Jan Seedorf, Kazim Mazhar,
Florian Schwabe and Irman Omerovic

Applied Cryptography in the Internet-of-Things

Abstract

The educational chapter at hand provides a three-week practical educational unit that aims at teaching students how to apply, enhance, and benchmark state-of-the-art symmetric cryptography in an actual Internet-of-Things (IoT) operating system in a locally virtualized lab environment as well as in an actual remotely deployed lab environment. Existing open-source code will be provided to participants as a starting point. This code essentially applies AES encryption in a basic block cipher mode to IoT messages. The students will then be provided a chain of challenges to enhance this code with more advanced block cipher modes and to execute their enhanced code in a virtualized environment as well as on actual IoT hardware in a remote lab.

The high-level learning objectives for this educational chapter are for students to a) get practical hands-on experience in programming, compiling, and running C programs on an actual IoT operating system, b) learn how to use, optimize, and enhance existing C code (which is provided as open-source code) that applies symmetric AES encryption to IoT messages, and c) learn how to execute and interpret scientific experiments with the goal of benchmarking various AES block cipher modes on different IoT hardware located on a remote IoT test bed.

Keywords

Applied Cryptography, Advanced Encryption Standard (AES), RIOT-OS

Part A—Educational Considerations

1 Preface

1.1 Didactic Fundamentals

The target group for this educational chapter is computer science students at the end of their bachelor's degree program. As prerequisites, the students are

required to have essential knowledge in i) Linux-based operating systems, ii) computer programming, and iii) cryptography. Particular knowledge of the C programming language is beneficial but not strictly required as a prerequisite for this educational chapter. Students will be organized into groups that stay the same for the whole educational chapter.

The following further didactic fundamentals are envisioned for this education chapter: i) Effort in lecture hours (i.e. solving challenges in the presence of an instructor/lecturer): 12; ii) Effort for self-study material (i.e. solving challenges without the presence of an instructor/lecturer): 12; iii) Suggested Credit Points (CP): 30% of 5CP.

1.2 Learning Objectives and Competence

1.2.1 Competence

The main high-level skills envisioned for students to gain are:

- The ability to write and run C programs that apply symmetric encryption algorithms in an actual IoT operating system
- The ability to execute and interpret scientific experiments in an IoT setting

1.2.2 Learning Objectives

The overall learning objectives of this educational chapter are for students to

- get practical hands-on experience programming, compiling, and running C programs on an actual IoT operating system,
- learn how to use, optimize, and enhance existing C code (which is provided as open-source code) that applies symmetric AES encryption to IoT messages,
- learn how to execute and interpret scientific experiments with the goal of benchmarking different AES block cipher modes on various IoT hardware located on a remote IoT test bed.

2 Overview

The educational chapter at hand provides a three-week practical educational unit for teaching students how to apply, enhance, and benchmark state-of-the-art symmetric cryptography in an actual Internet-of-Things (IoT) operating system in a virtual—as well as a remote—lab environment.

2.1 Practical Teaching Approach

In the course of this educational chapter, existing open-source code will be provided to participants as a starting point. This code essentially applies AES encryption in a basic block cipher mode to IoT messages. The students will then be provided with a series of challenges to enhance this code with more advanced block cipher modes and to execute their enhanced code in a virtualized environment as well as on actual IoT hardware in a remote lab.

The operating system (OS) chosen for this education chapter is RIOT-OS (Baccelli et al., 2018). Experiments will be conducted in two kinds of lab environments by the students:

- running code under RIOT-OS locally, on a virtual Linux machine
- running code under RIOT-OS in a remote IoT test bed on actual IoT hardware (Adjih et al., 2015)

2.2 Technical and Methodical Considerations

2.2.1 Technical Considerations

Technical requirements for this educational chapter concern only virtualization, as for the remote lab technically an existing IoT lab will be used as a basis[1]. Hence, some form of Linux virtualization environment must be provided for the students, either as a virtualized server environment[2] or as a local virtualization environment running on the students' host OS[3]. The chapter will assume the latter, i.e. Debian Linux as an operating system and Oracle Virtual Box as a local virtualization environment. Adapting the chapter to a virtualized server environment should be straightforward.

2.2.2 Methodical Considerations

In terms of the project's methodical implementation, the exercises (referred to as "challenges") are suggested to be provided to students via the online learning platform Moodle[4] (or via a similar learning platform). Participants can then provide feedback via Moodle through a survey that is conducted at the end of the educational chapter. For an overall evaluation of the appropriateness and difficulty of the challenges, time tracking for each indi-

1 https://www.iot-lab.info
2 Such as e.g. VMWare ESXi-Server (https://www.vmware.com/products/esxi-and-esx.html)
3 Such as e.g. Oracle Virtual Box (https://www.virtualbox.org)
4 https://moodle.org

vidual challenge can be provided by participants, e.g. via an Excel timesheet uploaded to the Moodle learning platform.

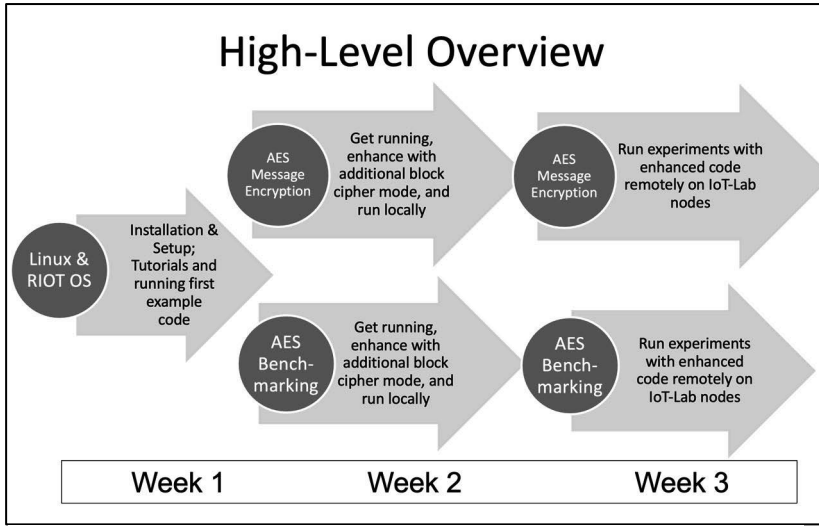

Figure 1: Weekly Overviewed of Lecture Chapter

3 Lecture Chapter Outline

The complete lecture chapter with concrete challenges for participants will be provided in the following sections. Here, a short summary of the content of the lecture chapter for each week is given (see Figure 1 for an overview):

- Week 1: Installation and setup of virtual Linux machine and RIOT-OS; Compiling first programs under RIOT-OS and conducting selected RIOT tutorials
- Week 2: Understanding and getting to run existing code (which is provided as open-source code) that applies a) AES encryption and b) AES benchmarking to IoT data in a virtual machine; modifying both types of code to automize experiments; enhancing the existing code so that additional block cipher modes can be used with AES encryption
- Week 3: Running modified, enhanced code on remote hardware in an IoT lab; executing experiments (with enhanced code) on remote hard-

ware in an IoT lab; scientifically interpreting the results of the experiments

Part B—Educational Chapter

4 Challenges Week 1

The high-level objectives for week 1 are:

- Getting more familiar with RIOT-OS by conducting some basic RIOT tutorials
- Getting RIOT-OS running as a virtualized working environment
- Being able to configure and compile a basic RIOT application with Make-Files

The content in a nutshell for week 1 is:

- Linux VM Installation & Setup
- RIOT Installation & Setup
- RIOT Tutorials

4.1 Challenge 1.1—Linux VM Installation & Setup

4.1.1 Preparation

- Download and install Oracle Virtual Box as a virtualization environment on your host OS[5].
- Download the current stable version of Debian Linux as a DVD ISO Image[6].

4.1.2 Installation and configuration

- Install Debian from the ISO Image as a virtual machine under Virtual Box with the following settings:
  - Virtual Box settings: 1GB Ram, Virtual HD Size: 25GB
  - Debian settings: Install with Apache web server and SSH Server
- Configure virtual network settings in Virtual Box to NAT, restart your new Debian virtual machine, and confirm that you have Internet access (e.g. by opening a random webpage of your choice in a browser).

5 Available at https://www.virtualbox.org

6 Available at https://cdimage.debian.org/debian-cd/current/amd64/iso-dvd/

4.1.3 Installing packages

- Get familiar with the apt-get command.
- Verify that you are able to install packages with the apt-get command, e.g. by installing the Firefox browser (if not, check e.g. that your Apt sources are configured correctly[7]).

4.2 Challenge 1.2—RIOT Installation & Setup

4.2.1 Preparation & repository Cloning

4.2.2 Configuring a first test project

- Look at the Make-File in your first_test directory and try to understand each line in it.
- Configure the Make-File so that the absolute path to the RIOT base directory is correct and that the executable output of your Make-File will be called first_test.
- Configure the main.c file in your first_test directory so that it contains a shell command named "whats_up" that will print out "the roof" when run in the shell[8].

4.2.3 Running a first test project

- Set up your network with the tapsetup command.
- Make and run your first_test project in two instances, each connected to a different tap device.
- Try out your whats_up shell command.
- Try sending messages from one RIOT instance to the other RIOT instance with the txtsnd command.

4.3 Challenge 1.3—RIOT Tutorials

4.3.1 Preparation

- Read the 'readme' for the official RIOT tutorials and ensure that your environment is properly prepared for the RIOT tutorials[9].

7 See e.g. https://wiki.debian.org/SourcesList
8 Similar to the example in https://github.com/RIOT-OS/RIOT/wiki/Creating-your-first-RIOT-project
9 See "Preparations" at https://github.com/RIOT-OS/Tutorials

4.3.2 Task execution

- Conduct tasks 1.1, 2, 3, and 4 from the official RIOT tutorials[10].

5 Challenges Week 2

The high-level objectives for week 2 are:

- Being able to encrypt and decrypt messages with AES in RIOT
- Benchmarking AES with different configurations

The content in a nutshell for week 2 is:

- Using and enhancing existing code that uses AES under RIOT for message confidentiality
- Using and enhancing existing code for benchmarking AES under RIOT

5.1 Challenge 2.1—Enhancing AES message encryption with an additional block cipher mode of operation

5.1.1 Repository cloning, running and understanding existing code

- Clone the repository at https://github.com/flori-schwa/IT-Sec2.
- Read and reproduce the following tutorials from the README.md on your local system:
  - *Part 4: AES in Electronic Codebook (ECB) mode*[11]
  - *Part 5: AES in Cipher Block Chaining (CBC) mode*[12].
  - Hint: You may have to make small adjustments in the Make-Files (e.g. RIOT path) to get everything working on your system.
- Try to understand how AES block cipher modes are being used and applied in the code to encrypt short messages.

5.1.2 Enhancing code with an additional block cipher mode

- Besides AES-ECB and AES-CBC, RIOT also supports the following cipher modes by default:

10 Located at https://github.com/RIOT-OS/Tutorials under "Tasks"

11 i.e.: https://github.com/flori-schwa/IT-Sec2/blob/master/Tutorials/Chapter_2_Crypto/04_AES_ECB_en.md

12 i.e.: https://github.com/flori-schwa/IT-Sec2/blob/master/Tutorials/Chapter_2_Crypto/05_AES_CBC_en.md

- CTR (Counter Mode): SP 800-38A[13]
- OCB (Offset Codebook Mode): RFC7253[14]
- CCM (Counter with CBC-MAC): SP 800-38C[15]
- Choose one of the modes above and write a program to encrypt + decrypt string messages with AES in that particular mode.
- Any code in the repository may be freely used as a starting point.
- The RIOT crypto documentation[16] provides an overview of all library functions necessary to solve this challenge.

5.2 Challenge 2.2—Enhancing AES Benchmarking with an additional block cipher mode of operation

5.2.1 Repository cloning and understanding existing benchmarking code

- Clone the repository at https://github.com/deus778/riot-aes-benchmark.
- Familiarize yourself with the code and get an understanding of how the benchmarking of the AES algorithm works.

5.2.2 Enhancing benchmarking code with an additional block cipher mode

- Extend the existing benchmark code to include your AES block cipher mode implementation from challenge 2.1. Note that you do not want to encrypt/decrypt messages here, instead, you want to fill the input with random bytes.
- When running the benchmark, you should be able to choose between AES-ECB, AES-CBC and the mode you implemented.

5.2.3 Running first experiments

- Run multiple benchmarks using different configurations and note how it affects the performance and significance (e.g. standard deviation) of your results.

13 https://doi.org/10.6028/NIST.SP.800-38A
14 https://datatracker.ietf.org/doc/html/rfc7253
15 https://doi.org/10.6028/NIST.SP.800-38C
16 https://api.riot-os.org/group__sys__crypto.html

6 Challenges Week 3

The high-level objectives for week 3 are:

- Getting enhanced AES message encryption to run in a remote IoT-Lab
- Executing scientific AES benchmarking experiments in a remote IoT-Lab
- Presenting and interpreting results

The content in a nutshell for week 3 is:

- Migrating enhanced AES code to the remote IoT-Lab
- Running experiments in the remote IoT-Lab
- Creating graphs from results and discussing the results observed

6.1 Challenge 3.1—Running enhanced AES message encryption in a remote IoT-Lab

6.1.1 Account Creation and SSH access

- Create an account[17] for accessing the remote FIT IoT-Lab test bed[18] (hereafter referred to as "IoT-Lab" or "remote IoT-Lab").
- Generate SSH keys, associate your SSH key with your IoT-Lab account, and test your SSH access to the IoT-Lab SSH front end at a site of your choice[19].

6.1.2 Running first experiments

- Read the introduction[20] of the FIT IoT-Lab[21] in order to understand how to run your code on a remote node in the test bed.
- Get to run your AES message encryption code[22] in a selected block cipher mode on a remote node via the web portal[23].

17 https://www.iot-lab.info/testbed/login?next=%2Fdashboard
18 https://www.iot-lab.info/docs/getting-started/user-account/
19 https://www.iot-lab.info/docs/getting-started/ssh-access/
20 https://www.iot-lab.info/docs/getting-started/introduction/
21 https://www.iot-lab.info
22 i.e. your modified and enhanced version of the code from https://github.com/flori-schwa/IT-Sec2 from last week's challenges
23 https://www.iot-lab.info/learn/

6.1.3 Running more experiments

- Run your AES message encryption code with different selected block cipher modes on different remote nodes (i.e. on different board types) via the web portal.

6.2 Challenge 3.2—Running enhancing AES benchmarking in an IoT-Lab and executing scientific experiments on a remote test bed

6.2.1 Running first manual experiments

- Get your enhanced AES benchmarking code[24] to run on the remote IoT-Lab, e.g. on different selected nodes with various selected block cipher modes.
- You can execute your code in the remote IoT-Lab either via the web portal or via SSH access.

6.2.2 Automizing experiments

- Find a way to automate the execution of a multitude of different settings to be run on different boards sequentially.
  - One way of achieving this could be to compile your code in multiple teams with different settings (e.g. with AES-128-CBC, AES-256-CBC, AES-128-ECB, AES-256-ECB, etc.) for each board and then automate the sequential execution of experiments via shell scripts
  - An alternative to achieve automized experiments could be to re-factor the benchmarking code so that it can run multiple AES block cipher modes and different key sizes in a single run, and then compile this re-factored code on different selected hardware boards
  - Hint: A combination of the methods above is certainly possible.
  - Hint: For automation of your experiments, SSH access seems to be the preferred method of code execution in the remote IoT-Lab.

6.3 Challenge 3.3—Presentation and discussion of results

6.3.1 Graph generation

- Generate result graphs from your results (e.g. by importing .csv-files into Excel).

---

24 I.e. your modified and enhanced version of the code from https://github.com/deus778/riot-aes-benchmark from last week's challenges

6.3.2 Result analysis

- Interpret and discuss your results and graphs. What can you observe and conclude?[25]

7 Conclusion

This educational chapter provides a practical three-week course for computer science students with the objective of learning in depth how to apply, enhance, and benchmark state-of-the-art symmetric cryptography in an actual Internet-of-Things (IoT) operating system. Experiments are conducted in a locally virtualized lab environment as well as in an actual remotely deployed lab environment. The basis for this educational chapter is existing code as a starting point, which has been developed specifically for this course and is provided as open-source code. During the course, students enhance this code and run experiments with their enhanced code in weekly challenges, which have been presented in detail.

This educational chapter was successfully conducted in the winter term 2021/2022 at the University of Applied Sciences Stuttgart, Germany (HFT Stuttgart). The detailed challenges provided here contain the experiences of executing this educational chapter in practice, i.e. are aligned and adjusted based on experiences gained, in particular with respect to the time needed by students and, hence, the allocation of challenges to weeks. In general, all challenges were solvable for the students.

Additional resources such as a challenge-evaluation Excel sheet or virtual machines with intermediate states for the individual challenges will be provided by the authors upon request.

Acknowledgments

This work has been supported by the Digilab4U (Open Digital Lab for You) project[26], funded by the German Federal Ministry of Education and Research (BMBF) under No. 16DHB2112.

The authors are deeply grateful to Peter Kietzmann, Cenk Gündogan, Matthias Wählisch, Thomas Schmidt, and the whole RIOT team at HAW

25 For an example of a scientific paper that presents and discusses such scientific results in relation to running cryptography on IoT hardware, refer to (Kietzmann et al., 2021).

26 https://digilab4u.com

Hamburg and FU Berlin for their help in getting us started with cryptography under RIOT-OS and in developing this educational chapter.

References

Adjih, C., Baccelli, E., Fleury, E., Harter, G., Mitton, N., Noel, T., Pissard-Gibollet, R., Saint-Marcel, F., Schreiner, G., Vandaele, J., & Watteyne, T. (2015). FIT IoT-LAB: A large scale open experimental IoT testbed. In *2nd World Forum on Internet of Things (WF-IoT) (WF-IOT '15)*. DOI:https://doi.org/10.1109/WF-IoT.2015.7389098

Baccelli, E., Gündogan, C., Hahm, O., Kietzmann, P., Lenders, M., Petersen, H. Schleiser, K., Schmidt, T.C., & Wählisch, M. (2018). RIOT: an Open Source Operating System for Low-end Embedded Devices in the IoT. *IEEE Internet of Things Journal, 5 (6)*, 4428–4440.

Kietzmann, P., Boeckmann, L., Lanzieri, L., Schmidt, T. C., & Wählisch, M. (2021). A Performance Study of Crypto-Hardware in the Low-end IoT. In *International Conference on Embedded Wireless Systems and Networks (EWSN)*, https://www.ewsn.org/file-repository/ewsn2021/Article8.pdf

Authors

Prof. Dr. Jan Seedorf
University of Applied Sciences, Stuttgart
(HFT Stuttgart)
Schellingstr. 24
70174 Stuttgart, Germany
https://www.hft-stuttgart.de/p/jan-seedorf
jan.seedorf@hft-stuttgart.de

Kazim Mazhar
University of Applied Sciences, Stuttgart
(HFT Stuttgart)
Schellingstr. 24, 70174 Stuttgart, Germany
https://www.hft-stuttgart.de
kazim.mazhar@hft-stuttgart.de

Florian Schwabe
University of Applied Sciences, Stuttgart
(HFT Stuttgart)
Schellingstr. 24,
70174 Stuttgart, Germany
https://www.hft-stuttgart.de
florian.schwabe@hft-stuttgart.de

Irman Omerovic
University of Applied Sciences, Stuttgart
(HFT Stuttgart)
Schellingstr. 24,
70174 Stuttgart, Germany
https://www.hft-stuttgart.de
irman_omerovic@protonmail.com