Nautilus Vault Data Processing

This document contains information for implementation and planning of the Nautilus trusted execution environment, WASM binaries, and data schema definitions and validation.

## Nautilus TEE

### Teaclave

Data sealing and unsealing and configuration of the trusted execution environment (TEE). Our TEE is based on the [Apache Teaclave](https://github.com/apache/incubator-teaclave-sgx-sdk) SGX SDK.

We propose to use the [bare metal servers at PhoenixNAP](https://phoenixnap.com/bare-metal-cloud/3rd-gen-intel-xeon-scalable-processors) to test performance on SGXv2 hardware. Benchmarking code is already available from [Bingle’s thesis code](https://github.com/binglekruger/CCR-SGX/).

### Sealing and Unsealing

Which method do we use for sealing and unsealing? Need to investigate the current setup in the wallet, HPKE necessity and viability.

### Attestation

There is a question about using DCAP and the SGX hardware driver, which we currently use.

### SGX Performance Benchmarking

We need to test how SGXv2 scales when compared to SGXv1. Currently there is a [document](https://docs.google.com/document/d/14Xrv09a0tsFDLjbWs2xHQhGmMto_dAf2D6NymIc2CnY/edit) that contains some information about the scaling of SGX with an increase in the Enclave Page Cache (EPC)

Tests:

* To test the performance of sealing/unsealing ([CCR-SGX code](https://github.com/binglekruger/CCR-SGX/)) and test binary execution (WASM).
* Test the performance of summary statistics ([CCR-SGX code](https://github.com/binglekruger/CCR-SGX/))

D2.m2.medium from PhoenixNAP limits EPC to 2028 M (~2 GB)

* The machine has two CPUs, so the EPC should have a total of 4 GB (although there should be a slight performance drop when EPC > 2 GB).

Ice Lake Xeon-SP (and the future Xeon-SP platforms) doesn't support EPID attestation.

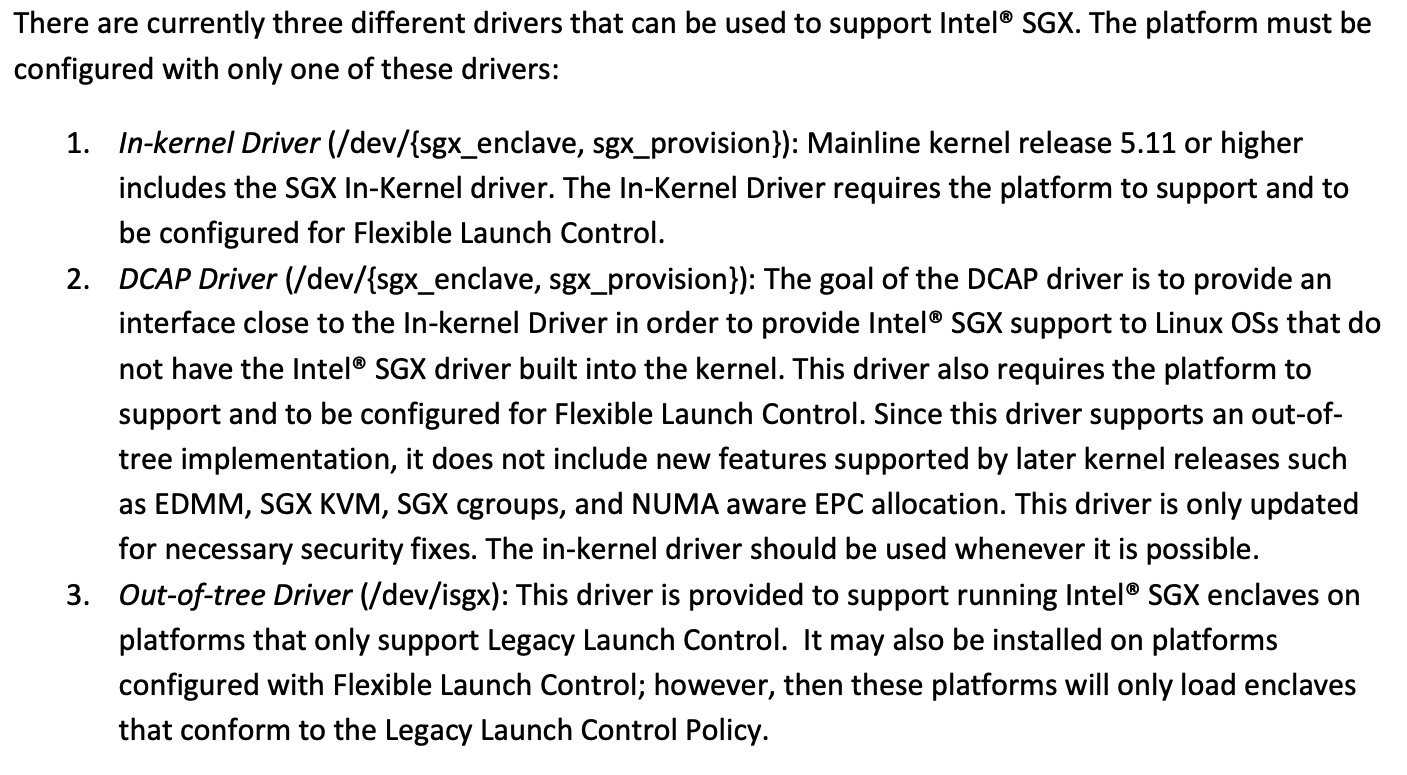
Results **(in seconds)** obtained on PhoenixNAP for a data set with 16 fields:

| # Rows | SgxSealedData | SgxUnsealedData | Seal SgxFile | Unseal SgxFile | Summary Statistics | Parse | Vector |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 000 | 0.0021860 | 0.0019214 | 0.0050529 | 0.0065334 | 0.0001532 | 1.2426615 | 0.0001383 |
| 10 000 | 0.0727096 | 0.0392152 | 0.1021837 | 0.0949076 | 0.0009443 | 12.1125652 | 0.0013181 |
| 100 000 | 0.2586848 | 0.2688657 | 0.4800884 | 0.7215854 | 0.0107194 | 121.9952074 | 0.0116502 |
| 1 000 000 | 5.6880436 | 3.9095862 | 7.5474768 | 7.6205670 | 0.1303121 | 1211.69523 | 0.0838330 |
| 10 000 000 |  |  |  |  | 1.8860420 | 12254.6033 | 75.97994 |
|  |  |  |  |  |  |  |  |

Dataset Size

| # Rows | 1 000 | 10 000 | 100 000 | 1 000 000 | 10 000 000 |
| --- | --- | --- | --- | --- | --- |
| Size (MB) | 0.283 | 2.8 | 28.5 | 285.4 | 2860 |

**We need to investigate multi-threading:** <https://github.com/apache/incubator-teaclave-sgx-sdk/tree/master/samplecode/thread>



## Vault Execution Binaries

We currently have a [basic median function](https://github.com/ntls-io/wasm-exec-sgx/tree/main/get-median-wasm) written in Rust and compiled to WASM binary. The addition of other functions is not a current priority. [We also have *standard deviation* and *mean* on separate branches]

### Append Function

The function to append data to an existing data pool is distinct from the other functions in that it is only used internally. For that reason it may not be necessary to execute it as a WASM binary, and instead run it as Rust code directly inside the enclave. This would have performance and maintenance advantages. The mean, median, linear regression functions, etc would be developed as WASM files for use as DRTs and as examples for external code providers.

Are there any arguments for maintaining the append function as a WASM binary similar to the other executables?

## Data Schema

Previous research on data schema definition was done by Pi on the [Data Schema Options](https://docs.google.com/document/d/1ZR25tgpHCkr3HXdLfMeo0D41GXOeFW6C_Sp3Y8xJ7co) document.

An option for schema definition is the [cue dev package](https://docs.dagger.io/1010/dev-cue-package/) from Daggar.

Work in Process

* Schema file validation
* Data API - store a file in cosmos, query stored files
  + Create 2 APIs (send oracle node responses to enclave & gather code and data to send to enclave)
  + Create API to query oracle nodes, validate responses and send resposes to Execution Service
* Smart Contract / Enclave API Functions
* Build sealing/unsealing
* API schema

Smart contract inputs

* Number of rows for appended dataset
  + Would be in the arguments of transaction that adds a new contributor
* Data package hash
  + In transaction for adding new contributor
  + append\_new\_contributor, update\_data\_package, smart contract approves if those two are successful
  + Could also just use append\_new\_contributor
* MVP matching column length, rows, data type in enclave while append\_datasets compute\_data\_hash

Data API:

* <https://github.com/ntls-io/nautilus-trusted-compute/pull/69>

WASM binaries:

* <https://github.com/ntls-io/wasm-exec-sgx/pull/9>
* <https://github.com/ntls-io/wasm-exec-sgx/pull/10>

Store

* Nautilus address
* Data package hash
* Number of rows
* Smart Contract ID ( Application ID )
* Smart Contract Address
* Contributor Token ID ( Asset ID )
* DRT ID ( Asset ID )
* DRT Name
* DRT Exchange Rate

Collections

* Append collection
* Smart contract
* Data package
* Wallet [maybe just use the NW metadata database]
* DRT collection
* Data schema types
  + schema\_desc\_short (128 bytes) to smart contract maybe