Module 2.2
AWS IoT Client Certificate using Zymbit

CompuThings *Technology;

Module Target

- Create Client certificate using Zymbit
- Create AWS IoT certificate using AWS CA
- Send data to AWS IoT

Create Client Cert Using Zymbit

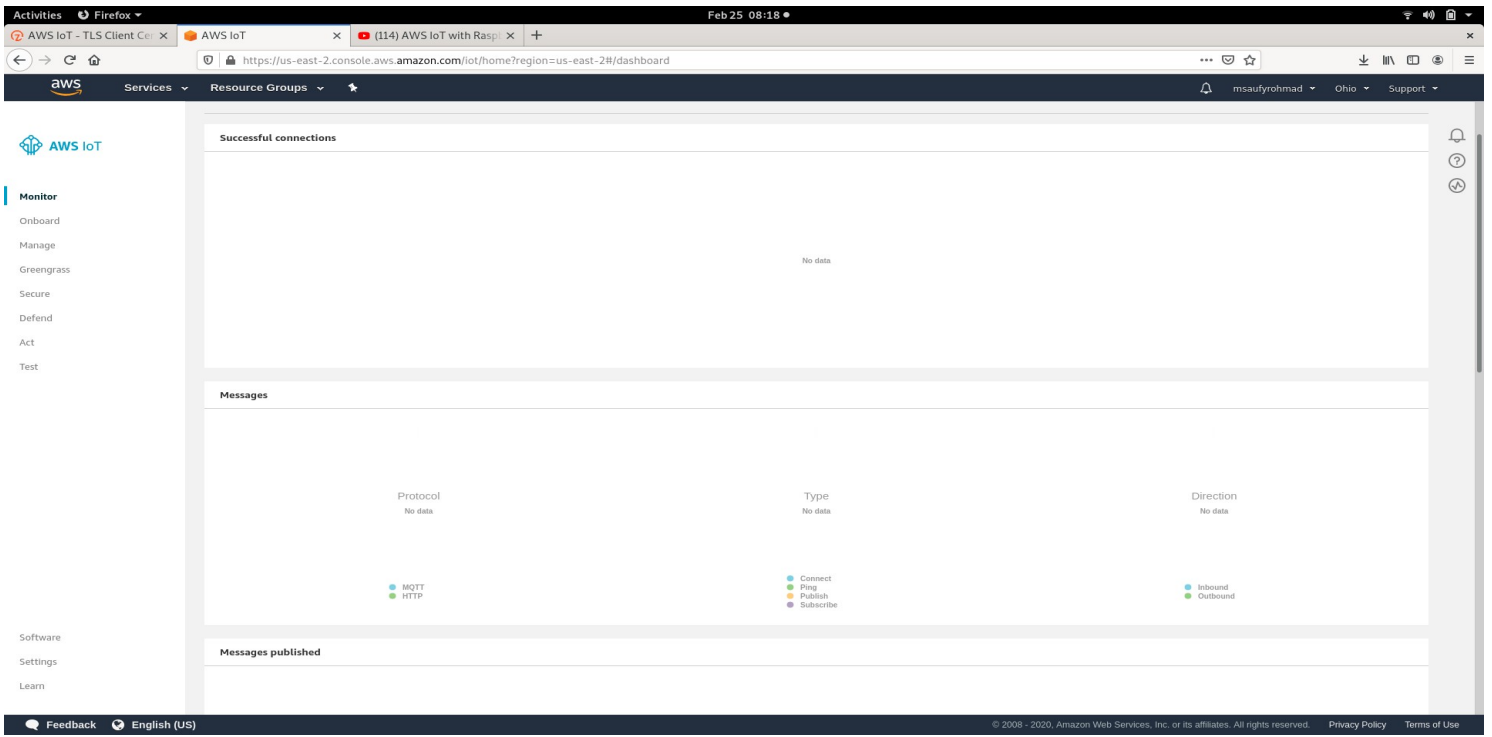
- openssl req -key nonzymkey.key -new -out zymkey.csr -engine zymkey_ssl -keyform e -subj "/C=US/ST=California/L=Santa Barbara/O=Zymbit/OU=Zymkey/CN=rpi.edge.zymbit.com"

Register with AWS CA

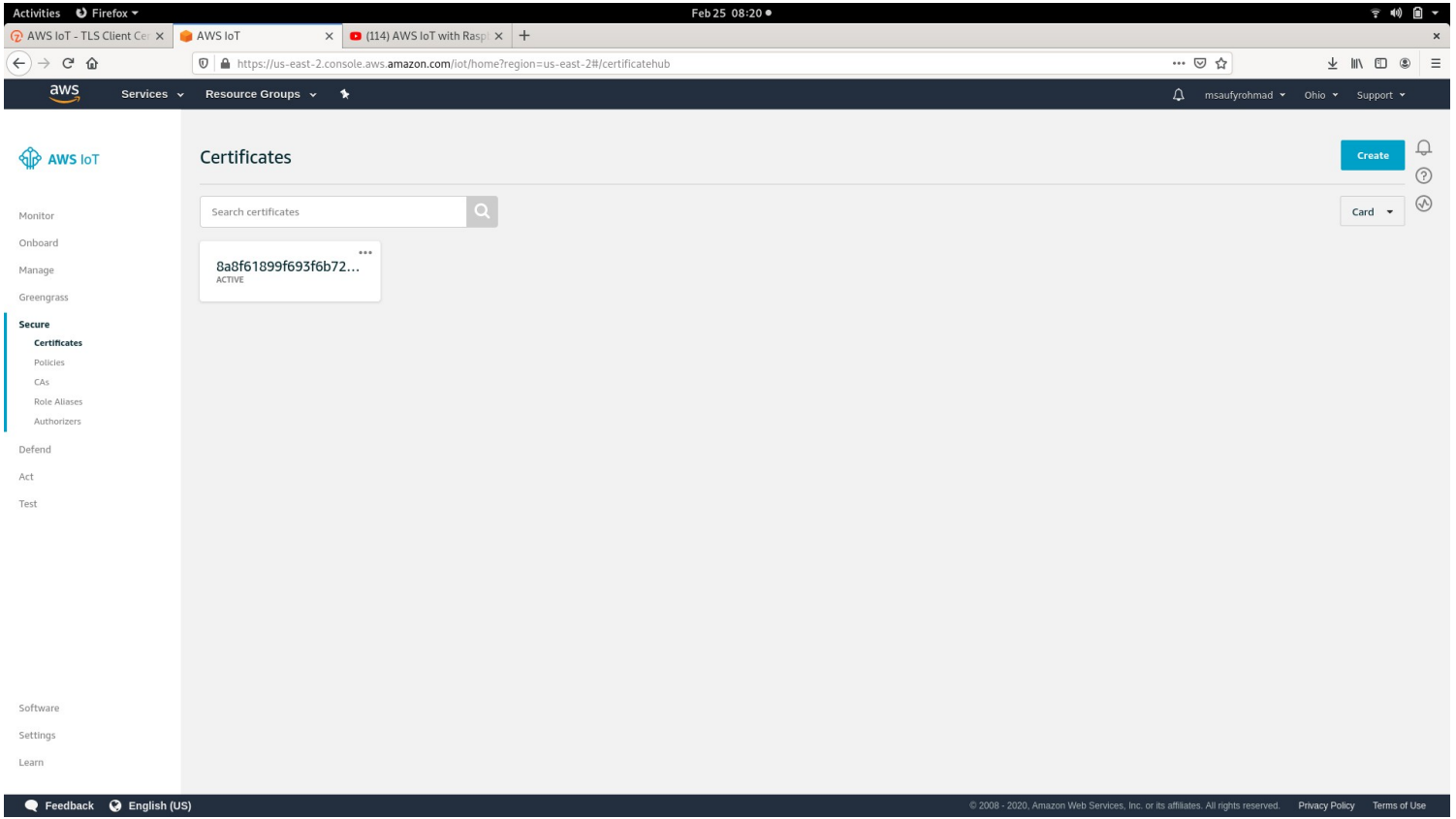
- Using AWS' Certificate Authority is the easier option in terms of setup and allows you to use a trustworthy Certificate Authority that Amazon uses its services.

Register With AWS IoT

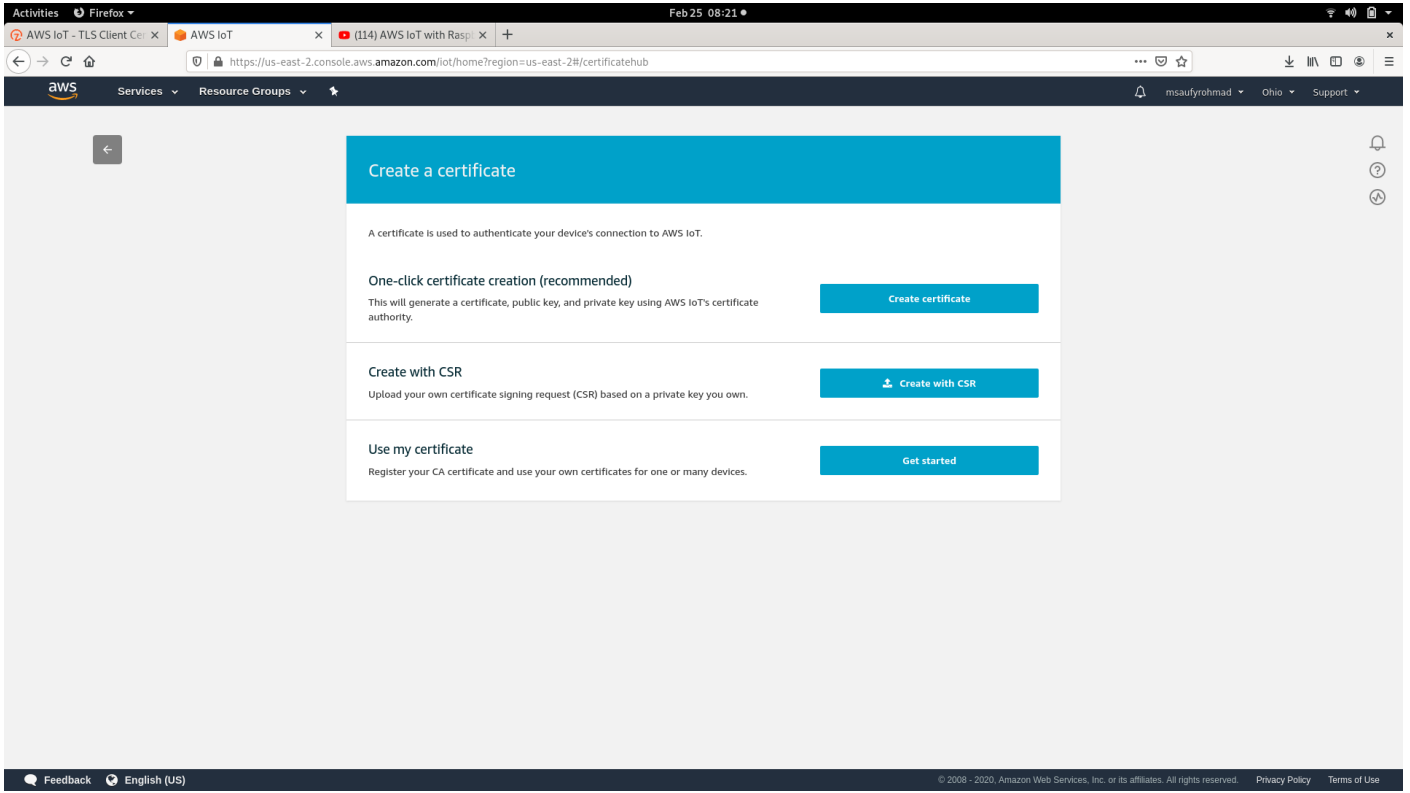
- Register Your Account and go to IoT Core



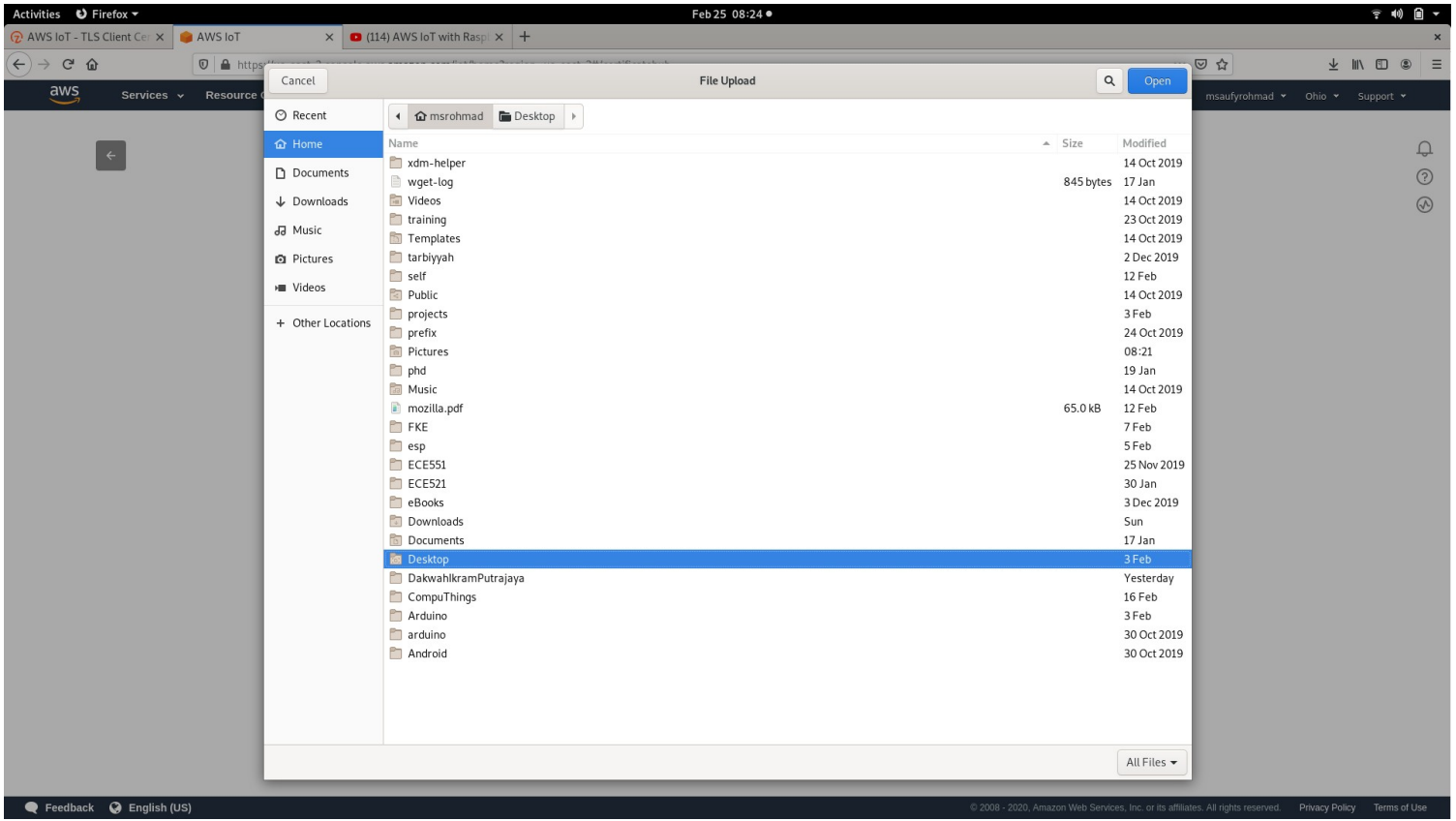
Go to Secure - > Certificate



Go to Create with CSR

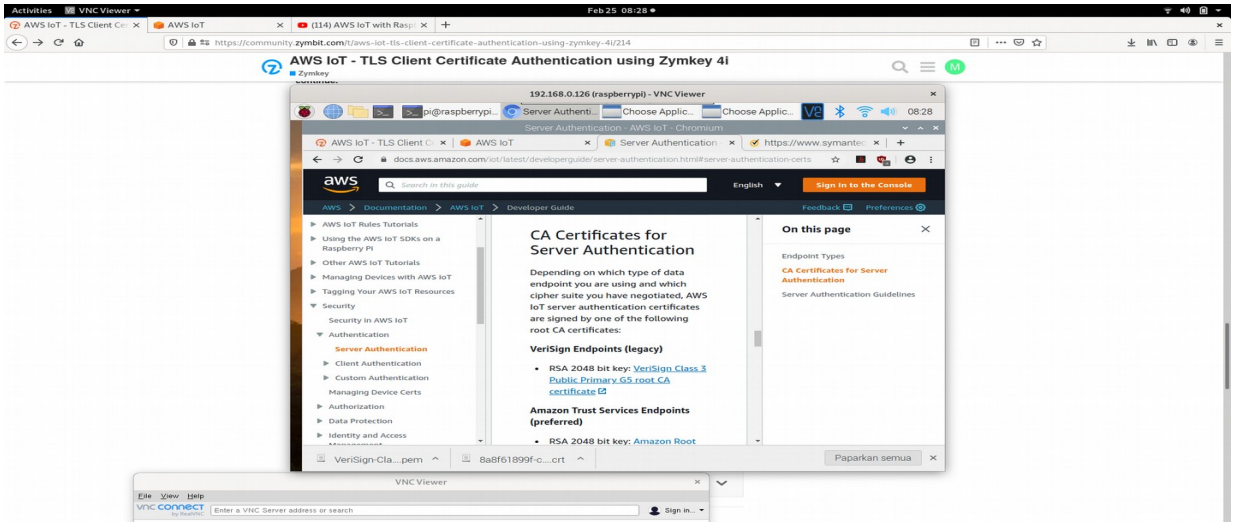


Point to Your zymkey.csr file



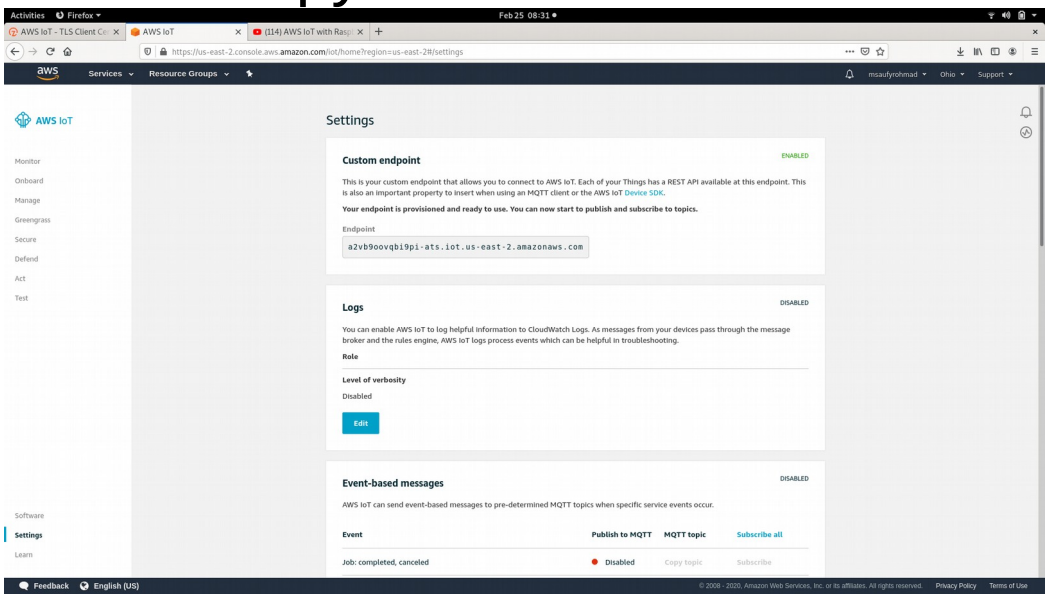
Download AWS Root CA

- Download the generated Cert. That is for you.
- Download server cert into AWS_CA.pem
- Put in same directory



Create AWS IoT Custom Endpoint

- IoT Core - > Setting
- Look into custom endpoint and copy it.



Run Script to Connect and Send Data

- curl --tlsv1.2 --cacert **CA_files/zk_ca.pem** --cert **zymkey.crt** --key nonzymkey.key --engine zymkey_ssl --key-type ENG -v -X POST -d "{ \"hello\": \"world\"}" " https://endpoint.iot.region.amazonaws.com:8443/topics/hello/world"
- Change the first one with AWS_CA.pem that you get
- Change the second bold with cert from AWS. (rename it to zymkey.crt to make it easy)
- Change the end point to the one that copy from custom endpoint in previous slide

Give Permission

- chmod 755 to zymkey.crt

Run the full command

- pi@raspberrypi:~/zymbitsecureiot $ curl --tlsv1.2 --cacert AWS_CA.pem --cert zymkey.crt --key nonzymkey.key --engine zymkey_ssl --key-type ENG -v -X POST -d "{ \"hello\": \"world\"}" "https://a2vb9oovqbi9pi-ats.iot.us-east-2.amazonaws.com:8443/topics/hello/world"

Verify the Secure connection

```
* Connected to a2vb9oovqbi9pi-ats.iot.us-east-2.amazonaws.com (2600:1f00:6000::12da:570b) port 8443 (#0)
* ALPN, offering h2
* ALPN, offering http/1.1
* successfully set certificate verify locations:
*   CAfile: AWS_CA.pem
  CApath: /etc/ssl/certs
* TLSv1.3 (OUT), TLS handshake, Client hello (1):
* TLSv1.3 (IN), TLS handshake, Server hello (2):
* TLSv1.2 (IN), TLS handshake, Certificate (11):
* TLSv1.2 (IN), TLS handshake, Server key exchange (12):
* TLSv1.2 (IN), TLS handshake, Request CERT (13):
* TLSv1.2 (IN), TLS handshake, Server finished (14):
* TLSv1.2 (OUT), TLS handshake, Certificate (11):
* TLSv1.2 (OUT), TLS handshake, Client key exchange (16):
* TLSv1.2 (OUT), TLS handshake, CERT verify (15):
* TLSv1.2 (OUT), TLS change cipher, Change cipher spec (1):
* TLSv1.2 (OUT), TLS handshake, Finished (20):
* TLSv1.2 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.2 / ECDHE-RSA-AES128-GCM-SHA256
* ALPN, server did not agree to a protocol
* Server certificate:
```

Verify the Secure connection...2

```
* TLSv1.2 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.2 / ECDHE-RSA-AES128-GCM-SHA256
* ALPN, server did not agree to a protocol
* Server certificate:
*  subject: CN=*.iot.us-east-2.amazonaws.com
*  start date: May  7 00:00:00 2019 GMT
*  expire date: Apr 10 12:00:00 2020 GMT
*  subjectAltName: host "a2vb9oovqbi9pi-ats.iot.us-east-2.amazonaws.com" matched cert's "*.iot.us-east-2.amazonaws.com"
*  issuer: C=US; O=Amazon; OU=Server CA 1B; CN=Amazon
*  SSL certificate verify ok.
> POST /topics/hello/world HTTP/1.1
> Host: a2vb9oovqbi9pi-ats.iot.us-east-2.amazonaws.com:8443
> User-Agent: curl/7.64.0
> Accept: */*
> Content-Length: 19
> Content-Type: application/x-www-form-urlencoded
>
* upload completely sent off: 19 out of 19 bytes
< HTTP/1.1 403 Forbidden
< content-type: application/json
< content-length: 65
< date: Tue, 25 Feb 2020 00:34:58 GMT
```

Verify the Secure connection...3

```
> POST /topics/hello/world HTTP/1.1
> Host: a2vb9oovqbi9pi-ats.iot.us-east-2.amazonaws.com:8443
> User-Agent: curl/7.64.0
> Accept: */*
> Content-Length: 19
> Content-Type: application/x-www-form-urlencoded
>
* upload completely sent off: 19 out of 19 bytes
< HTTP/1.1 403 Forbidden
< content-type: application/json
< content-length: 65
< date: Tue, 25 Feb 2020 00:34:58 GMT
< x-amzn-RequestId: d5ffa7e4-a8c9-a22a-18fe-178471066d96
< connection: keep-alive
< x-amzn-ErrorType: ForbiddenException:
<
* Connection #0 to host a2vb9oovqbi9pi-ats.iot.us-east-2.amazonaws.com left intact
{"message":null,"traceId":"d5ffa7e4-a8c9-a22a-18fe-178471066d96"}
```

Conclusion

- This is another demonstration of how zymkey can be use as hardware root of trust for normal security services.