DEMO: RESCURE: Retrofit Security for Critical Infrastructures

Mario Münzer
Stefan Ilić
muenzer@technikon.com
ilic@technikon.com
Technikon Forschungs- und Planungsgesellschaft mbH
Villach, Austria

Georgios Selimis
Rui Wang
Georgios.Selimis@intrinsic-id.com
Rui.Wang@intrinsic-id.com
Intrinsic ID
Eindhoven, Netherlands

Frans M.J. Willems
Lieneke Kusters
F.M.J.Willems@tue.nl
C.J.Kusters@tue.nl
Eindhoven University of Technology
Eindhoven, Netherlands

ABSTRACT

Low-cost interconnected devices, so-called Internet-of-Things (IoT), commonly have no dedicated or posses insufficient hardware security features. This is challenging, as IoT devices are becoming an integral part of critical infrastructures providing much needed additional functionality but also creating a significant security threat to the infrastructure. Due to the scale of IoT integration in critical infrastructures, a key issue in initial deployment and replacing of the devices is often the cost. RESCURE delivers a low-cost IoT security solution based on unique hardware anchors. More precisely, we are using PUFs (Physical Unclonable Function) technology based on SRAM (Static Random-Access Memory), which provides a unique and unclonable identifier as well as a root key for each device. As SRAM-PUFs-based approaches require no additional specialized hardware, it also presents a viable approach of retrofitting existing embedded devices already used.

KEYWORDS

Internet-of-Things, Embedded systems security, Physically Unclonable Function, End-to-end encryption, multiple observations

1 INTRODUCTION

The IoT device, as defined by ARM [1], is a piece of hardware mostly equipped with a sensor transmitting data over the internet. As cost saving measure, IoT devices are often based on small, inexpensive, and resource constrained chips. This design choice helps with scalability, as IoT devices usually have wide deployment, but reduces support for existing security solutions which often rely on dedicated hardware security features. Furthermore, IoT devices are often using M2M (Machine-to-Machine) communication, have 24/7 uptime and are deployed in field, making them harder to access physically and replace. Due to given risk factors, many recent attacks target on IoT devices as the first step in order to compromise the underlying infrastructures [7]. RESCURE is a European research project, focused on developing low-cost IoT security solution for device protection and secure communication while keeping selected approach applicable to existing devices. The US Department of Homeland Security [6] recommends that devices rely on hardware with incorporated security features, e.g, Arm TrustZone [2]. In RESCURE, to cover a wide range of devices while following set recommendations, we focus on the most commonly available hardware component of IoT devices, namely the SRAM. Generating the root key using SRAM-PUF technology is a low-cost alternative to storing a key in protected memory. Furthermore, since the SRAM is already available on any IoT device, our scheme supports retrofitting the existing hardware to a secure system.

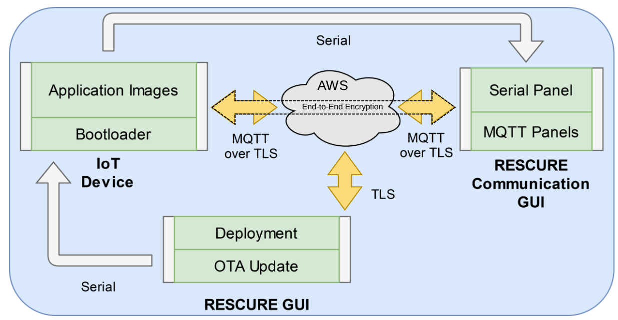

Figure 1: OTA update & deployment flow of demonstrator

2 RESCURE

Due to inherent process variation during the manufacturing of SRAM, small and uncontrolled variations occur in the silicon material giving each SRAM a unique initial state [4]. The initial state is not constant but varies between each SRAM power up phase to a certain degree, no matter the production line. Nevertheless, as the intra-subject difference is much less pronounced than the inter-subject difference, even between SRAMs from the same manufacturer, this enables us to uniquely identify devices based on the initial state and generate appropriate root key [3]. In order to turn the noisy SRAM initial state into a reliable and device-unique root key, a helper data scheme is applied on top of SRAM-PUF. For error correction, we implemented an algorithm based on concatenation of BCH code (15,7,5) and repetition code (7,1). Given the 5% intersubject difference observed on the device at room temperature, our helper data scheme can regenerate the root key with the error rate of 10^{-7}.

In RESCURE, we implemented three distinct security features based on the root key extracted from the SRAM-PUF: 1) secure connection and device identification with the cloud; 2) E2E (End-to-end) encrypted communication with backend; and 3) secure OTA (Over-the-air) software/firmware update.

The first usage of SRAM-PUF root key, in RESCURE, is a runtime generation of SECP256R1 key pair. During the device enrolment phase, we capture the key pair calculated on the device and generate an appropriate device certificate, which we register with the cloud provider (in our case Amazon Web Services). This enables us to tie the key pair, and therefore SRAM-PUF root key, with a thing ID, a unique identifier with whom Amazon identifies the device.

This approach prevents the cloning of software as a different key would be generated on a different device. Also, as an added benefit, generating keys at runtime avoids the possibility of their extraction from flash and the need for complex key protection schemes. The key generation itself is based on seeding HMACDRBG (hash-based message authentication code - deterministic random bit generator) using the root key and using it as an input to SECP256R1 generation.

The key pair required for E2E encryption is also generated in the same manner. Meanwhile, the backend (in our case node.js script running on PC) generates its own SECP256R1 keypair. Both the device and the backend register to the cloud provider and subscribe to the same MQTT topic. Once they exchange their public keys, both sides generate shared secret using ECDH key exchange protocol. The data encryption and authentication between these endpoints is based on this shared secret. Thus, another security layer is established on top of the TLS connection, preventing cloud provider access to the unencrypted data.

The last feature based on SRAM-PUF root key is the OTA firmware update. New application images, during transit and storage at Amazon S3 servers, are encrypted using ChaCha20 symmetric cipher. To prevent an attacker from obtaining this symmetric key from the device flash, we encrypt it using the root key. In all of the given cases, the root key is zeroed out immediately after use. Further, even if we reserve and use a certain amount of the SRAM, this space is only needed within the bootloader and released right after the root key generation. In RESCURE we have also studied new methods that can improve reliability of the root key reconstruction. As such, we have developed a new scheme that we call the multiple observations helper data scheme. The scheme can construct helper data that is based on multiple SRAM-PUF observations instead of a single observation. The more observations are used, the more reliable the key reconstruction is. We have built a MATLAB GUI that explains the functionality of the scheme, and demonstrates its performance through simulations, see Section 3.2.

3 DEMONSTRATION

We present the functionality developed in RESCURE by showcasing two distinct demonstrators: initial application deployment on board and procedure of OTA firmware update (D1) and MATLAB application that demonstrates the multiple observations helper data scheme (D2). For each demonstrator, the authors (presenters) will explain the underlying usage of SRAM-PUF and achieved results.

3.1 D1: Deployment and OTA Firmware Update

In this demonstrator, we aim to illustrate deployment and OTA update work-flow using the B-L475E-IOT01A board [8], based on STM32L475VG [9] MCU. The architecture of the system is presented in Figure 1. The demonstration starts by flashing the initial application image, bootloader and necessary meta-data using RESCURE GUI. Once run, the application automatically establishes a connection to AWS based on SECP256R1 key pair generated by using the SRAM-PUF root key at runtime. Using this connection, we send periodic sensor data (temperature data) to a predetermined MQTT topic. In the second part of the demonstration, we create an OTA update job using RESCURE GUI. The OTA Update Agent running on the board is notified by AWS, on a dedicated MQTT topic, that a new update is available and starts the update procedure. It regenerates SRAM-PUF root key and decrypts symmetric encryption key stored in flash. Using this key, the downloaded chunks, representing the firmware update, are decrypted as they arrive using ChaCha20. When the download is finished, the reboot of the IoT device is triggered and the new application is executed. For this demonstration purpose, the updated application image enables, as an additional functionality, end-to-end encryption based on SRAM-PUF.

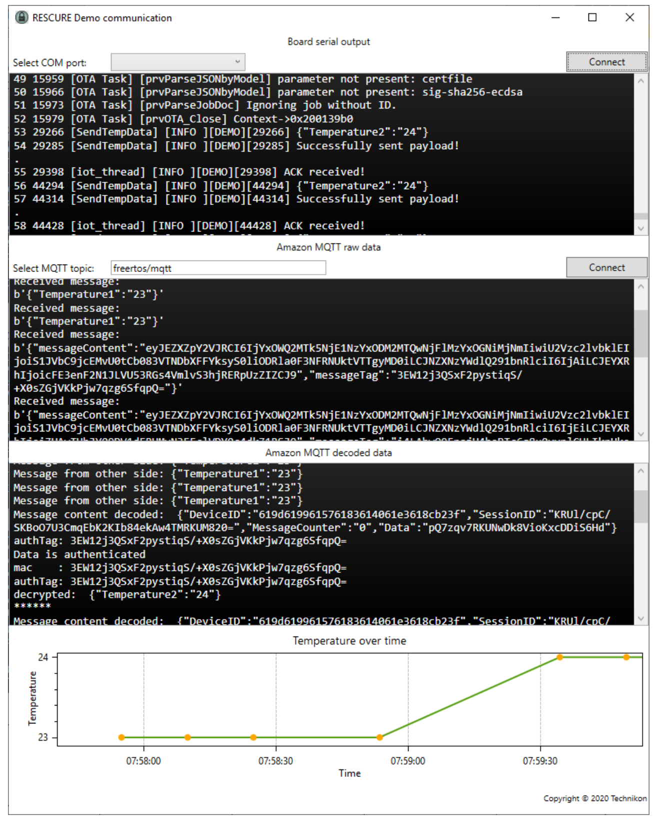

Figure 2: RESCURE communication GUI

As presented in Figure 2, the first section of the RESCURE communication GUI is displaying the raw data captured by the IoT device, which in turn represents the temperature sensor data in its unencrypted state. Following, respectively in the second section of the communication GUI, the encrypted messages received are displayed, and represents the communication traffic at AWS' side. Following further, in the third section of the GUI, finally the decrypted messages are displayed, which in turn represents the end-point of the end-to-end communication. Resulting, it is clearly demonstrated that the messages cannot be decrypted on the AWS' side and in turn are only visible to the IoT and backend. Additionally, we present the temperature sensor data, which is the example data transferred between IoT and backend, in a corresponding graph.

3.2 D2: Multiple Observations Helper Data Scheme

In this demonstrator, we aim to illustrate performance and security of the multiple observations helper data scheme. The demo is running in MATLAB, where we use a statistical model[5] to simulate the SRAM-PUF observations. Furthermore, we use a concatenated LDPC(256,128) and repetition code as the error-correcting code. During the demo we visualize the helper data construction and key reconstruction in real-time. We vary the number of used observations, and plot the resulting reconstruction error rate (FER) in real time. We show that FER decreases when more observations are used. Furthermore, it is possible to vary the rate of the error-correcting code, by adjusting the used repetition rate. Note that a smaller rate means increased efficiency of the scheme, since less SRAM cells are required to achieve the same key length. The simulation results show that a similar FER can be achieved with smaller repetition rate and thus considering multiple observations can improve efficiency of the scheme.

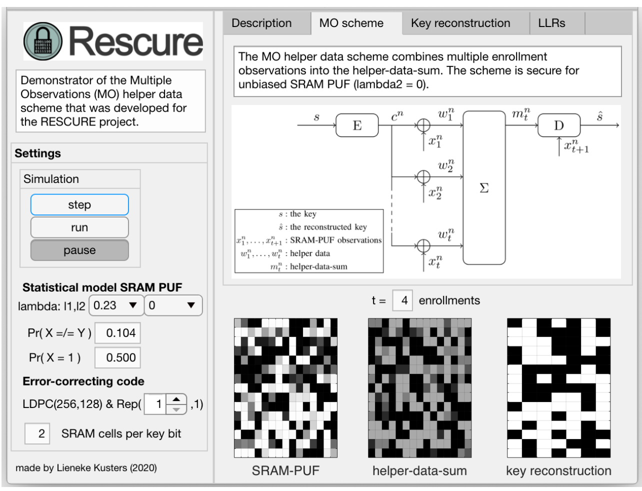

Figure 3: RESCURE MATLAB GUI

Finally, we calculate the log-likelihood ratios (LLRs) after observing the new helper data, both for the decoder and for an attacker. For the decoder, it shows how the updated helper data (based on more observations) improves the reliability of the estimated code bits. For the attacker, it represents the information leakage about the code bits. The result shows that for an unbiased SRAM-PUF, the LLRs of the attacker are constant and do not change when more observations are used. Therefore, it should convince a viewer that no leakage occurs as a result of the multiple observations helper data for unbiased SRAM-PUFs. Instead, for a biased SRAM-PUF the derived LLRs show that information about the code bits is leaked to an attacker. Therefore, the scheme is not secure in case of biased SRAM-PUFs.

A screenshot of the MATLAB GUI is shown in Figure 3.

4 CONCLUSION

We presented SRAM-PUF based security enhancements developed in RESCURE. The objective of this project is to provide a suitable and cost-effective way to retrofit existing devices by adding tamper-protection, secure storage, end-to-end communication encryption, unclonable ID and device authentication. We aim to increase the security of IoT critical infrastructures providing solution applicable to a variety of devices including low-end, resource-constrained devices.

ACKNOWLEDGMENTS

This work was partially supported by the Eurostars-2 joint programme with co-funding from the European Union Horizon 2020 research and innovation programme under the grant agreement E11897 RESCURE "Retrofit Security for Critical infrastructures". Furthermore, this work was also partially supported by the Austrian Ministry for Transport, Innovation and Technology under the framework of "IKT der Zukunft" with the FFG grant agreement project 865233.

REFERENCES

[1] ARM. 2020. *IoT Devices.* Retrieved April 08, 2020 from https://www.arm.com/glossary/iot-devices

[2] ARM. 2020. *TrustZone Technology.* Retrieved April 03, 2020 from https://developer.arm.com/ip-products/security-ip/trustzone

[3] Jorge Guajardo, Sandeep S. Kumar, Geert-Jan Schrijen, and Pim Tuyls. 2007. FPGA Intrinsic PUFs and Their Use for IP Protection. In *Cryptographic Hardware Embedded Syst. - CHES.* 63–80.

[4] Daniel E. Holcomb, Wayne P. Burleson, and Kevin Fu. 2009. Power-Up SRAM state as an identifying fingerprint and source of true random numbers. *IEEE Trans. Comput.* 58, 9 (2009), 1198–1210. https://doi.org/10.1109/TC.2008.212

[5] Roel Maes, Pim Tuyls, and Ingrid Verbauwhede. 2009. A soft decision helper data algorithm for SRAM PUFs. In *IEEE Int. Symp. Inf. Theory - ISIT.* 2101–2105. https://doi.org/10.1109/ISIT.2009.5205263

[6] U.S. Department of Homeland Security. 2016. *Strategic Principles for securing the Internet of Things (IoT).* Retrieved March 19, 2020 from https://www.dhs.gov/sites/default/files/publications/Strategic_Principles_for_Securing_the_Internet_of_Things-2016-1115-FINAL....pdf

[7] Ioannis Stellios, Panayiotis Kotzanikolaou, Mihalis Psarakis, Cristina Alcaraz, and Javier Lopez. 2018. A Survey of IoT-enabled Cyberattacks: Assessing Attack Paths to Critical Infrastructures and Services. *IEEE Communications Surveys I& Tutorials* PP (07 2018), 1–1. https://doi.org/10.1109/COMST.2018.2855563

[8] STMicroelectronics. 2020. *B-L475E-IOT01A.* Retrieved April 08, 2020 from https://www.st.com/en/evaluation-tools/b-l475e-iot01a.html

[9] STMicroelectronics. 2020. *STM32L475VG.* Retrieved April 08, 2020 from https://www.st.com/en/microcontrollers-microprocessors/stm32l475vg.html