USENIX Security '24 Artifact Appendix: Shaken, not Stirred - Automated Discovery of Subtle Attacks on Protocols using Mix-Nets

Jannik Dreier
Université de Lorraine, CNRS, Inria, LORIA

Pascal Lafourcade
Université de Clermont Auvergne, LIMOS

Dhekra Mahmoud
Université de Clermont Auvergne, LIMOS

A Artifact Appendix

As stated in the research paper, using the protocol verifier ProVerif, we (re)discover automatically attacks on four protocols. The artifacts correspond to several ProVerif files and a script installing ProVerif and the corresponding dependencies.

Each ProVerif file corresponds to a security property and a protocol we have considered in the paper. In our case, for each file, ProVerif gives either a proof that the corresponding security property is verified, or an attack.

A.1 Abstract

This artifact allows reproducing the results described in the research paper (more specifically, the outcome of the analysis depicted in Tables 4 and 5 of the paper). It contains the ProVerif version used, a script for installing ProVerif and the corresponding dependencies, ProVerif files to be run and a detailed Readme on how to run the files, the expected outcomes, and the execution time.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

During the execution of our artifact, there is no risk to the evaluators' machines' security, data privacy, or other ethical concerns.

A.2.2 How to access

All of our files are publicly accessible and can be accessed online through the Gitlab repository[1] at commit `a877ffca4339b09ae0b674074aa3186e6508230c`. This can be done using a web browser[2] or by cloning the repository and selecting the commit using the following commands:

```
git clone https://gitlab.limos.fr/dhmahmoud/usenix24-632
cd usenix24-632
git checkout a877ffca4339b09ae0b674074aa3186e6508230c
```

[1] https://gitlab.limos.fr/dhmahmoud/usenix24-632/

[2] https://gitlab.limos.fr/dhmahmoud/usenix24-632/-/tree/a877ffca4339b09ae0b674074aa3186e6508230c

A.2.3 Hardware dependencies

The evaluation of the artifact can be conducted on a standard laptop with at least 8GB of RAM.

A.2.4 Software dependencies

The main software tool needed for the artifacts is the automatic protocol verifier ProVerif [3]. The installation script from the Gitlab repository requires Ubuntu Linux, and has been successfully tested on a fresh Ubuntu Desktop 24.04 LTS, installed from ISO image [4].

A.2.5 Benchmarks

The only data required for the artifacts are the ProVerif files contained in the Gitlab repository.

A.3 Set-up

A.3.1 Installation

If ProVerif (version 2.05) is already installed, then one simply needs to clone the repository and start running the files (see Section A.4). Otherwise, after cloning the repository as described in Section A.2.2, one needs to execute the following commands:

```
cd usenix24-632
sh run_install-dep-pv.sh
```

Some commands may need admin rights and ask the for the user's password.

Moreover, when the executing of the script, the user may be prompted to choose options. Simply select the default option by pressing ENTER.

[3] https://bblanche.gitlabpages.inria.fr/proverif/manual.pdf

[4] https://ubuntu.com/download/desktop

A.3.2 Basic Test

To check whether the installation of ProVerif was successful or not (i.e., whether the installation produced a correct executable), one need to execute:

```
./proverif2.05/proverif --help
```

This command should display the list of options supported by ProVerif and begin as follows.

```
Proverif 2.05. Cryptographic protocol verifier, by Bruno Blanchet, Vincent Cheval, and Marc Sylvestre
```

A.4 Evaluation workflow

A.4.1 Major Claims

The major claims of our analysis are the following:

(C1): For all tested protocols, when the MixNets do no request ZKPs from senders, or when the MixNets use weak ZKPs, all the tested security properties are **not** verified.

(C2): For all tested protocols, when the MixNets use strong ZKPs, all the tested security properties are verified.

Claim **C1** is verified in experiment **E1** and Claim **C2** is verified in experiment **E2**. All our claims correspond to Table 4 and Table 5 of our paper.

A.4.2 Experiments

(E1): [Without-ZKP, Weak-ZKP]

Preparation: In this experiment, the ProVerif files corresponding to the the Claim **C1** are the ones ending with Without-ZKP and Weak-ZKP.

Execution: After choosing a ProVerif file ending with Without-ZKP or Weak-ZKP, execute the following command:

```
./proverif2.05/proverif <file>_Weak-ZKP.pv
```

for Weak-ZKP, and:

```
./proverif2.05/proverif <file>_Without-ZKP.pv
```

for Without-ZKP.

To run all files, run:

```
./run-all-E1.sh
```

Results: The end of the output (when executing individually) or the excerpts (when using the script) should be the following:

```
------------------------------------------
Verification summary:
Observational equivalence cannot be proved.
------------------------------------------
```

The results should also correspond to the ones given in Tables 4 and 5, i.e., for any given protocol and property, in case of a weak or absent ZKP, the property does not hold.

According to the timings given in the paper, the expected time for running the (E1) script is around one hour.

(E2): [Strong-ZKP]

Preparation: In this experiment, the ProVerif files corresponding to the the Claim **C2** are the ones ending with Strong-ZKP.

Execution: After choosing a ProVerif file ending with Strong-ZKP, execute the following command:

```
./proverif2.05/proverif <file>_Strong-ZKP.pv
```

To run all files, run:

```
./run-all-E2.sh
```

Results: The end of the output (when executing individually) or the excerpts (when using the script) should be the following:

```
------------------------------------------
Verification summary:
Observational equivalence is true.
------------------------------------------
```

The result should also correspond to the one given in Tables 4 and 5, i.e., for any given protocol and property, in case of a strong ZKP, the property does hold.

According to the timings given in the paper, the expected time for running the (E2) script is around one minute.

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/usenixsec2024/.