Integrating Intel TDX remote attestation into the secure shell protocol

Fabian Wesemann
firstname.lastname@stud.hs-flensburg.de
Flensburg University of Applied Sciences
Flensburg, Schleswig-Holstein, Germany

Abstract

Intel Trust Domain Extensions (TDX) allow the creation of Trusted Execution Environments (TEE) to prevent third parties from accessing the workload's memory. Integrating the remote attestation capability of Intel TDX into the SSH protocol offers a way to allow users or software to restrict connections to servers running inside a Trusted Domain, thereby protecting intellectual property, user data or meeting regulation criteria.

This paper presents the design of a challenge response protocol as well as a prototype implementation based on OpenSSH. It features the creation of Intel TDX quotes, issuance of a derived JWT using Azure attestation services and client-side validation of signatures and claims, integrated into the SSH connection process.

1 Introduction

In traditional cloud computing settings, the platform owner has significant control over executed workloads. Virtual Machines (VM) allow the separation of different tenants on the same host, but the workloads are still accessible for privileged software required to execute the VM, e.g. the hypervisor [7]. In addition to disk encryption (protecting *data at rest*) and protocols like SSH and TLS (protecting *data in transit*), Confidential Computing aims to protect *data in use*, which is stored in RAM.

The following sections explain how two existing solutions for the protection of *data in transit* and *data in use* can be combined using the remote attestation feature of Intel TDX.

2 Motivation and use cases

Secure communication between multiple machines over a network is crucial for modern day computing. In addition to a secure connection that protects the data against eavesdroppers, there are scenarios where it may also be desirable to enforce client side rules on the other side of a network connection, like the presence of a Trusted Execution Environment.

Those scenarios include, but are not limited to

- transfering source code including intellectual property to a remote git repository
- tunneling network traffic to a remote database to store sensitiv data
- transfering backups to another datacenter

As we will see in section 3.3, SSH is widely used in the above scenarios. Implementing remote attestation at this level allows the adoption with different software and use cases without having to add Confidential Computing awareness manually to every piece of software.

3 Background

3.1 Intel TDX

The Intel Trust Domain Extensions (TDX) offer hardware based security improvements, allowing the creation of Virtual Machines in a secure environment, so called Trusted Domains[7]. By encrypting a Virtual Machines memory using keys only accessible to the CPU, the workload is isolated and can not be inspected or manipulated by a cloud service provider or other tenants.

Compared to traditional cloud environments, a VM is susceptible to manipulation or inspection by third parties. The cloud providers employees, infrastructure and software anywhere in the stack from the bios over to the hypervisor managing the resources are all part of the trust boundary [7]. Confidential Computing technology like Intel TDX works at the VM level by protecting integrity and limiting the access to the memory using logical and cryptographic measures, effectively limiting the trusted components[7].

3.2 Remote Attestation

To verify that a workload (in our case the SSH server) is running in a Trusted Domain, a relying party can use the remote attestation feature included in TDX [7]. Inside the Trusted Execution Environment, measurements of the TD are gathered as a *report*. Using the Intel SGX Attestation, a *quote* is generated from this report[4]. The authenticity of the quote and the reported data can be verified using the Intel Trust Authority[8].

For Intel TDX capable VMs on the Microsoft Azure platform, the remote attestation works different. To generate the report, the Azure preview version of the Intel Trust Authority CLI tool is needed[9]. The resulting *quote* and collected runtime information is then sent to the Azure attestation service to retrieve a JSON Web Token signed by Azure, which can then be used as proof for the relying party to validate the integrity of the Trusted Execution Environment.

In addition to the collected runtime information, arbitrary data can be specified when generating the quote. The sha512 hash of this data will be included in the signed JWT.

For this project, Azure Documentation on SGX Attestation[1] and an official TDX example[3] where used to implement the attestation.

3.3 Secure Shell Protocol

The Secure Shell protocol (SSH) is used to establish secured communication over TCP/IP between two computers. Use cases include shell access (for administration and deployment), tunneling (e.g. to access a remote database not exposed to the internet) or the transfer of files and data via a secure channel for other software like *git* or

rsync. SSH uses public key cryptography to protect the traffic as well as authenticating identities of the server and the client resp. the user[20].

The addition of remote attestation adds an extra layer of security, since not only the identity of the server but also the environment in which it operates in is validated against the clients expectations.

There are different implementations of SSH. This project focuses on the widely adopted[17] *OpenSSH*, which provides (among others) the necessary client (`ssh`) and server (`sshd`)[15].

4 Design

The SSH protocol supports a *services* mechanism. A client can request a service from the server. The server must either *accept* the request or terminate the connection if the service is not supported [14, Sec. 10], which is a good fit for mandatory attestation. According to the naming conventions in [14], the new service implementation is named `ra-ssh-attestation`.

The SSH connection is initiated by the client. All attestation related communication happens after the key exchange and user authentication, but before the session is fully established. This prevents reconnaissance attacks, as the content of the attestation report is not observable by parties monitoring network traffic or attackers trying to request attestation without being able to authorize themselves as legitimate users of the system.

After the servers identity is verified using the *hostkey* and the client has authenticated itself using one of the supported methods (e.g. password or public key), the client requests the `ra-ssh-attestation` service.

The Server answers this request by sending a `SSH2_MSG_SERVICE_ACCEPT` message and the service name as specified in [14]. When the request is accepted, the client sends an `RA_SSH_TOKEN_REQUEST` message with a randomly generated nonce. The server generates a proof to attest the Trusted Execution Environment including the nonce and sends it to the client for verification in a `RA_SSH_TOKEN_RESPONSE` message.

On the challenging side, the client receives the proof and validates it. This includes checking the presence of an Intel TDX TEE as well as verifying that nonce sent with the clients request was used to generate the report.

Failure during the attestation process terminates the connection. After the client finished the validation, both sides proceed with the usual SSH connection process.

5 Implementation

The implementation is based on the *OpenSSH*[15] project. The SSH server - running in the Trusted Domain - takes the role of the *attester*. The SSH client is the *challenger* and will only establish a connection when the server is able to attest the Trusted Execution Environment.

5.1 Environment

Development and testing was done using an Azure Standard DC2es v5 Confidential VM[6] running Ubuntu 22.04.

The `intel-trustauthority-cli` version is 1.4.0.

5.2 Quote and token generation

The server generates an Intel TDX Report by invoking the `intel-trustauthority-cli` command, passing a randomly generated nonce as `user_data` (fig. 1, Step 4). The path to the binary can be configured in the `sshd_config` file. Source of randomness for nonce generation is `arc4random_buf`, as done in other parts of OpenSSH.

To obtain a JSON web token that can be validated by the client, the server sends the generated quote and runtime data (containing the nonce) to an Azure attestation endpoint using the *libcurl* C library[19] (see fig. 1, step 5). The API URL was taken from the Azure TDX example application[3, `maa_config.json`]. For the issued token, see listings 1 and 3. The JWT is then sent to the challenging SSH client in the `RA_SSH_TOKEN_RESPONSE` message (fig. 1, step 7, 8).

5.2.1 Privilege separation. For security reasons, `sshd` runs SSH sessions under a separate, unprivileged user account until the authentication is completed. The unprivileged child can communicate with the parent `sshd` process using the `monitor` implementation which passes requests and replies between the two (see [16], `monitor.c, monitor.h`). Since the quote generation needs root privileges, the `MONITOR_REQ_RA_SSH_TOKEN` and `MONITOR_ANS_RA_SSH_TOKEN` messages were added to generate the quote with the required privileges and pass the result back to the unprivileged child.

5.3 Verifying the evidence

The verification is implemented in `azure_attestation_client.c` and exposed as a single function `validate_azure_jwt` accepting the token (received from the server) and the nonce (as picked by the client). It returns a single value indicating success or failure.

To parse and validate the JWT, the *libjwt* C library[5] is used with a custom `jwt_key_p_t` key provider[13] implementation to fetch the Azure signing keys with *libcurl*[19] using the information contained in the JWT header (listing 1). Extraction of the attestation information from the claims is done using *jansson*[12], which is also used internally by *libjwt*.

5.3.1 Signature. The process of verifying an Azure JWT signature is based on the description in [11].

First, the Azure X509 certificate must be fetched using the endpoint (`jku`) and key identifier (`kid`) provided in the JWT header (listing 1, (fig. 1, step 9). The client should ensure that the certificate is provided by Azure.

The OpenSSL library[18] is used to get the public key from the certificate. Signature verification is done by *libjwt* in the decoding step.

5.3.2 Claims. After the JWT signature is verified, the client checks whether the claims (listing 3) match the expected values seen in table 1 and verifies the existence of a Trusted Execution Environment [2].

5.3.3 Nonce. Additionally, the `x-ms-runtime.user_data` field (listing 3) contains the SHA512 hash of the `user-data` provided when the quote was generated using the `intel-trustauthority-cli`, which the client expects to match the SHA512 hash of the nonce it

Key	Value
`eat_profile`	`https://aka.ms/maa-eat-profile-tdxvm`
`x-ms-attestation-type`	`tdxvm`
`x-ms-compliance-status`	`azure-compliant-cvm`
`x-ms-runtime.user_data`	$SHA512(nonce)$

Table 1: JWT claims and their expected values [2]

sent along with the `RA_SSH_TOKEN_REQUEST` (see fig. 1, step 3). Note that, while the user data is specified in base64 at quote generation, the hash is calculated on the raw data.

```
{
  "alg": "RS256",
  "jku": "https://sharedeus2e.eus2e.attest.azure.net/certs",
  "kid": "6qubGPaYpJMjCD9chNyuh/ztq87166pwivQJz1quFRQ=",
  "typ": "JWT"
}
```

Listing 1: Azure JWT header

6 Related work

Knauth et al. [10] integrated Intel SGX attestation into TLS. Using X509 extensions, the attestation information is added to the existing certificates. On the client side, the relying party extracts and verifies the information.

7 Conclusions

This work demonstrates how remote attestation can be integrated into the existing SSH protocol. The prototype implements the challenge-response protocol, including the generation of the report, integration of the Azure Attestation Service for JWT creation and the verification of the presented evidence in the challenging client.

8 Source

The implementation is published at https://github.com/tuftedd eer/openssh-tdx-remote-attestation/. The repository includes instructions to set up the virtual machine on Microsoft Azure as well as a Docker image containing the modified `ssh` and `sshd` with all dependencies needed.

References

[1] 2022. Attestation - attest sgx enclave. https://learn.microsoft.com/en-us/rest/api/attestation/attestation/attest-sgx-enclave?view=rest-attestation-2022-08-01&tabs=HTTP. Accessed: 2024-08-18. (Aug. 2022).

[2] 2024. Azure attestation eat profile for intel® trust domain extensions (tdx). https://learn.microsoft.com/en-us/azure/attestation/trust-domain-extensions-eat-profile. Accessed: 2024-09-15. (Aug. 2024).

[3] [n. d.] Azure tdx attestation example. https://github.com/Azure/confidential-computing-cvm-guest-attestation/tree/tdx-preview/tdx-attestation-app. Accessed: 2024-08-18. ().

[4] Pau-Chen Cheng, Wojciech Ozga, Enriquillo Valdez, Salman Ahmed, Zhongshu Gu, Hani Jamjoom, Hubertus Franke, and James Bottomley. 2024. Intel tdx demystified: a top-down approach. *ACM Comput. Surv.*, 56, 9, Article 238, (Apr. 2024), 33 pages. DOI: 10.1145/3652597.

[5] [SW] Ben Collins, libjwt version 1.17.2. VCS: https://github.com/benmcollins/libjwt.

[6] 2022. Dcesv5 and dcedsv5-series confidential vms. https://learn.microsoft.com/en-us/azure/virtual-machines/dcesv5-dcedsv5-series. Accessed: 2024-08-26. (Aug. 2022).

[7] Intel. 2022. Intel® trust domain extensions white paper. https://www.intel.com/content/dam/develop/external/us/en/documents/tdx-whitepaper-final9-17.pdf.

[8] [n. d.] Intel® trust authority. https://docs.trustauthority.intel.com/main/articles/introduction.html. Accessed: 2024-08-18. ().

[9] [n. d.] Intel® trust authority cli for intel tdx (azure preview). https://github.com/intel/trustauthority-client-for-go/tree/azure-tdx-preview/tdx-cli. Accessed: 2024-08-18. ().

[10] Thomas Knauth, Michael Steiner, Somnath Chakrabarti, Li Lei, Cedric Xing, and Mona Vij. 2019. Integrating remote attestation with transport layer security. (2019). https://arxiv.org/abs/1801.05863 arXiv: 1801.05863 [cs.CR].

[11] Thomas Van Laere. 2023. Azure confidential computing: verifying microsoft azure attestation jwt tokens. https://thomasvanlaere.com/posts/2023/03/azure-confidential-computing-verifying-microsoft-azure-attestation-jwt-tokens/. Accessed: 2024-09-15. (Mar. 2023).

[12] [SW] Petri Lehtinen and contributors, jansson version 2.14. VCS: https://github.com/akheron/jansson.

[13] [n. d.] *libJWT Documentation*. Accessed: 2024-08-23.

[14] Chris M. Lonvick and Tatu Ylonen. 2006. The Secure Shell (SSH) Transport Layer Protocol. RFC 4253. (Jan. 2006). DOI: 10.17487/RFC4253.

[15] [SW], OpenSSH. VCS: https://github.com/openssh/openssh-portable.

[16] [n. d.] Openssh privilege separation. https://github.com/openssh/openssh-portable/blob/master/README.privsep. Accessed: 2024-08-22. ().

[17] [n. d.] Openssh users. https://www.openssh.com/users.html. Accessed: 2024-08-18. ().

[18] [SW], OpenSSL. VCS: https://github.com/openssl/openssl.

[19] [SW] Daniel Stenberg and contributors, libcurl version 8.8.0. URL: https://curl.se/libcurl/, VCS: https://github.com/curl/curl.

[20] [n. d.] What is ssh? | secure shell (ssh) protocol. https://www.cloudflare.com/learning/access-management/what-is-ssh/. Accessed: 2024-08-18. ().

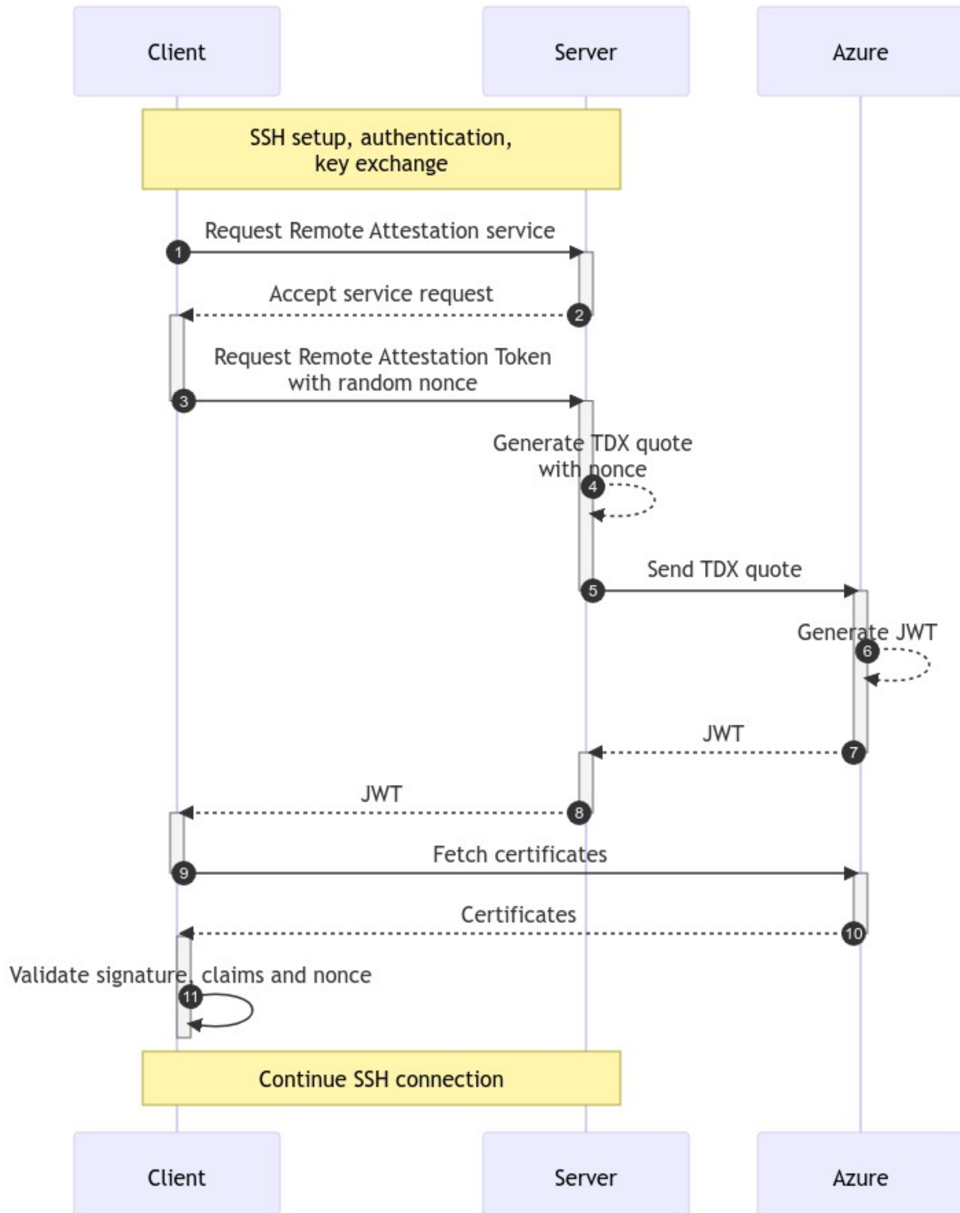

Figure 1: Remote attestation flow

```
Quote: BAACAIEAAAAAAAAAk5pyM/ecTKmUCg2zlX8GByxxGNDcfxhLFr0pGDnEuPoAAAAABAEHAAAAAAAAAAAAAAAAJeQ2JoQIQ7Glop3PO4soFtaqXMJ82cnqW
↪  hSe+Rgb8Geb3OszjUJRsnUapv3pj+EMAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAADnGAY
↪  AAAAAALTz4nSNgAcR2B+nXVtP7h7zCTMhdLJOBnyaiB

<...>
```

```
runtime_data: eyJrZXlzIjpbeyJraWQiOiJIQ0xBa1B1YiIsImtleV9vcHMiOlsic2lnbiJdLCJrdHkiOiJSU0EiLCJlIjoiQVFBQiIsIm4iOiJ5dzY5T0FBQXU
↪ zZ1E1YTJ1N0Rsa2FLWmhXbVNJLTU5RUszTFljN01nS2t1VnIyTDZzc1R5NDlTLXllUFdJY2Zrb2U5RldobVJTMVk3RVh2RUd6UXhuUkRkdUxqYjhweWJ5UGJH
↪ OU53ZzNYOTktNGNiY3pWaHhJajNaUHFESFhvckhFTURnOWZLT240RUo1X1Y5enJ3VkEtVXpOcmxXdUJZRDZxY0l4bmIzMS1MYlRsRm1SMjNIN204TklPaDlmY
↪ lE3NG1WRXlzYnMzWjQyNEIzbS11em5sX2RZanN5RTl0N1B1eGRsQTZUNVhleVdpN2hXRmdFV2lTbGMwS181b3diLUFaM0NPd1ZERTlzU21zU1NKZ2pvaHlkZ0
↪ 1RbzZpMVhNZG1SV1BYNk10alF4YjBmTWhzdHpOR2xNWmItVVZqYzhEdmVWdWFUZ3J5MFcwT0dNNE1jQ1EifSx7ImtpZCI6IkhDTEVrUHViIiwia2V5X29wcyI
↪ 6WyJlbmNyeXB0Il0sImt0eSI6IlJTQSIsImUiOiJBUUFCIiwibiI6InVQVHRlQUFCQ0JvVW9MT0JlVlVMZXJmdkVQVXFySjZ3UG80WVhMTHAwdFVNTkE2bk5V
↪ LW11TGpOUUxHLUg4blBwMFlucEVfZFE1eGc2NnhyT1RlU0NvWE9TR0xCUlMyR05zc0NSYTJyYWtWdnZMZGtQMVNpakYtYlNXV0U0VVJnbU9Zck5fS1BQWTQ5T
↪ Wt4OG9BYnhubUUydnAwSjJSRGJPTG8tbWxlaERPLUdlQm5sVThkMnpybWhONlpMaTlwM1VENkRISGhuMDlXcXdwWXZrb11leGVHd0cxVmNNZXlhY3JWSUk3Z0
↪ NsQy1LX2lzWWRud1hIVW1jalJhWF9IN1ZkSkVfb1V0ajk4anpweVhfRWIwWkVGV29tWF9NMmo5T1AxTjZmbm1IUHM1ZDA5c09DYXpSOVpUVnh6TnBEYmRoREp
↪ OR2pPTFpERE04djVmdzY4R1hRdyJ9XSwidm0tY29uZmlndXJhdGlvbiI6eyJjb25zb2xlLWVuYWJsZWQiOnRydWUsInJvb3QtY2VydC10aHVtYnByaW50Ijoi
↪ Nm5aWm5ZYUpjNEtxVVpfeXZBLW11Y0ZkWU5vdXZsUG5JVG5OTVhzSGwtMCIsInNlY3VyZS1ib290Ijp0cnVlLCJ0cG0tZW5hYmxlZCI6dHJ1ZSwidHBtLXBlc
↪ nNpc3RlZCI6dHJ1ZSwidm1VbmlxdWVJZCI6IkQyNjg1NzUxLURERTMtNDgxNC1BMkEyLTU2NUE2OUZFOTExQiJ9LCJ1c2VyLWRhdGEiOiIyRjFCQTFBQ0UyQz
↪ Y4OUQwQTNBMzM0MEI5NzQ3MkU3RkNBNURCMDNCNjYyMUIyRkI3NjFGQjA2OEUzNzdCRTFDQTNENDk0NEU1MDYwQzYwN0VEMzlDM0UzRkNDQUYwMTE0QzMyMDJ
↪ BRDUzMEZBMzE0Mzg2M0UxQjRFMTQ4NUIyMCJ9
user_data: ZmFiaWFu"
```

Listing 2: Output of `trustauthority-cli whith shortened quote`


```
{
  "attester_tcb_status": "UpToDate",
  "dbgstat": "disabled",
  "eat_profile": "https://aka.ms/maa-eat-profile-tdxvm",
  "exp": 1724352313,
  "iat": 1724323513,
  "intuse": "generic",
  "iss": "https://sharedeus2e.eus2e.attest.azure.net",
  "jti": "e7709b98c3556ae5d4310d0c99d76c5e6491e94ce2d6ff5cdec56333fddb1909",
  "nbf": 1724323513,
  "tdx_mrconfigid": "000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000",
  "tdx_mrowner": "000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000",
  "tdx_mrownerconfig": "000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000",
  "tdx_mrseam": "9790d89a10210ec6968a773cee2ca05b5aa97309f36727a968527be4606fc19e6f73acce350946c9d46a9bf7a63f8430",
  "tdx_mrsignerseam": "000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000",
  "tdx_mrtd": "b4f3e2748d800711d81fa75d5b4fee1ef309332174b24e067c9a8816873a49e3c19579a1ca6d3ccfa26ccd4bb6fc17bb",
  "tdx_report_data": "32accd5a211f291be609e1c6de2aa44e5145f3f21d32175cb6f520a3404e611b0000000000000000000000000000000000000000
  ↪  00000000000000000000000",
  "tdx_rtmr0": "000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000",
  "tdx_rtmr1": "000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000",
  "tdx_rtmr2": "000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000",
  "tdx_rtmr3": "000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000",
  "tdx_seam_attributes": "0000000000000000",
  "tdx_seamsvn": 260,
  "tdx_td_attributes": "0000000000000000",
  "tdx_td_attributes_debug": false,
  "tdx_td_attributes_key_locker": false,
  "tdx_td_attributes_perfmon": false,
  "tdx_td_attributes_protection_keys": false,
  "tdx_td_attributes_septve_disable": false,
  "tdx_tee_tcb_svn": "04010700000000000000000000000000",
  "tdx_xfam": "e718060000000000",
  "x-ms-attestation-type": "tdxvm",
  "x-ms-compliance-status": "azure-compliant-cvm",
  "x-ms-policy-hash": "9NY0VnTQ-IiBriBplVUpFbczcDaEBUwsiFYAzHu_gco",
  "x-ms-runtime": {
    "keys": [
      {
        "e": "AQAB",
        "key_ops": [
          "sign"
        ],
        "kid": "HCLAkPub",
        "kty": "RSA",
```

```
        "n": "yw69OAAAu3gQ5a2u7DlkaKZhWmSI-59EK3LYc7MgKkuVr2L6ssTy49S-yePWIcfkoe9FWhmRS1Y7EXvEGzQxnRDduLjb8pybyPbG9Nwg3X99-4cb
        ↪  czVhxIj3ZPqDHXorHEMDg9fKOn4EJ5_V9zrwVA-UzNrlWuBYD6qcIxnb31-LbTlFmR23H7m8NIOh9fbQ74mVEysbs3Z424B3m-uznl_dYjsyE9t7P
        ↪  uxdlA6T5XeyWi7hWFgEWiSlc0K_5owb-AZ3COwVDE9sSmsSSJgjohydgMQo6i1XMdmRWPX6MtjQxb0fMhstzNGlMZb-UVjc8DveVuaTgry0W0OGM4
        ↪  McCQ"
      },
      {
        "e": "AQAB",
        "key_ops": [
          "encrypt"
        ],
        "kid": "HCLEkPub",
        "kty": "RSA",
        "n": "uPTteAABCBoUoLOBeVULerfvEPUqrJ6wPo4YXLLp0tUMNA6nNU-muLjNQLG-H8nPp0YnpE_dQ5xg66xrOTeSCoXOSGLBRS2GNssCRa2rakVvvLdk
        ↪  P1SijF-bSWWE4URgmOYrN_KPPY49Mkx8oAbxnmE2vp0J2RDbOLo-mlehDO-GeBnlU8d2zrmhN6ZLi9p3UD6DHHhn09WqwpYvkoYMxeGwG1VcMeyac
        ↪  rVII7gClC-K_isYdnwXHUmcjRaX_H7VdJE_oUtj98jzpyX_Eb0ZEFWomX_M2j9OP1N6fnmHPs5d09sOCazR9ZTVxzNpDbdhDJNGjOLZDDM8v5fw68
        ↪  GXQw"
      }
    ],
    "user-data": "2F1BA1ACE2C689D0A3A3340B97472E7FCA5DB03B6621B2FB761FB068E377BE1CA3D4944E5060C607ED39C3E3FCCAF0114C3202AD530
    ↪  FA3143863E1B4E1485B20",
    "vm-configuration": {
      "console-enabled": true,
      "root-cert-thumbprint": "6nZZnYaJc4KqUZ_yvA-mucFdYNouvlPnITnNMXsHl-0",
      "secure-boot": true,
      "tpm-enabled": true,
      "tpm-persisted": true,
      "vmUniqueId": "D2685751-DDE3-4814-A2A2-565A69FE911B"
    }
  },
  "x-ms-ver": "1.0"
}
```


Listing 3: Azure JWT claims