USENIX Security '26 Artifact Appendix: Trustworthy and Confidential SBOM Exchange

Eman Abu Ishgair[†] Chinenye Okafor[†] Marcela S. Melara[‡] Santiago Torres-Arias[†]

[†]Purdue University, {eabuishg, okafor1, santiagotorres}@purdue.edu

[‡]Intel Corporation, marcela.melara@intel.com

A Artifact Appendix

A.1 Abstract

This artifact appendix provides a reference implementation of Petra, a system for confidentiality- and integrity-preserving Software Bill of Materials (SBOM) exchange. Petra enables selective disclosure of SBOM fields using Ciphertext-Policy Attribute-Based Encryption (CP-ABE) while preserving verifiable integrity through Merkle-authenticated SBOM trees. The artifact includes: (i) Petra's implementation, (ii) CP-ABE bindings, (iii) SBOM preprocessing and evaluation scripts to reproduce the performance and storage overhead results reported in the paper. Evaluators can reproduce all major experimental claims either by installing Petra directly on their machine or using the containerized setup.

A.2 Description & Requirements

The artifact evaluates Petra's ability to securely encrypt, decrypt, and verify redacted SBOMs at scale. Experiments measure: tree construction cost, Merkle hashing cost, encryption and decryption overhead, storage overhead across plaintext, encrypted, and decrypted SBOM trees.

The evaluation uses Chainguard's bom-shelter SBOM dataset containing SPDX and CycloneDX SBOMs of varying sizes and complexity.

A.2.1 Security, privacy, and ethical concerns

The artifact does not require disabling security mechanisms, elevated privileges (except installing dependencies), or access to sensitive personal data. The evaluation processes publicly available SBOM datasets and locally generated cryptographic keys. No network services are contacted except for dependency installation and dataset retrieval. The included Key Management Service (KMS) runs locally for testing purposes only.

A.2.2 How to access

The artifact is available as a public, git repository:

```
git clone https://github.com/TSELab/SBOMCtl.git
cd SBOMCtl
git checkout \
d49dbf6c82202dd40eb5cb68b8eb375741495fcc
```

We also made a git-archived version available via Zenodo: https://doi.org/10.5281/zenodo.17906622

The repository contains:

1. Petra source code
2. Evaluation scripts
3. Configuration files
4. Dockerfile for containerized evaluation

A Docker image containing all required dependencies for Petra and its evaluation pipeline, can also be rebuilt locally using the Dockerfile included in the source archive.

A.2.3 Hardware dependencies

None. The evaluation was carried out on a commodity x86-64 machine equipped with a 2.10 GHz Intel Xeon E5620 CPU and 256 GB of RAM. No specialized hardware (e.g., GPUs, FPGAs, or hardware accelerators) is required to evaluate this artifact.

A.2.4 Software dependencies

Local machine setup Linux or macOS, Python >= 3.9, Rust toolchain (rustc, cargo), OpenSSL, Docker.

Python dependencies Installed via pip install -r requirements.txt

Rust dependencies Built via maturin for CP-ABE bindings

A.2.5 Benchmarks

The evaluation uses the Chainguard's bom-shelter SBOM corpus, which contains SPDX and CycloneDX SBOMs between 1 and 5000 packages per SBOM. The dataset access methods:

1. Direct clone from GitHub:

```
# From artifact root directory(SBOMCtl)
cd sbom-data
git clone \
https://github.com/chainguard-dev/bom-shelter
cd bom-shelter
git checkout \
89ed4943e027c9bf148e6a80d7b2b78ae8d7771a
```

2. Zenodo: https://doi.org/10.5281/zenodo.17859760 Download into `SBOMCtl/sbom_data` directory

A.3 Set-up

A.3.1 Installation

Option A: Docker (recommended) For a fully containerized setup (that has all the petra setup and dataset), you can run Petra using the provided Docker file. This option packages Petra, cpabe bindings, and the bom-shelter dataset inside the container, so you can start directly from preprocessing and evaluation.

The following commands build and start the evaluation container:

```
# From artifact root directory(SBOMCtl)
docker build -f dockerfiles/petra_evaluation \
-t petra-eval .

docker run -d --name petra-eval petra-eval \
tail -f /dev/null
```

To enter the container at anytime:

```
docker exec -it petra-eval bash
```

Option B: Native (non-containerized) Setup You can set up a virtual environment in two ways:

```
# From artifact root directory(SBOMCtl)
cd privateSBOMExchange
make init
```

or

```
# From artifact root directory(SBOMCtl)
cd privateSBOMExchange
virtualenv venv
source venv/bin/activate
pip install -r requirements.txt
```

Build CP-ABE bindings cpabe is the python bindings for the rust crate rabe, this is needed to support CP-ABE functionality. Before you run any other code, make sure that these bindings are built by issuing:

```
# From artifact root directory(SBOMCtl)
cd privateSBOMExchange
sudo apt install rustc cargo
cd src/cpabe
maturin develop
```

This will build the rust code, as well as install an editable version of the python bindings into the current virtual environment.

Install Petra Petra is the main system. If cpabe is installed, you should be able to install the petra package:

```
# From artifact root directory(SBOMCtl)
cd privateSBOMExchange
pip install -e .
```

SBOM Signing Test Key Generation generates the SBOM signing and verification keys for Petra. Assuming you have openssl installed, generate DER format ECDSA keys:

```
# From artifact root directory(SBOMCtl)
cd privateSBOMExchange
cd tests
openssl ecparam -name prime256v1 -outform der\
-genkey -out privkey.der -noout
openssl ec -inform der -in privkey.der -pubout\
-outform der -out pubkey.der
```

Start the KMS Service The KMS service handles the key management, to start it run:

```
# From artifact root directory(SBOMCtl)
cd privateSBOMExchange
python src/petra/internals/kms.py
```

A.3.2 Basic Test

After the setup, you should be able to run the Petra CLI by running the test files under /tests. To run a functional encryption and decryption test including functional KMS:

```
# From artifact root directory(SBOMCtl)
cd privateSBOMExchange
python tests/test_models.py
```

Expected outcome:

```
Printing raw SBOM tree
done encrypting
Printing hashed and encrypted SBOM tree
Merkle Root Hash for SBOM:
```

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): Petra enables selective disclosure of SBOM fields while preserving integrity across redistribution. Demonstrated by encryption, decryption, and membership-proofs.

(C2): Petra introduces <13% storage overhead for encrypted SBOMs on average. Demonstrated by storage measurements in Section 6.2.1 and reproduced via evaluation scripts.

(C3): CP-ABE encryption and decryption overhead scales with SBOM size and node count but remains practical. Demonstrated by timing measurements in Figures 5 and 6.

A.4.2 Experiments

The following steps run Petra against two policies: security weaknesses, and intellectual property policy, which are two separate experiments that we run at once.

Preprocessing [20–30 compute-minutes]

Execution: This step parses SBOMs, tries to build SBOM trees and filters SBOMs with erroneous formats.

```
# From artifact root directory(SBOMCtl)
cd privateSBOMExchange
python evaluation/preprocess_sboms.py
```

Results: Valid SBOMs copied to `input_sboms` directory

Performance Evaluation [8 - 10 compute-hours]:

Execution: This step executes the following script to build the SBOM tree, encrypt and decrypt SBOM trees, verify integrity properties and record timing results (C1).

```
# From artifact root directory(SBOMCtl)
cd privateSBOMExchange
python evaluation/test_evaluations_AES.py
```

Results: For each tested policy, Petra saves the performance results in a JSON file named after the policy under `privateSBOMExchange/output` directory, the results contains the following for each SBOM file: file size, build tree time, hash time, encrypt time, decrypt time, sbom tree storage, encrypted tree storage, decrypted tree storage, tree nodes count, policy

Plot Generation [5 minutes]:

Execution: The generated performance files in Performance Evaluation are automatically consumed by the plotting scripts in evaluation/, which aggregate metrics across policies and produce comparative performance visualizations.

Generate performance plots with:

```
# From artifact root directory(SBOMCtl)
cd privateSBOMExchange
python evaluation/plot_perf_size_AES.py
```

Results: For each tested policy, the corresponding plots are automatically saved in a subdirectory named after the policy inside the `privateSBOMExchange/output` directory. This step should reproduce encrypt time vs tree nodes count plots that correspond to figure 5. It also reproduces decrypt time vs tree nodes count plots that correspond to figure, these plots cover C3.The script finally calculates the following statistics:

1. Average Percentage Increase from SBOM Tree Size to Encrypted Tree Size.
2. Average Percentage Increase from Encrypted Tree Size to Decrypted Tree Size.
3. Ratio of encrypted tree storage to SBOM tree storage.
4. Mean encryption – decryption time difference.
5. Mean decryption time percentage.

The computed statistics average percentage increase from SBOM tree size to encrypted tree size, and average percentage increase from encrypted tree size to decrypted tree size corresponds to encryption and decryption storage overhead results in Section 6.2.1 of the paper (C2).

The computed statistics mean decryption time percentage corresponds to decryption time percentage in the Abstract and the Introduction.

By comparing your generated plots and statistics with those in the paper, you can verify that Petra exhibits the expected performance behavior on your dataset or configuration.

A.5 Version

Based on the LaTeX template for Artifact Evaluation V20231005. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/usenixsec2026/.