USENIX Security '25 Artifact Appendix: S/MINE: Collecting and Analyzing S/MIME Certificates at Scale

Gurur Öndarö[1], Jonas Kaspereit[1], Samson Umezulike[2], Christoph Saatjohann[1], Fabian Ising[2], and Sebastian Schinzel[1,2]

[1]*Münster University of Applied Sciences*
[2]*Fraunhofer SIT and National Research Center for Applied Cybersecurity ATHENE*

A Artifact Appendix

A.1 Abstract

We present the first comprehensive analysis of real-world S/MIME certificates used for digitally signing and encrypting emails. Our study focuses on the structure of the S/MIME PKI, the cryptographic properties of issued certificates, and their compliance with emerging standards such as the CA/Browser Forum's S/MIME Baseline Requirements. To conduct this study, we collected over 41 million unique X.509 certificates from public address books, i.e., LDAP servers, of which 38 million are suitable for use with S/MIME in email clients. Despite the inherent complexity of S/MIME trust chains, we developed and applied a custom verification tool that successfully reconstructed and validated certificate chains.

The artifact includes several components to facilitate reproducibility and support future research. It provides our custom-built chain verification tool, the scripts used for certificate collection and analysis, as well as the data required to replicate the core results presented in the paper. Due to the presence of personally identifiable information in the certificates (e.g., names and email addresses), we will provide the raw certificate dataset only for the second phase of the artifact evaluation but will not publish it. However, we do publish the list of IP addresses of the LDAP servers from which the certificates were collected, along with the crawling script, enabling others to replicate the data collection independently.

The goal of this artifact evaluation is to validate the replicability of our findings, evaluate the usability of our tooling, and confirm the completeness and transparency of our methodology. By releasing our tooling and methodology, we aim to enable the community to extend our work, reassess the ecosystem over time, and contribute to the ongoing improvement of S/MIME infrastructure security.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

Our scripts and tools will be executed on the virtual machine that we provide and host for the artifact evaluation. Our scripts do not contain destructive steps, and no security mechanisms have to be disabled. Please note that our public artifact contains only the scripts and tools, not the certificate dataset. Due to privacy concerns, we will not release the dataset publicly. However, it is made available to the artifact reviewers for their evaluation of the "Functional" and "Reproduced" badges.

A.2.2 How to access

Our artifact is available on Zenodo (`https://doi.org/10.5281/zenodo.15533203`) and GitHub (`https://github.com/FHMS-ITS/SMINE/`). It includes the scripts and tools we developed and used during our research. However, the certificate dataset will not be released due to privacy concerns, which we discuss in more detail in the paper. Consequently, some scripts cannot be executed without a connection to a MongoDB containing certificates.

A.2.3 Hardware dependencies

None.

A.2.4 Software dependencies

No specific requirements, any modern OS capable of running Docker containers and Python scripts will suffice. The Python requirements are bundled in a `requirements.txt` file as is customary.

For artifact evaluation, we provided reviewers with access to a local LDAP server containing certificates, which was used to assess the functionality of our LDAP crawler. Researchers can set up a similar local LDAP server serving certificates on their own machine to test our crawler.

A.2.5 Benchmarks

None.

A.3 Set-up

A.3.1 Installation

To use the tools and scripts in the repository, you will need to have Python >=3.10 installed along with the required dependencies. We recommend using a virtual environment to manage dependencies.

```
apt install libsasl2-dev python-dev-is-python3
    libldap2-dev libssl-dev
python -m venv venv
source venv/bin/activate
pip install -r requirements.txt
export PYTHONPATH=$(pwd)
```

To test the crawler, set up a basic LDAP server, such as OpenLDAP (available via Docker at https://hub.docker.com/r/bitnami/openldap). Once the server is running, upload certificates to it by adding them to the `userCertificate;binary` attribute of the LDAP entries.

A.3.2 Basic Test

The script `~/SMINE/basic_checks.py` is provided to assert that all components are functional and correctly configured. It can be executed with Python as usual and will run a series of checks, printing instructions on how to proceed if any fail, and `All checks passed!` otherwise.

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): We implemented a crawler that crawls available certificates from an LDAP server using wildcard searches. (E1)

(C2): **Section 4** We collected 41,692,142 unique X.509 certificates from 2,879 LDAP servers, of which 38,374,783 fulfill the requirements for use with S/MIME in email clients. We described the skewed distribution of S/MIME certificates in Section 4.3, the Tables 2 and 3 show the distribution among countries and continents.
Section 5 We described the distribution of different configurations of Key Usage, Extended Key Usage and existing email address in Table 4. We found that 22,385,707 S/MIME certificates contained at least one email address, in total 22,672,795 email addresses and 806,746 unique domains. Of these, 15,865,134 contain at least one email address issued for a domain in the Tranco 1 million list[1], covering 30,872 domains. Using a list of free mail domains[2], we found that 9,123,368 certificates contain at least one free mail address and name the most common free-mail and non-free-mail domains. In Figure 3, we show that most certificates have a validity period of multiple years and three years is the most common value. In Figure 4, we show the development of different public key algorithms over time. We find that CAs have begun implementing the S/MIME Baseline Requirements but most trusted certificates are still non-compliant. We show that a large subset of certificates is issued by a small number of CAs in Figure 5 and Table 5.
Section 6 We construct full certificate chains for the majority of certificates and connect 20% to a widely trusted root CA. All these values can be reproduced with experiment (E2).

(C3): In Section 7 we analyzed the cryptographic keys contained and found vulnerabilities in 71,214 certificates. Table 7 shows an overview. The analysis can be reproduced with experiment (E3).

(C4): We implemented a tool that reconstructs and verifies S/MIME certificate chains for a given leaf certificate.

A.4.2 Experiments

(E0.1): (Optional) Execute the preprocessing scripts. These scripts were used to enrich the certificates in the database with additional information. This includes, for example, parsing X.509 certificates, classifying and flagging S/MIME certificates, reconstructing and validating certificate chains, and labeling certificates as CA certificates. The README file in the `processing` directory contains more information about the individual tasks.
Preparation: Activate the environment as described in appendices A.2.2 and A.3.
Execution: Execute the `run.py` script in the `processing` directory to run all tasks.

```
cat assets/example_cert.json | python run.
    py certs > results.json
cat assets/example_host.json | python run.
    py hosts
```

(E1): [5 human-minutes + 30 compute-minutes]: Apply the crawler to the LDAP server (possibly in Docker) to crawl for certificates.
Preparation: The crawler can be tested against the LDAP server set up in appendix A.3.1. For the artifact evaluation, we provided an OpenLDAP Docker instance on the virtual machine, which served a certificate sample from our dataset and listened on port 389. Activate the environment as described in appendices A.2.2 and A.3.

[1] https://tranco-list.eu/list/VQ2QN

[2] https://gist.github.com/okutbay/5b4974b70673dfdcc21c517632c1f984

Execution: Execute the crawler script in the `ldap_crawler` directory.

```
python crawler.py 127.0.0.1,636,389 crawls
```

First, the crawler attempts to connect to port 636 on localhost. If port 636 is unavailable, it falls back to port 389. The crawling results (certificates) and logs are then written to the `crawls` directory.
Results: The `crawls` directory will contain a results file located in the results subdirectory, which includes the certificates crawled for the specified IP address. This experiment demonstrates the functionality of our crawler.

(E2): [1 human-hour, including validating the results. Little to no compute time when using the cache.]: Execute the scripts in the `analysis` directory.
Preparation: Activate the environment as described in appendices A.2.2 and A.3.
Execution: Execute the scripts:

```
python analysis/general_stats.py
python analysis/certs_per_host.py
python analysis/crawled_hosts_per_month.py
python analysis/certs_distribution.py
python analysis/smime_client_support.py
python analysis/smime_email_addresses.py
python analysis/smime_validity.py
python analysis/key_usage.py
python analysis/smime_key_algorithms.py
python analysis/smime_cas.py
python analysis/smime_br.py
```

Each script first attempts to load the results from a cache file located in the assets directory, which is generated after each run. If this file does not exist, the script will establish a connection to the database to execute the query. You can use the `refresh` argument to force the query to be executed directly against the database like:

```
python analysis/smime_br.py refresh
```

Results: The scripts will output the numbers, tables, and figures used in the paper.

(E3): [1 human-hour, including validating the results. Little to no compute time when using the cache.]: Execute the scripts in the `analysis/key_analysis` directory.
Preparation: Activate the environment as described in appendices A.2.2 and A.3.
Execution: Execute the scripts:

```
python analysis/key_analysis/
    weak_key_certs_count.py
python analysis/key_analysis/fastgcd_status.
    py
python analysis/key_analysis/badkeys.py
python analysis/key_analysis/ec_keys.py
python analysis/key_analysis/
    factordb_status.py
python analysis/key_analysis/
    pwnedkeys_blocklist_merge_status.py
```

Results: The scripts would output the numbers, tables, and figures used in the paper, provided they are run with the appropriate dataset.

(E4): [5 human minutes]: Reconstruct certificate chains with the chain verification tool.
Preparation: Activate the environment as described in appendices A.2.2 and A.3.
Execution: Execute the Python tests in the `chain_verification` directory:

```
python tests/run_all_tests.py
```

To reconstruct a certificate chain using the chain verification tool, run experiment E0.1. This experiment includes the `cert_chains_task.py` script located in the tasks directory, which applies the chain verification tool to the given certificate. After execution, refer to the `results.json` file for the chain verification output.
Results: The chain verification tool provides a chain for the given certificate

A.5 Notes on Reusability

The crawler, in combination with the list of LDAP servers, can be used to recreate the certificate dataset. Please do so responsibly and follow best practices for scanning. Analysis scripts can be used and extended to analyze the resulting dataset.

A.6 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/usenixsec2025/.