PURE: Payments with UWB RElay-protection

Daniele Coppola, Giovanni Camurati, Claudio Anliker, Xenia Hofmeier, Patrick Schaller, David Basin, and Srdjan Capkun, *ETH Zurich*

https://www.usenix.org/conference/usenixsecurity24/presentation/coppola

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 33rd USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 33rd USENIX Security Symposium.

August 14–16, 2024 • Philadelphia, PA, USA

978-1-939133-44-1

USENIX Security ’24 Artifact Appendix: PURE: Payments with UWB RElay-protection

Daniele Coppola, ETH Zurich
Giovanni Camurati, ETH Zurich
Claudio Anliker, ETH Zurich
Xenia Hofmeier, ETH Zurich
Patrick Schaller, ETH Zurich
David Basin, ETH Zurich
Srdjan Capkun, ETH Zurich

A Artifact Appendix

A.1 Abstract

Our artifacts consist of the source code for the proof-of-concept (PoC) implementation, the datasets collected to evaluate our implementation, and the formal model of PURE in Tamarin. The source code consists of an Android app implementing the stand-alone and the integrated version of the protocol. Moreover, it contains the firmware needed to flash the UWB board used for the ranging. We provide analysis scripts used to derive the reliability and security results presented in the paper.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

Our artifacts do not pose any risk for the evaluators.

A.2.2 How to access

All artifacts can be accessed via Github. The project consists of three repositories

- pure-poc[1]: contains the proof-of-concept code which can be run on Android phones equipped with UWB boards.
- pure-sec-rel[2]: contains the analysis of the proposed protocol in terms of reliability and security.
- pure-models[3]: contains the Tamarin models used to formally prove the security of the proposed EMV extension.

A.2.3 Hardware dependencies

To run the Android apps, two Android phones are necessary. For the UWB ranging the following hardware is necessary

- 2 x Qorvo DWM3000EVB
- 2 x Nordic Semiconductor nRF52 DK (pca10040)
- 2 x USB-C to USB-A adapter for connecting the boards to the phones.

Both the Android app and the UWB firmware can be tested separately in case the phones or the boards are missing. The interface between the Android app and the UWB implementation is such that:

- The Android phones write a ranging key over UART and expect timing information from the board.
- The board expects a ranging key over UART and reports the timings to the phones.

The experiments in Section A.4.2 do not strictly require this hardware because they rely on datasets that we collected. However, the listed hardware is required to execute transactions.

A.2.4 Software dependencies

The UWB firmware requires Segger J-Link to flash the boards. It can be installed following the instructions on Segger website[4]. Tamarin prover is required to verify the formal proofs.

A.2.5 Benchmarks

None

A.3 Set-up

A.3.1 Installation

Android application The Android application can easily be built and installed using Android Studio. It is necessary to enable on the phones the developer options and USB/WiFi debugging.

[1]`https://github.com/pure-uwb/pure-poc/tree/f73633f9716e42d0b8917912c72c912b479f3153`

[2]`https://github.com/pure-uwb/pure-sec-rel/tree/77cc1792f79bec02b59168ee922dfe335c972ad3`

[3]`https://github.com/pure-uwb/pure-models/tree/8aef1c0254642b476d9736ae5770c1992e620b27`

[4]`https://www.segger.com/downloads/jlink/JLink_Linux_V786e_x86_64.deb`

UWB firmware We provide a vagrant script to create a VM from which it is possible to build and install the binary on the boards.

Data Analysis We provide a python virtual environment containing all packages needed to run the analysis. Moreover, a download script can be executed to download the datasets from a publicly available link.

A.3.2 Basic Test

In order to flash the boards, please make sure that they are visible from within the VM executing the command *lsusb*. Moreover, in the VM running *nrfjprog -i* should list the identifier of the connected UWB boards. Installing the Android apps should be straightforward with Android Studio.

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): The absolute threshold necessary to limit a ghost peak attack to 2^{-49} on a Qorvo DWM3000 is 702 *PURE*. This result is proven by Experiment 1 and shown in Figure 9.

(C2): An absolute threshold of 702 achieves a 2% false rejection rate (FRR) in realistic payment scenarios with a maximum accepted relay of 46 cm. This result is proven by Experiment 2 and shown in Table 2.

(C3): The DH share is authentic and secret, and the AIP is authentic. This result is described in Section 3.3 and can be verified through Experiment 3.

(C4): PURE introduces at most 78 ms delay. This result is presented in Table 2 and supported by Experiment 4.

A.4.2 Experiments

(E1): *[10 human-minutes]: In this experiment the probability of a ghost peak attack against an absolute threshold is evaluated. We provide the dataset collected while injecting UWB packets into a DWM300 board with an increasing number of correct STS bits. The script analyses the corrected CIRs and outputs Figure 9. We do not include the code used for the data collection.*

Preparation: Following the README *pure-sec-rel/pure-reliability*, download the dataset and install the required python dependecies.

Execution: Execute *python3 nsame-analyze-plot.py –single nsame_2000_2500*.

Results: The generated plot shows the success rate of a ghost peak attack as a function of the absolute threshold used to verify a peak.

(E2): *[10 human-minutes]: This experiment analyzes the CIRs collected in contactless payment scenarios. The data collection process (not included in this experiment) is shown in Figure 10.*

Preparation: Following the README in *pure-sec-rel/pure-reliability*, download the dataset and install the required python dependecies.

Execution: Execute the *reliability_analysis.py* with the parameters listed in the README.

Results: The script prints the FRR for different accepted relay distances (46 cm, 85 cm, 95 cm).

(E3): *[30 human-minutes; 1 hour of computation]: This experiment provides the model and proofs of the Mastercard protocol extension proposed in PURE.*

Preparation: Install tamarin following the instruction on the tamarin webpage[5].

Execution: Execute tamarin as *tamarin-prover –prove EMV_Mastercard_Secure_Ranging_Ext_proof.spthy –derivcheck-timeout=0*. The proofs took 65 minutes on a computing server running Ubuntu 20.04.3 with two Intel(R) Xenon(R) E5-2650 v4@2.20GHz CPUs (with 12 cores each) and 256GB of RAM. We have limited the number of threads to 14 and the memory consumption (RAM) to 32GB. Depending on the available computational resources, this may take longer.

Results: Once the proofs finished running, a summary of each lemma is printed in the console. The lemmas *auth_to_terminal_dh*, *secrecy_SIGMA* are the additional provne lemmas related to the extension.

(E4): *[10 human-minutes] We provide the timings collected running the PoC implementation on a Samsung S21 and a Pixel 4. This experiment prints the average and standard deviation run time of each section of the protocol.*

Preparation: Install the python dependencies in *timings/requirements.txt* in the pure-poc repo.

Execution: From *pure-poc/timings* execute *python process.py*.

Results: The script outputs the mean and standard deviation runtime of each part of the protocol for the standalone and the integrated version.

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/`.

[5]`https://tamarin-prover.com/`. The easiest way to install Tamarin is through Homebrew