

## APPENDIX A ARTIFACT APPENDIX

This artifact allows to reproduce the symbolic analysis described in research paper “A Unified Symbolic Analysis of WireGuard”. It contains:

- 1) Used versions of TAMARIN and PROVERIF.
- 2) The reference models of WireGuard.
- 3) The scripts to generate all evaluation files in PROVERIF, to evaluate them, to compute DNF for all security properties, all evaluation files in TAMARIN and the scripts to evaluate them.

### A. Access, Requirements, Installation, Checks & Benchmarks

All our files are publicly available and can be accessed online either through Gitlab repository (commit hash: cefa5c14103badcf895495dff048919065cfb6a4), Docker image (tag: 913b61a1087a7be9de7db2dadf980080ce9a06a934a3f9734440dda2b8bfc34a) or Zenodo (<https://doi.org/10.5281/zenodo.10126619>). Docker image contains all software pre-installed and requires a running Docker Engine<sup>1</sup>. Gitlab repository contains an installation script that has been successfully tested on a fresh Ubuntu Server 22.04.3 LTS, installed from ISO image<sup>2</sup>.

#### 1) Access through Gitlab and software installation:

```
$ git clone https://gitlab.limos.fr/palafour/ndss2024-AE364
$ cd ndss2024-AE364
$ sh run_install-dep-tam-pv.sh
```

#### 2) Access through Docker (no installation required):

```
$ docker pull wganalysis/artifacts
$ docker run -it wganalysis/artifacts bash
```

#### 3) Hardware requirements to run the artifacts:

- **Configuration (C1)** A standard laptop with 8 cores of CPU 1.8 GHz and 16 Go of RAM. This architecture can be used to run experiment **E1** but shall not be used to run experiments **E2**, **E3**, **E4**.
- **Configuration (C2)** A dedicated server, with at least 256 cores of CPU 1.5 GHz and 512 Go of RAM, on which experiments **E2**, **E3** and **E4** shall be run.

4) *Basic checks:* to check whether the Docker started successfully, or whether installation through Gitlab worked, execute:

```
$ tamarin-prover test
```

In the end you should see the following:

```
*** TEST SUMMARY ***
All tests successful.
The tamarin-prover should work as intended.
```

```
:-) happy proving (-)
```

If result is `tamarin-prover: not found`, open a new terminal and repeat previous command. Then execute:

```
$ eval $(opam env --safe); proverif -help
```

In the beginning you should see the following:

```
Proverif 2.04. Cryptographic protocol verifier
```

<sup>1</sup><https://docs.docker.com/engine/install/>

<sup>2</sup><https://ubuntu.com/download/server>

### 5) Benchmarks:

- Experiment **E1** can be run on a standard laptop of configuration **C1**, results are obtained in 20 minutes.
- Experiments **E2**, **E3** and **E4** shall be run on a server of configuration **C2**. For this architecture, results are available in 9 hours for **E2**, 12 hours for **E3** and 2 hours for **E4**.

### B. Major Claims

We assess the following security properties:

- Agreement properties: agreement of `RecHello` message (from Responder to Initiator), agreement of first `TransData` message (from Initiator to Responder), agreement of next `TransData` messages (from Initiator to Responder and from Responder to Initiator), for WireGuard with or without cookies and for two fixed versions of WireGuard.
- Secrecy properties: secrecy and PFS of session key before derivation (named  $k_6$  in protocol description), from Initiator's and Responder's view, secrecy and PFS of derivated keys (named  $C^i$  and  $C^r$ ), from Initiator's and Responder's view, for WireGuard with or without cookies and for two fixed versions of WireGuard.
- Anonymity, for WireGuard with or without cookies and for two fixed versions of WireGuard.

Agreement and secrecy for WireGuard without cookie are verified in experiment **E2**, fixes for anonymity are verified in experiments **E3** and **E4**. Experiment **E1** concerns PFS of session key before derivation from Initiator's view.

### C. Evaluation

1) **Experiment (E1):** [PFS of session key before derivation from Initiator's view for WireGuard without cookie] [5 human-minutes + 15 compute-minutes on configuration **C1**] this experiment corresponds to Section 6.A of our research paper. Execute:

```
$ cd process_complete_minimal_tests
$ sh run_all.sh
```

This will launch, sequentially:

- Generation of PROVERIF files from reference `.spthy` files. For this evaluated property, there are 64 files.
- Evaluation of all PROVERIF files.
- Computation of DNF from all evaluated PROVERIF files.
- Evaluation of dedicated TAMARIN file.

TAMARIN file for this property is available in folder `__tamarin__`. It contains one *lemma* named `Secrecy_IK6_PFS`, which is deduced from previous DNF. Output should be (numbers correspond to computation duration and may be different):

```
Generate ProVerif queries
Generate ProVerif files
0:00.79
[WARNING] Running as root is not recommended
Evaluate ProVerif queries for isk6 pfs
2:16.28
Generate CNF and DNF files
0:00.46
```

```
Evaluate Tamarin Lemma
[Saturating Sources] Step 1/5
[Saturating Sources] Step 2/5
[Saturating Sources] Step 3/5
[Saturating Sources] Step 4/5
[Saturating Sources] Step 5/5
[Saturating Sources] Saturation aborted, more than 5
iterations. (Limit can be change with -s=)
2:28.59
```

Once computation is finished, directory `process_complete_minimal_tests` contains new folders, named `secrecy_isk6_pfs` and `results`. Folder `secrecy_isk6_pfs` contains all generated PROVERIF files (\*.pv), all corresponding evaluation files (\*.pv.log) and all sub-folders used to compute DNF, as described in research paper, Section 6. Folder `results` contains 2 files:

- `wireguard_secrecy_isk6_pfs.cnfdnf`
- `wireguard_secrecy_isk6_pfs_all_trusted.tamarin`

Content of `.cnfdnf` file corresponds to the content of Table 2 of research paper, for a part of **DNF3\***, which is  $(R_s^* \wedge R_u^* \wedge R_x) \vee (R_s^* \wedge R_v^* \wedge R_y) \vee (R_c^* \wedge R_s^* \wedge R_x \wedge R_y)$ . Content of `.tamarin` file corresponds to the log files of TAMARIN resolution for the property. Execute:

```
$ cd results
$ grep "verified\\|falsified" *.tamarin
```

Output should be:

```
Secrecy_IK6_PFS (all-traces): verified (268 steps)
```

2) **Experiment (E2):** [Agreement and Secrecy for WireGuard without cookie] [5 human-minutes + 9 compute-hours on configuration **C2**] this experiment corresponds to Section 6.A of our research paper. Execute:

```
$ cd process_complete_without_cookie
$ sh run_all.sh
```

This will launch, sequentially:

- Generation of PROVERIF files from reference `.spthy` files (up to 4860 files per property).
- Evaluation of all PROVERIF files.
- Computation of DNF from all evaluated PROVERIF files.
- Evaluation of dedicated TAMARIN file.

Output message is as in experiment **E1**. Our experiment and obtained durations, on a server of configuration **C2** are detailed on our Gitlab repository (A-A1). After computation, directory `process_complete_without_cookie` contains new folders, `secrecy_*`, `agreement_*`. These contain all generated PROVERIF files (\*.pv), all corresponding evaluation files (\*.pv.log) and all sub-folders used to compute DNF. New folder `results` contains two types of files: `*.cnfdnf` and `*.tamarin`. Link between content of Table 2 of research paper and `*.cnfdnf` files is described in Table I. Each `*.spthy` file in folder `__tamarin__` is dedicated to a security property and evaluates one *lemma* which is deduced from previously computed **DNF1\***, **DNF2\***, **DNF3\***, **DNF4\***. Each `*.tamarin` file in folder `results` is the log file of their evaluation. Execute:

```
$ cd results
$ grep "verified\\|falsified" *.tamarin
```

DNF	Computed files
<b>DNF1, DNF1*</b>	wireguard_agreement_rechello.cnfdnf wireguard_agreement_transport_rtoi.cnfdnf
<b>DNF2, DNF2*</b>	wireguard_agreement_confirm.cnfdnf wireguard_agreement_transport_itor.cnfdnf
<b>DNF3, DNF3*</b>	wireguard_secrecy_isk6.cnfdnf wireguard_secrecy_isk6_pfs.cnfdnf wireguard_secrecy_isk_itor.cnfdnf wireguard_secrecy_isk_itor_pfs.cnfdnf wireguard_secrecy_isk_rtoi.cnfdnf wireguard_secrecy_isk_rtoi_pfs.cnfdnf
<b>DNF4, DNF4*</b>	wireguard_secrecy_rsk6.cnfdnf wireguard_secrecy_rsk6_pfs.cnfdnf wireguard_secrecy_rsk_itor.cnfdnf wireguard_secrecy_rsk_itor_pfs.cnfdnf wireguard_secrecy_rsk_rtoi.cnfdnf wireguard_secrecy_rsk_rtoi_pfs.cnfdnf

TABLE I  
LINK BETWEEN COMPUTED FILES AND TABLE 2 FROM RESEARCH PAPER

Each line should contain (all-traces):verified except for `inithello_untrusted_pki` which should contain (all-traces):falsified.

3) **Experiment (E3):** [Anonymity for fixed version of WireGuard without cookie, based on  $g^{uv}$ ] [5 human-minutes + 12 compute-hours on configuration **C2**] this experiment corresponds to Section 6.B of our research paper. Execute:

```
$ cd process_complete_with_fix_guv
$ sh run_evaluate-anonymity.sh
```

This generates 8 PROVERIF files, named `Anonymity_with_fix_guv_*`, with  $*$  = `_Rs`, `_Rc`, `_Ru`, `_Rv`, `_Rx`, `_Ry`, `_RsRy`, `_WITHOUT_R`. Execute:

```
$ cd __anonymity__
$ grep "RESULT" *.log
```

Output should be:

- For files `_Ry`, `_Rs`, `_RsRy` and `_WITHOUT_R`, RESULT Observational equivalence is true.
- For all other files, RESULT Observational equivalence cannot be proved.

4) **Experiment (E4):** [Anonymity for fixed version of WireGuard without cookie, based on `psk`] [5 human-minutes + 2 compute-hours on configuration **C2**] this experiment corresponds to Section 6.B of our research paper. Execute:

```
$ cd process_complete_with_fix_psk
$ sh run_evaluate-anonymity.sh
```

This generates 9 PROVERIF files, named `Anonymity_with_fix_guv_*`, with  $*$  = `_Rs`, `_Rc`, `_Ru`, `_Rv`, `_Rx`, `_Ry`, `_RcRy`, `_RuRy`, `_WITHOUT_R`. Execute:

```
$ cd __anonymity__
$ grep "RESULT" *.log
```

Output should be:

- For files `_Rc`, `_Ru`, `_Ry`, `_RcRy`, `_RuRy` and `_WITHOUT_R`, RESULT Observational equivalence is true.
- For all other files, RESULT Observational equivalence cannot be proved.