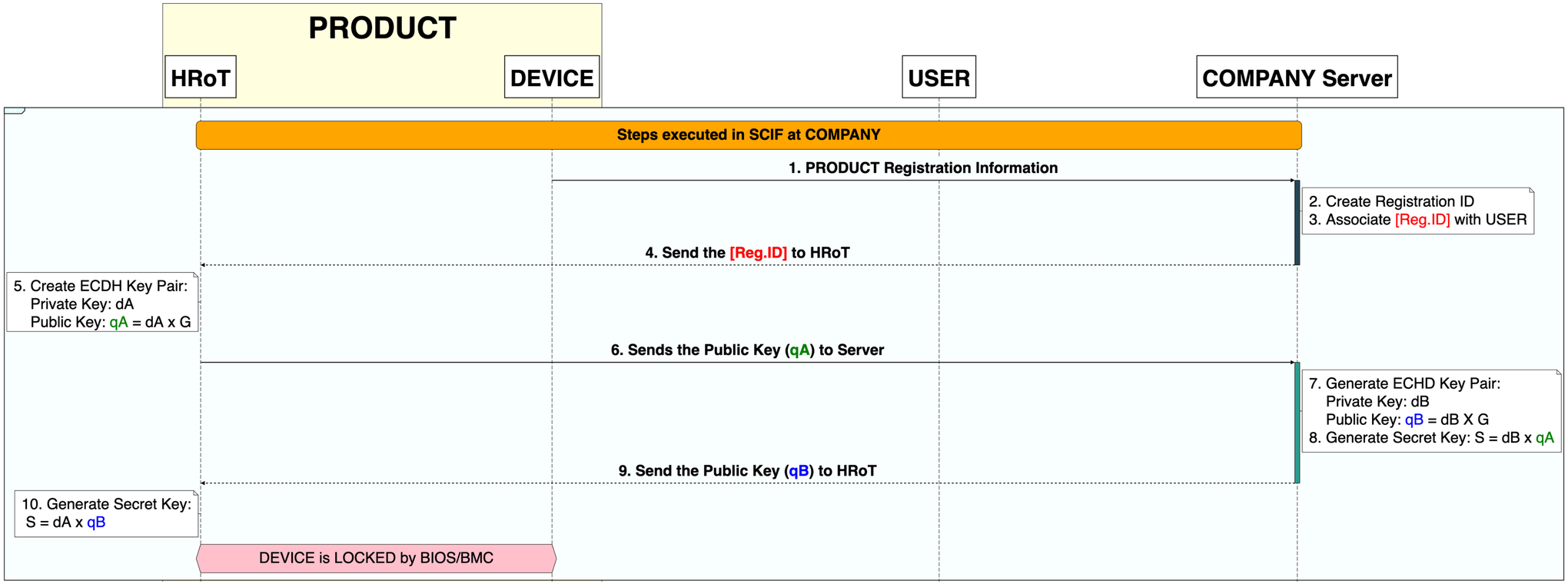
Device Locking Steps

Shipping



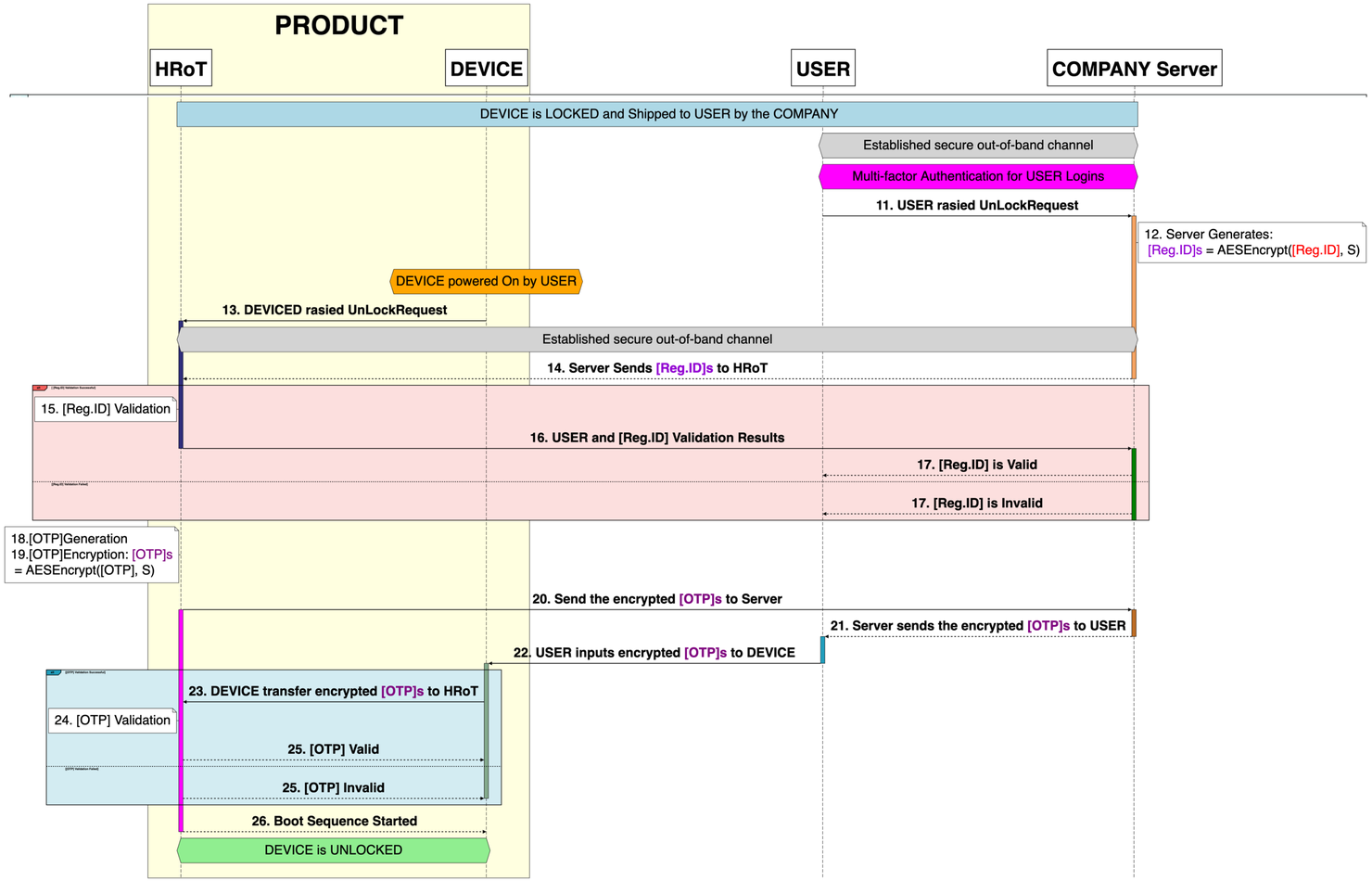
Device Unlocking Steps

Proposed Protocol: Locking



Device Locking Steps

Proposed Protocol: Unlocking



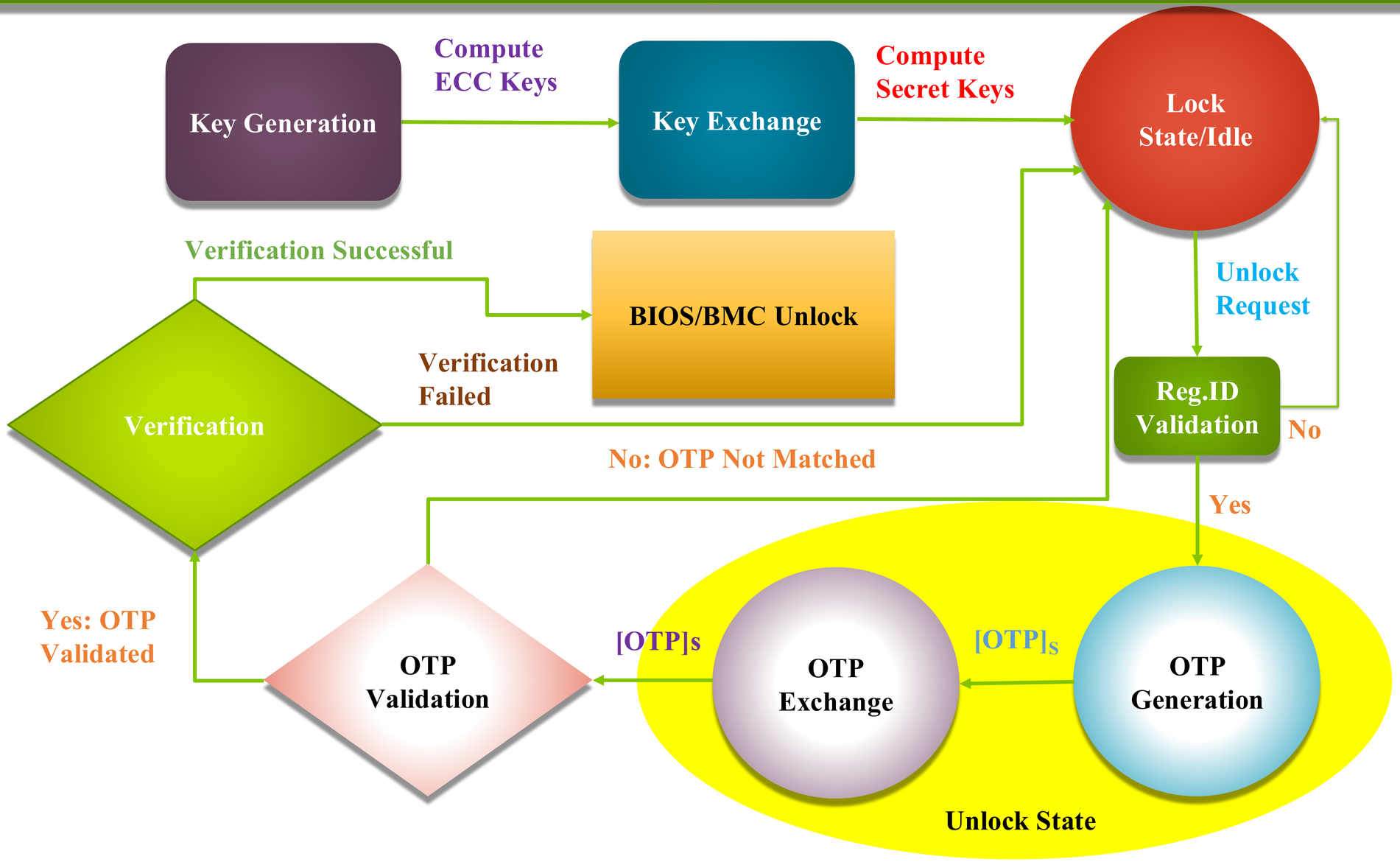
Device Unlocking Steps

PIT-Cerberus Formal Verifications

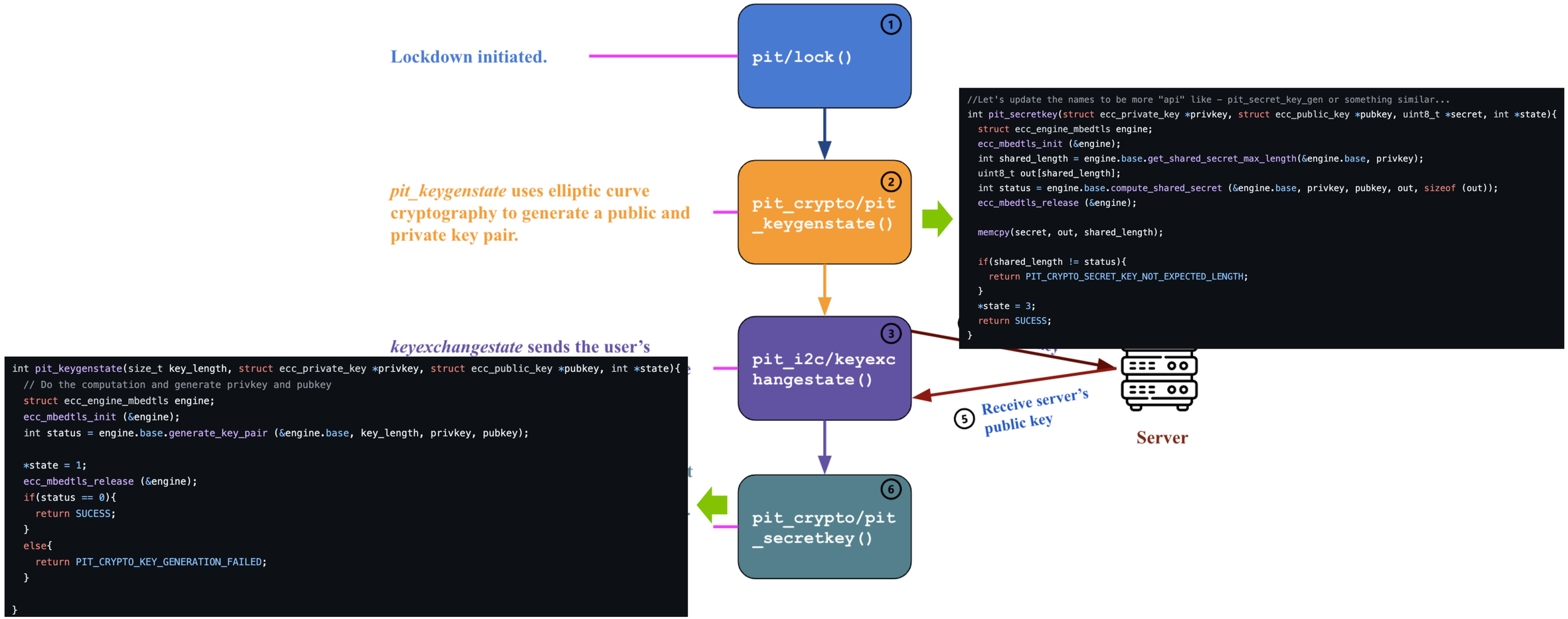
The PIT protocol establishes mutual authentication as it needs to satisfy the following conditions:

- The Recipient verifies the Device's authenticity, expressed as Recipient authenticates Device.
- The Device verifies the Recipient's authenticity, expressed as Device authenticates Recipient.
- The Recipient needs to authenticate themselves to the Server, expressed as Server authenticates Recipient.

PIT-Cerberus Control State Diagram



HRoT API Flowchart for Cerberus Open-Source Framework

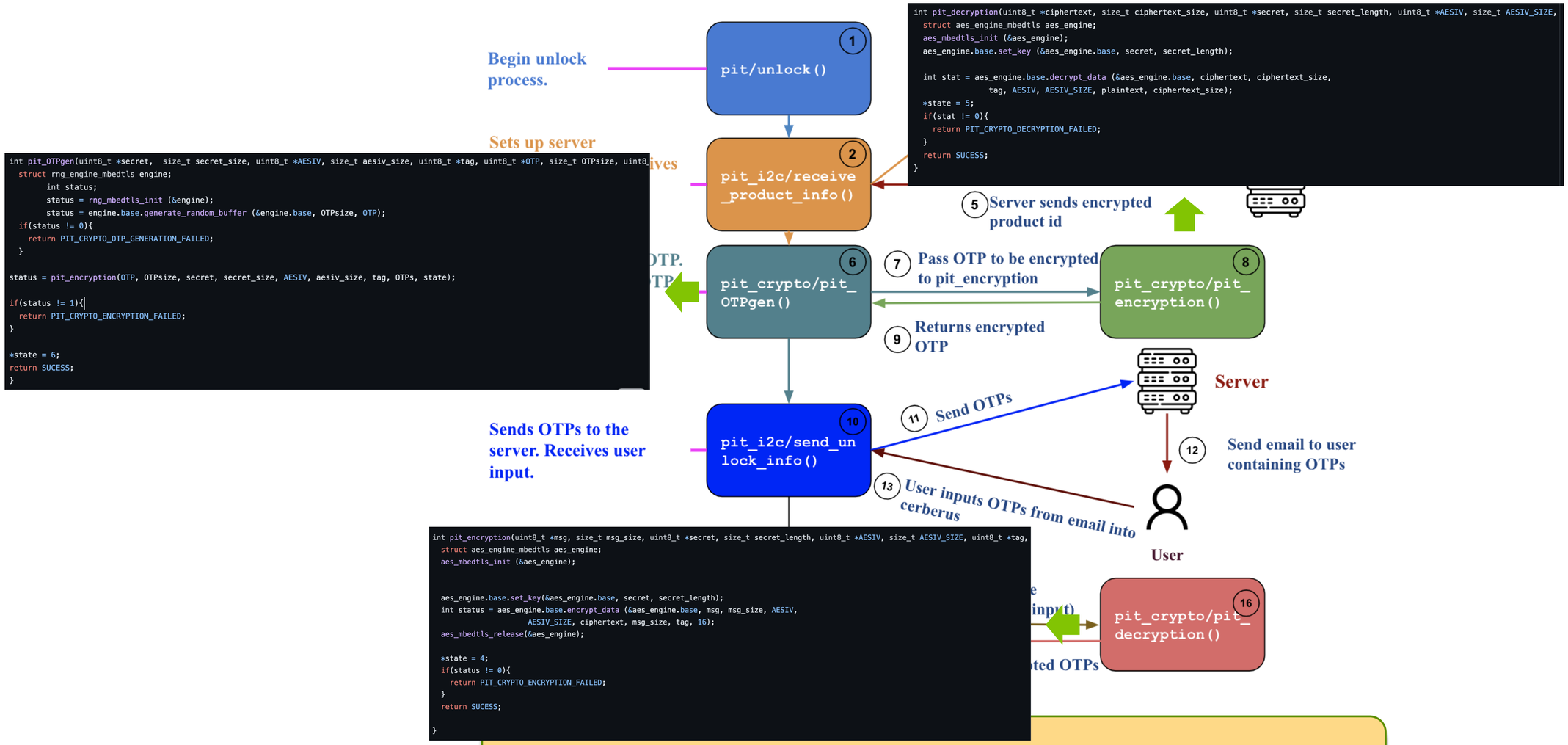



```
//Let's update the names to be more "api" like - pit_secret_key_gen or something similar...
int pit_secretkey(struct ecc_private_key *privkey, struct ecc_public_key *pubkey, uint8_t *secret, int *state){
  struct ecc_engine_mbedtls engine;
  ecc_mbedtls_init (&engine);
  int shared_length = engine.base.get_shared_secret_max_length(&engine.base, privkey);
  uint8_t out[shared_length];
  int status = engine.base.compute_shared_secret (&engine.base, privkey, pubkey, out, sizeof (out));
  ecc_mbedtls_release (&engine);

  memcpy(secret, out, shared_length);

  if(shared_length != status){
    return PIT_CRYPTO_SECRET_KEY_NOT_EXPECTED_LENGTH;
  }
  *state = 3;
  return SUCESS;
}
```

```
int pit_keygenstate(size_t key_length, struct ecc_private_key *privkey, struct ecc_public_key *pubkey, int *state){
  // Do the computation and generate privkey and pubkey
  struct ecc_engine_mbedtls engine;
  ecc_mbedtls_init (&engine);
  int status = engine.base.generate_key_pair (&engine.base, key_length, privkey, pubkey);

  *state = 1;
  ecc_mbedtls_release (&engine);
  if(status == 0){
    return SUCESS;
  }
  else{
    return PIT_CRYPTO_KEY_GENERATION_FAILED;
  }

}
```

API Flowchart for Lock state

HRoT API Flowchart for Cerberus Open-Source Framework



API Flowchart for Unlock state

Evaluation & Discussion

```
pit_crypto: test_OTPgen
pit_crypto: test_OTPvalidation
pit_crypto: test_decryption
pit: test_pit_lock
Device is Locked.
pit: test_pit_unlock
User initiated Unlock Request......
PRODUCT ID Validation Successful. pid_status is: 1
OTP Generation and Encryption Successful.
Please Enter your OTP:
b'\x03hs\x7f ?\x86\xfc\xa9\xfa\xb3\x12\xec\x1cGa1\x07^\xf3\x04\xca!\xa9]\x11\xea0wpb?1\x13\x16?\xf8\xc1\xac\x11\xe6\xda\x8c7\x07\xce\xedU
\x1d\xc9\tOaH\xa3\xdf\x98i\xb5AD\x1c?=V{\xef_\x0bx\x80\xcex\xb3\xed\x0e]9\x9f\xda\xe1W\x8e\xa2\r\xdb\xd54\xce\xdd\xe3\xe8q\x15\x00/b[0\xf
0\xc8\xc7\xee,\xce\x9f\x1b\xe6\xbd,\xcdY&\x1aj^c3\xbd\xef\x80\xc1\xd7~g\xc0\xcf\x1e'
Encrypted OTP sent to Server.
OTP Validation Successful. pid_status is: 1
```

Output From HRoT Terminal

Encrypted OTP sent to the User's Email

```
tallahassee:~/Rakesh$ python3 pid-server.py
Generated: Server ECC Key Pair.

Server X value : 768465705340819100253629315231074600515073783807449255146945688342708940824
Server y value : 109252678345389310469317484089853095411478063296788366629630286751158294466709
listening
accepted
Connected by ('127.0.0.1', 40564)
Recived: Cerberus ECC Key Pair.

Client X value : 864774840645440664077226990236261786998423991788212449554964793995626422512 39
Client y value : 457020251591109096967518400286968162193238137950203812681101274813747646578 07

Generating shared secret..... Successful.
listening
listening
accepted
Connected by ('127.0.0.1', 34996)
[DEMO]: Encrypted OTP : b'\x03hs\x7f ?\x86\xfc\xa9\xfa\xb3\x12\xec\x1cGa1\x07^\xf3\x04\xca!\xa9]\x11\xea0wpb?1\x13\x1
6?\xf8\xc1\xac\x11\xe6\xda\x8c7\x07\xce\xedU\x1d\xc9\tOaH\xa3\xdf\x98i\xb5AD\x1c?=V{\xef_\x0bx\x80\xcex\xb3\xed\x0e]9
\x9f\xda\xe1W\x8e\xa2\r\xdb\xd54\xce\xdd\xe3\xe8q\x15\x00/b[0\xf0\xc8\xc7\xee,\xce\x9f\x1b\xe6\xbd,\xcdY&\x1aj^c3\xbd
\xef\x80\xc1\xd7~g\xc0\xcf\x1e'
```

Output From Server Terminal

PIT Library Compatibility of OpenSSL and Python Server

```
Openssl attempting to read the server's public key

Public-Key: (256 bit)
pub:
    04:77:46:1f:52:9c:73:ca:9e:f1:2f:c7:cd:55:32:
    fd:94:76:ff:f1:18:ab:18:57:78:df:e3:d2:57:a4:
    9e:40:b0:34:95:ba:ec:ef:43:5c:fe:5c:4c:2b:6c:
    ee:df:cd:34:8f:da:c8:68:ff:41:c4:ed:10:71:a5:
    24:ec:26:97:d5
ASN1 OID: prime256v1
NIST CURVE: P-256

Openssl attempting to read the server's private key

Private-Key: (256 bit)
priv:
    be:bc:cf:59:55:fe:04:81:d9:a9:76:a4:1c:e6:ce:
    20:bb:ae:64:7b:4c:af:fa:45:03:37:0b:8a:98:43:
    f5:5f
pub:
    04:77:46:1f:52:9c:73:ca:9e:f1:2f:c7:cd:55:32:
    fd:94:76:ff:f1:18:ab:18:57:78:df:e3:d2:57:a4:
    9e:40:b0:34:95:ba:ec:ef:43:5c:fe:5c:4c:2b:6c:
    ee:df:cd:34:8f:da:c8:68:ff:41:c4:ed:10:71:a5:
    24:ec:26:97:d5
ASN1 OID: prime256v1
NIST CURVE: P-256
```

Secret Key computed by OpenSSL and Python Server

```
Openssl attempting to read the client's public key :

Public-Key: (256 bit)
pub:
    04:19:4e:f7:a9:c5:92:d5:05:b9:68:a9:9c:7c:09:
    f8:c6:cc:50:88:5c:23:f4:e1:a0:7f:8d:28:46:d2:
    5b:c1:34:bd:08:e1:92:e6:34:7f:4f:d7:24:c8:55:
    46:4b:b0:8e:a2:52:e9:39:80:83:e0:4d:f5:50:24:
    68:cb:2c:be:a1
ASN1 OID: prime256v1
NIST CURVE: P-256
```

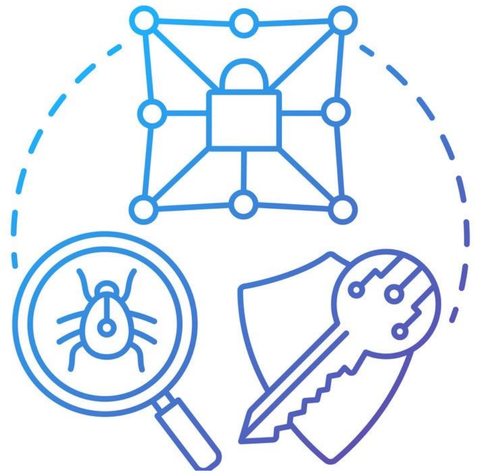
Cerberus Public Key in DER Format

```
Using openssl to derive shared secret :


Using the server to derive shared secret :


Is this shared secret the same as the one generated by client AND the same as the one generated by server? True
```

Secret Key computed by OpenSSL and Python Server

Security Analysis of the Protocol

- Attack Surface Analysis: Detailed Threat Modelling.
- Encryption and Key Management:
  - AES-GCM-256
  - ECDH-256
- Replay Attack Prevention: Integrated Lamport timestamps and unique session keys.
- Authentication and Integrity:
  - Digital Signature Algorithm (DSA).
  - Multi-Factor Authentication (MFA).

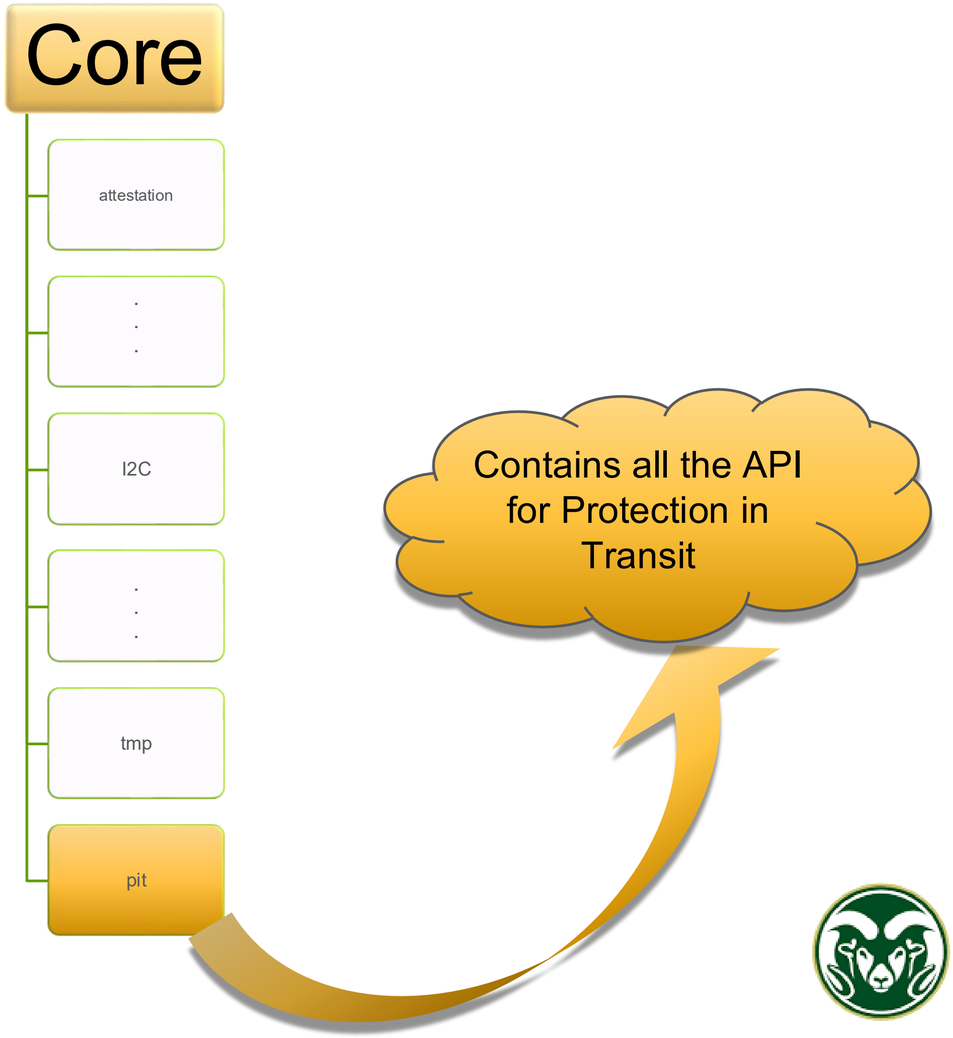


Security Analysis of the Protocol

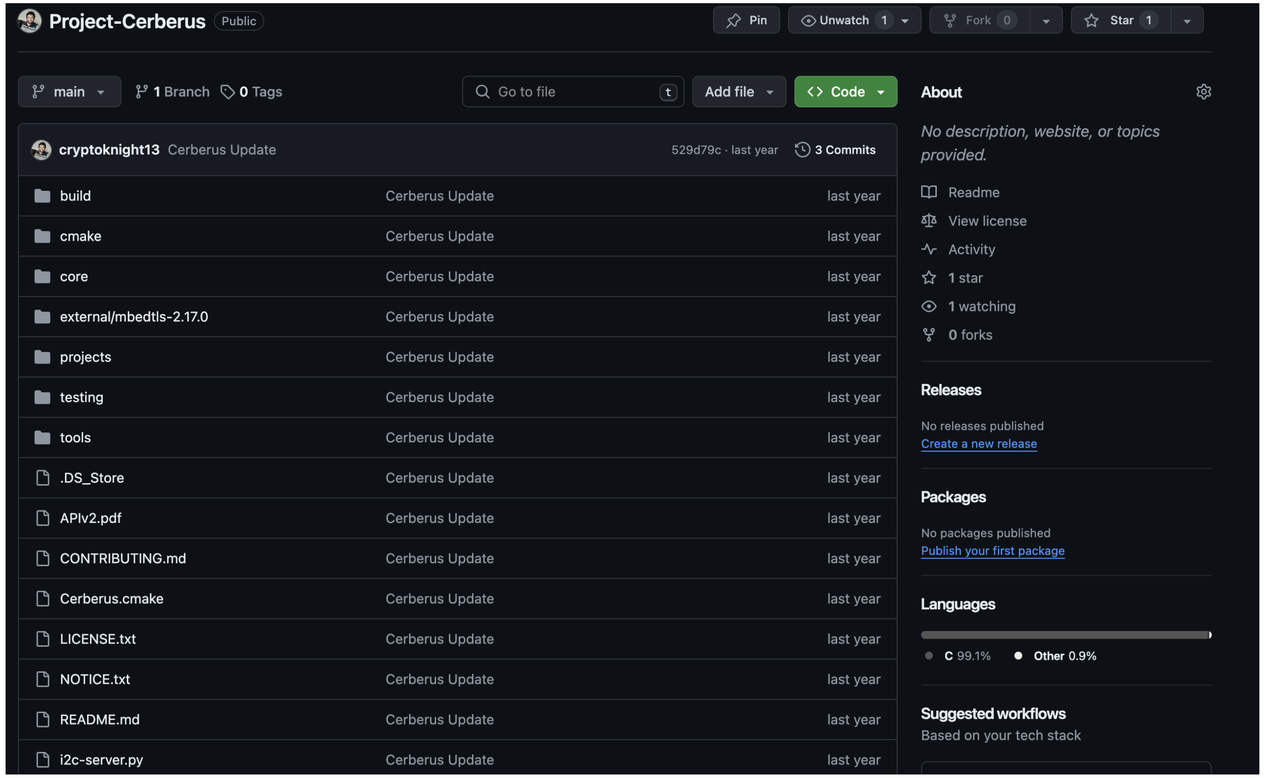
Protocol’s exceptions and failures handling capability

- Server unavailability.
- REG.ID validation fails.
- OTP validation fails.
- Unable to send OTPs to USER email.

PIT Modified Open-Source Cerberus Embedded Framework



Contains all the API for Protection in Transit



Link: https://github.com/cryptoknight13/Project-Cerberus.git

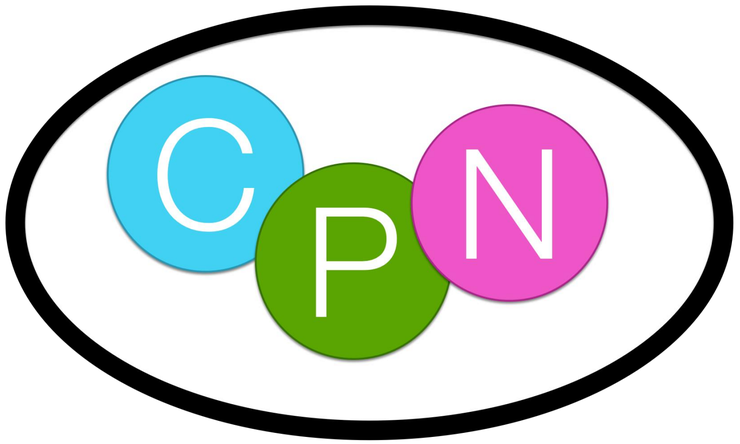
COLORADO STATE UNIVERSITY

American Megatrends

Microsoft

Future Work

- Enhancing PIT-Cerberus Capabilities:
  - Single User Multiple Sessions.
  - Multiple Users Multiple/Single Session(s).
- Verifying correctness of the protocol using Color Petri nets.
- Modification of OTP sending process.
- Hardware implementations and Firmware Update Ecosystems.



COLORADO STATE UNIVERSITY

Thank You

Question?