Software Implementation of a Secure Firmware Update Solution in an IOT Context

Lukas KVARDA[1], *Pavel HNYK*[1], *Lukas VOJTECH*[1], *Zdenek LOKAJ*[2], *Marek NERUDA*[1], *Tomas ZITTA*[1]

[1]Department of Telecommunication Engineering, Faculty of Electrical Engineering, Czech Technical University in Prague, Technicka 2, 166 27 Prague, Czech Republic
[2]Departmentf of Applied Informatics in Transportation, Faculty of Transportation Sciences, Czech Technical University in Prague, Konviktska 20, 110 00 Prague, Czech Republic

kvardluk@fel.cvut.cz, hnykpav1@fel.cvut.cz, vojtecl@fel.cvut.cz, lokaj@fd.cvut.cz, nerudmar@fel.cvut.cz

DOI: 10.15598/aeee.v14i4.1858

Abstract. *The paper presents the secure delivery of firmware updates to Internet of Things (IoT) devices. Additionally, it deals with the design of a safe and secure bootloader for an Ultra High Frequency Radiofrequency Identification (UHF RFID) reader. A software implementation of a secure firmware update solution is performed. The results show the implementation of the Advanced Encryption Standard (AES) encryption into existing IoT devices, e.g. UHF RFID reader, requires at least 49.7 kB of flash memory and 10 kB of RAM, and therefore there is space to integrate even more security features into existing devices without hardware upgrade.*

Keywords

Firmware, IoT, security.

1. Introduction

The Internet of Things (IoT) has become an important phenomenon due to the rapid pace of technological development. There is a constant stream of new smart devices to the market, including smartphones, smartwatches, smart refrigerators and many more. If this pace continues, IoT technology is soon adopted by many other industries including mobility, home automation, connected life and energy.

The main idea behind this trend is to connect things (sensors, actuators, lights, refrigerators, etc.) to existing network infrastructure [1] and [2]. The producers of IoT devices capitalise on the enthusiastic acceptance of this new technology by consumers and accelerate the production processes to shorten the time to market. The rapid pace of development has implications for the final quality of the devices which often break down due to Firmware (FW) or Software (SW) defects [3]. In a continuous development environment, FW has to be frequently updated for several reasons. The first one is the need to fix FW and SW bugs after the product has been released to market. Another reason is the addition of a new feature to a product that is already being used. A third reason, which is probably the most important, is the need to protect the devices against cyber-attacks. It can be for instance an attempt to manipulate an automated driving system which is a modification of the time parameter when the car starts to brake. If a default value 0.01 s is changed by an attacker to 1.1 s, a major collision can occur. Another example is a sophisticated sensor modification attack involving a change to the FW which affects the sensing parameters. These attacks are very dangerous because they can be undetected if no deeper analysis is done. Therefore, more attention has to be paid to these threats.

Updating FW remotely saves money and time because the customer does not need to send the device to the producer every time an update is needed. Gartner [4] predicts that approximately 20.8 billion of such devices will be in use by 2020.

The disadvantage, which we have hinted at, is the vulnerability to attacks targeting the update process when security is underestimated.

The rest of the paper is organized as follows: Section 2. describes recent update solutions, Section 3. presents safety and security issues, Section 4. deals with implementation details, Section 5. describes

measurement and results. Conclusion is presented in Section 6.

2. Today's Update Solutions

We distinguish several types of IoT devices based on applications, and each of these types requires a specific FW update solution. For example, Jurkovic and Sruk [5] deal with the question of using the Internet to conduct private experiments remotely – Remote Laboratories (RLs). These experiments take into account remote access including FW update of experimental test fixtures. To access the Raspberry Pi from the Internet, a Transport Layer Security (TLS) cryptographic protocol with a unique certificate and a server and client key is proposed. Another way to deliver updates is via an existing mobile network, as described in [6] where the update solution consists of GPRS modules connected on one side to a PC and on the other side to the device to be updated. The same technology is used in the solution proposed in [7], except that instead of GPRS modules, two inexpensive GPRS-enabled phones are used. The update solution consists of a controller (a PC) and a remote standalone unit which contains the 8-bit microprocessor to be updated. A drawback of these systems is that the speed decreases logarithmically with increasing distance from a GSM transmitter. Paper [8] deals with the secure delivery of FW updates to devices connected to a home network. The proposed network architecture consists of two basic nodes, a FW manager and a server. The firmware installed on the individual devices is managed by the FW manager which is stored in the home gateway and which can be connected to the Internet. The FW manager also provides FW data to the server which takes care of the uploaded FW from producers and provides it to the FW manager. The FW manager relies on hash chains to verify the integrity and authenticity of the image file. To encrypt the FW image file, a session key is generated using identity-based cryptography and bilinear pairing technology. The most vulnerable spot in this model is found between the FW manager and the device when using wireless technology to transmit data. Zaware and Shinde [9] proposed an implementation of the remote control and update of devices over Wi-Fi from any device (PC, tablet, mobile phone). To secure the communication channel, Wired Equivalent Privacy (WEP) encryption is used. This can, however, be easily broken, so Wi-Fi Protected Access (WPA) or WPA2 encryption is a better solution. As this solution does not address the need to secure the content that is transmitted (i.e., a FW image file), it is not suitable for the remote update of FW. Another method of updating IoT devices remotely is proposed in [10]. It relies on the use of a smartphone with Bluetooth connectivity. Field-Programmable Gate Arrays (FPGAs) can also be updated remotely [11], although given the price and the overall complexity of the Hardware (HW) and SW design as compared to single-chip processors, this is not often demanded.

As it is clear from the differences between the methods described above, none of the solutions can be considered perfect. Instead, we need to select the most appropriate method for each specific configuration of devices, taking into consideration the required level of security, memory size, computing power and power supply, while reflecting the need to protect assets, maintain asset value and ensure the desired quality of service.

3. Safety and Security

In the last few years we have seen an increase in the number of attacks targeting the FW of IoT systems as opposed to the operating system or the applications themselves, as neither antivirus software nor the operating system can detect low-level FW modification attacks. The attacks are also motivated by the fact that FW can contain code, data, calibration values, shared secrets and other important information. Updating FW remotely is an ideal solution for the distribution of security patches which eliminate weak spots detected in previous FW versions.

3.1. Security Threats to In-field Firmware Updates

When a FW image file is transmitted over an insecure channel and stored on a device, security issues may occur. Figure 1 shows a typical in-field firmware update process, broken into three sections based on location of an attack.

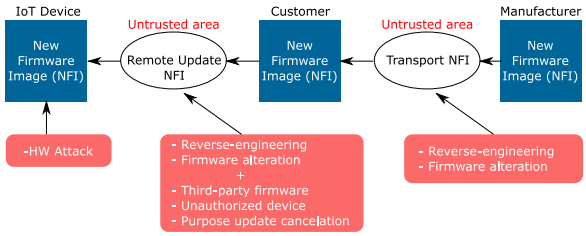

Fig. 1: In-field firmware update process (adapted from [12]).

Manufacturer – Customer – the manufacturer produces a new version of the FW image file (NFI) and distributes it over an untrusted network (i.e., the Internet) to the customer. The communication may be eavesdropped; an attacker takes hold of the entire FW image file which can then be reverse-engineered and

sensitive information can be extracted from it. The file can also be modified and returned to distribution.

Customer – IoT device – an update is distributed over an untrusted network, so eavesdropping is also possible. Unlike the NFI, additional risks such as the risk of loading unauthorised FW, the risk of loading FW onto unauthorised devices or the risk of intentional abortion of the update process can occur. Therefore, attacks are most likely to happen precisely in this section.

IoT device – an attack targeting the FW update process may involve the HW device itself. Such an attack is possible only when the device is in physical possession of the attacker. Additionally, the attacker must have access to laboratory equipment such as an electron microscope or a scanning probe microscope. Using this equipment, the attacker can monitor analogue buses, read data directly from microprocessor memory or use electromagnetic radiation to launch cryptanalysis attacks. Such attacks are very complex compared to what can be gained by them. This paper does not deal with these attacks.

3.2. Safety and Security Measure

Measures to ensure protection against the threats mentioned above fall into two categories based on what they address: safety and security.

1) Safety Measures

These methods operate at the protocol level and are aimed at preventing other types of attacks not eliminated by security measures. For example, these measures can protect the IoT environment against the aforementioned attacks which target the FW update process and which can cause the FW image file to become corrupted, truncated or incomplete. If a corrupted FW image file is loaded onto a device, the device can have a failure. There may also be damages to assets or even fatalities. To detect attacks, methods such as the following can be used:

- error detection and correction Cyclic Redundancy Check (CRC),
- block ciphers,
- packet acknowledgement.

These methods, however, cannot eliminate the threats, so they have to be combined with memory partitioning:

- single Banked Partitioning (SBP),
- dual Banked Partitioning (DBP).

The idea of SBP and DBP is that during the update process, a copy of a working FW version is stored in the Microcontroller Unit (MCU). If only a portion of the packets is transferred (e.g. due to an intentional abortion of the FW update process), the current FW is not overwritten with the corrupt FW image file. This means that two FW image files (the original and the updated version) have to be stored in the MCU at the same time. This requires additional processor memory, which results in higher cost.

2) Security Measures

In order to secure the FW update process, data confidentiality, authenticity and integrity must be ensured in addition to transmission security.

Integrity – the image file generated by the manufacturer has not been tampered with before this image file is received by the device. To detect attacks, methods such as the following can be used:

- Hash function – a fingerprint of an FW image file is obtained using a hash function. The fingerprint is attached to the data transmitted. Once the bootloader received all the data, it computes its own fingerprint and compares it with the one received. If the two are equal, the FW is not modified. A downside of using simple FW hashes is that an attacker can modify the file and subsequently compute a fingerprint. In this case, the bootloader is not able to detect that the FW has been manipulated.
- Digital signature – a FW fingerprint is computed first, then encrypted using a private key (asymmetric encryption). The signature thus obtained is attached to the FW image file and sent to the device. The bootloader decrypts the digital signature using a public key and performs a comparison as when using a hash function.
- Message Authentication Codes (MACs) – these are similar to digital signatures except a private key is used to encrypt and decrypt the fingerprint (symmetric encryption). A disadvantage of MACs as compared to digital signatures is that as anyone can verify a MAC, anyone can create one.

Authentication – refers to the verification of the origin of a message. It checks that both the FW image file and the target device originate from the manufacturer. The MAC and digital signature methods do permit this because by successfully deciphering the fingerprint, they confirm that the source is authorised.

Tab. 1: Asymmetric and symmetric encryption compared.

	Symmetric	**Asymmetric**
Key type	Private	Private, public
Key generation	Simple (randomly generated)	Complicated (special structure, expensive)
Key distribution	Difficult	Easy
Time to process	Shorter	Longer
Implementation	Easy	Complicated

Privacy – in the case of a remote FW update process this safety attribute helps to keep the FW image file confidential and prevent from unauthorised reading and the application of reverse engineering. The data is protected by encrypting the FW image file using a predetermined symmetric-key algorithm or, alternatively, asymmetric key encryption. Both types of encryption are used to ensure data confidentiality along with methods ensuring data integrity or authentication. A basic overview of these is depicted in Tab. 1.

Although the content of the FW image file is unknown to the attacker, a real threat is FW malicious modifications. The attacker might find the location of the requested variable and achieve the desired results in further iterative attempts. Such an attack is indeed very difficult to implement, but not impossible. For this reason, encryption is often combined with MACs.

4. Implementation Details

As mentioned earlier, security features have to be implemented into IoT devices. The bootloader of the current Intelligent UHF RFID reader does not have any. We decided to add secure bootloader functionality to the device. There are essentially two options: software or hardware. The advantages of a HW implementation include:

- support for various encryption methods (AES, Data Encryption Standard (DES) and 3DES) without CPU intervention,
- lower computing time (only a few hundred cycles per block as opposed to several thousand cycles in SW),
- support for a One-Time Programmable (OTP) array for secure key storage,
- on-chip Random Number Generator (RNG),
- lower power consumption – operates at lower CPU frequency.

As the design is not new but is a finished device, we have focused on a SW security solution for its lower cost (no changes to board layout, no need for a new MCU, ease of implementation). An implementation of Image file encryption, i.e. a PC application is developed in order to encrypt the firmware image file, Fig. 2, and Device flash, i.e. a PC application is developed in order to load the firmware image file onto the device, Fig. 3, is performed in order to flash the device with an encrypted firmware image file. The application of Image file encryption can generate an AES encrypted image file with configurable key length. The user only selects an unencrypted firmware image file and fills in the encryption key and initialisation vector. The application of Device flash sends the image to the device over UDP with a custom service protocol.

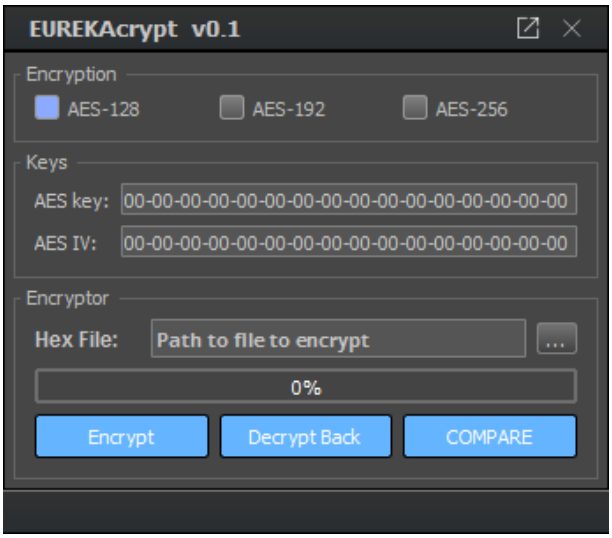

Fig. 2: A PC application to encrypt the firmware image file.

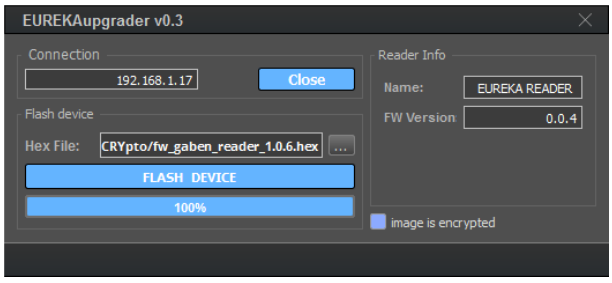

Fig. 3: A PC application to flash the device with a new firmware image file.

In the present version, an AES-based system in Cipher Block Chaining (CBC) mode with configurable key length for testing purposes is used. This is sufficient for the current application, but an AES Counter

with CBC-MAC (CCM) implementation may be explored in the future to ensure integrity and authenticity. An implementation of a secure bootloader is developed for the existing hardware-based Inteligent UHF RFID Reader.

4.1. AutoEPCIS UHF RFID Reader

The reader is equipped with 4 antenna ports (SMA connectors); each port can deliver a maximum output of 29 dBm, Fig. 4. The reader can be powered through a DC connector (5 V) or via PoE (48 V). The reader has an Ethernet module and RS232 interface which is integrated into a GPIO connector. The reader supports low-level reader protocol (LLRP) and a custom protocol via UDP and the RS232 interface.

The core of the reader consists of a PIC32MX340F512H microprocessor, which controls the RFID chip, a W5500 chip for Ethernet connectivity, which is considered to be unattackable [13] and other peripherals. The MCU operates at 80 MHz, has a 512 kB flash memory and a 32 kB RAM; there is no external memory.

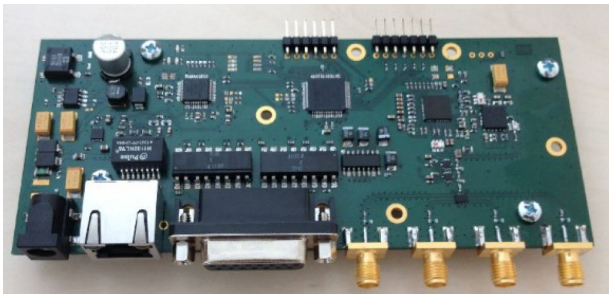

Fig. 4: AutoEPCIS UHF RFID reader – the view from above.

4.2. The Old Bootloader

The old version of the bootloader does not provide any security features. A firmware image file takes up over half of the flash memory, preventing the use of SBP whereby the memory is divided into two partitions, one being used by the existing firmware and the other (Buffer zone) by the new firmware. Without SBP, a new firmware version is loaded directly onto the device. If an error occurred during the load process, there is no way of reverting to the original firmware version. The bootloader has a 34 kB flash memory and takes up 5.65 kB of RAM. The firmware takes up 275 kB of flash memory and 19 kB of RAM.

4.3. The Redesigned Bootloader

In the new version of the firmware, the web server is removed and the reader is controlled only via LLRP. This results in a much lighter version of the firmware which permits the use of the SBP method.

The implementation of AES encryption is based on the library written by Brad Conte [14]. It is chosen for two reasons, one being easy implementation and the other being freedom from use restrictions as the implementation is released into the public domain. An automatic encryption detection algorithm for testing purposes is also developed. The PC application can send various encrypted image files without letting the device know the key length selected to encrypt the image. As a result, the device can be flashed with one command only (DATA_flash).

We use our own protocol to send encrypted data to the device; the same protocol is used to control the RFID reader. Encrypted data is sent to the device via UDP, the device checks if there is a DATA_flash command in the protocol header – if yes, incoming packet is decrypted and saved to a temporary flash storage area. Once the entire incoming image file is successfully saved to the temporary flash storage area, a consistency check is run; CRC-16 is used to verify this action. If an error occurs during the flash procedure, the device reports the error to the host based on a defined error code chart and the flash procedure has to start over. The host can choose to stop the flash procedure at any time; the device reverts to the last successfully flashed firmware version.

The memory utilisation of the working version of the FW is reduced to 151 kB of flash memory and 16.38 kB of RAM. The bootloader takes up 49.7 kB of flash memory and 9.95 kB RAM.

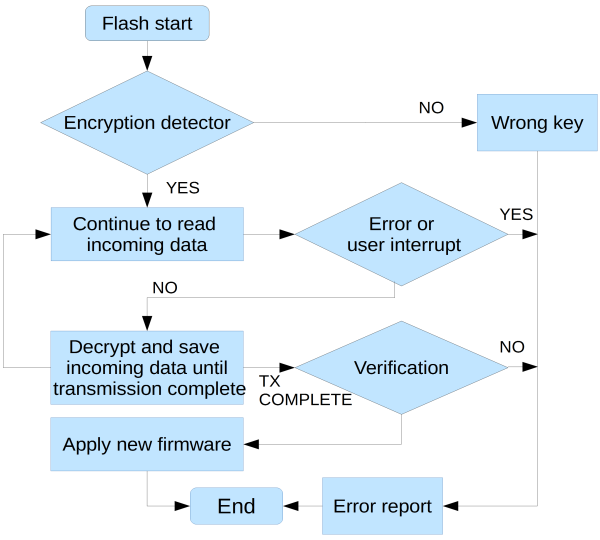

Fig. 5: Device flash diag.

4.4. Keys Management

As we use symmetric encryption (AES CBC), the same key has to be available to both the bootloader and the encryption application on the PC, Fig. 3. In the present implementation, the key is stored in the bootloader flash memory space and a code-protect mode is used to prevent unauthorised readout. Additionally, the JTAG and debug interfaces are disabled. The key cannot be modified in any other way than by rewriting the bootloader with a new key using a PICkit programmer. We are aware that this solution is less than ideal and the key should be stored on secure HW; however, the proposed solution of storing the key is sufficient considering the value of the assets. Using a PICkit programmer, the device can only be erased (the boot and program flash memory cannot be read or modified until the device is completely erased).

5. Measurements and Results

A test bench is designed to validate the functionality of the bootloader and the bootloader with a FW image file encryption capability based on an existing AutoEPCIS UHF RFID reader v.2. It consists of an update terminal, the IoT device to be updated and a router (Mikrotik RB2011UiAS-2HnD-IN). A desktop PC is used as the update terminal: it encrypts the FW image file and sends it to the device through the router. A USB-UART adapter for debugging, an Ethernet cable for data communication and a PICkit 3 programmer for the initial upload of the bootloader to the MCU are also needed. These components are shown in detail in Fig. 6. Both the bootloader and the firmware are coded in the MPLABX IDE v3.30 environment by Microchip Technology Inc. The compiler MPLAB XC32 v1.34 is used. The PC application which can flash the device with a new firmware image file, Fig. 3, and the application which can encrypt the firmware image, Fig. 2, is developed in QTcreator 3.3.1 (open source).

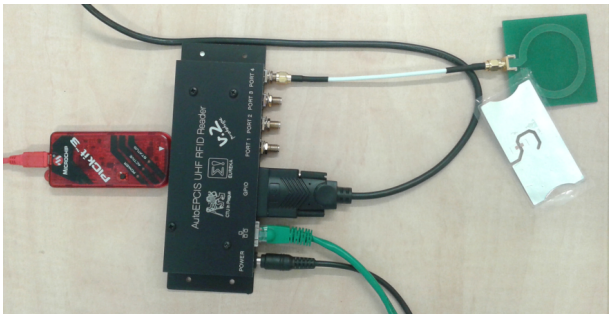

Fig. 6: Test bench – details of the RFID reader connection.

We compared the RAM and flash memory utilisation of the bootloader with and without encryption, Fig. 7. The microprocessor has an additional 12 kB of flash memory. The memory utilisation of the bootloader is almost three times higher than of the flash MCU boot. This is due to the large library for the W5500 chip which is needed for communication over Ethernet. We deal with this problem by remapping the bootloader starting address as shown in [15]. The bootloader is placed at the end of the flash memory space, as this is easier to implement and there is no need for a special compiler configuration. Considering the size of the current program (151 kB) and the total size of the flash memory (512 kB), we opted for a 100 kB bootloader. The memory is twice bigger than necessary to allow for future extension. There is also sufficient memory for future FW development (261 kB).

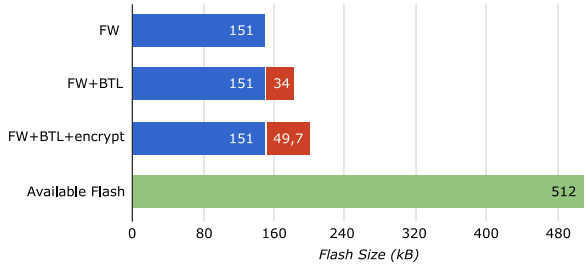

Fig. 7: Flash memory utilisation by bootloader type.

Figure 8 shows RAM utilisation by bootloader type. The added bootloader uses 5.65 kB of RAM, while the bootloader with the encryption capability uses 10 kB of RAM. The difference between encryption and no encryption is 4.3 kB of RAM.

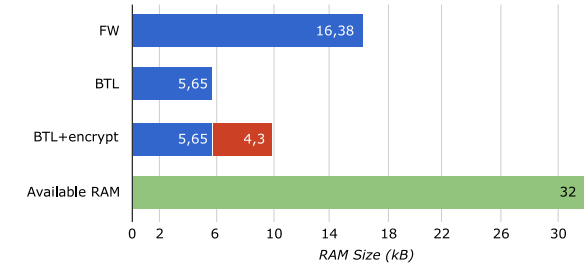

Fig. 8: RAM utilisation based on bootloader type.

Next, the time it takes to flash the device when using different encryption methods is measured. The firmware image file has a size of 433 kB. As can be seen in Fig. 9, it takes almost twice as long to flash the device when using the AES-128 encryption method as when using no encryption. In fact, AES-128 is sufficient method for the IoT devices. Nevertheless, the comparison is also performed for AES-192 and AES-256.

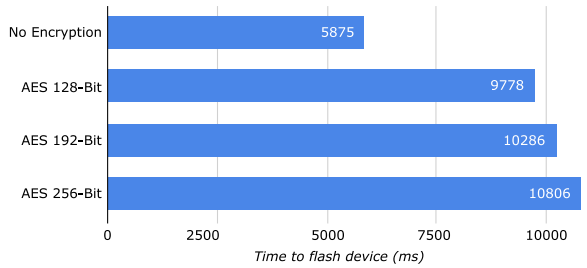

Fig. 9: Flash time by encryption method.

6. Conclusion

The paper describes a software implementation of a secure firmware update solution in an IoT context. Our goal is to find out whether it would be possible to integrate security features such as AES into existing devices. We have concluded that this is possible and the experimental data indicates that there is space to integrate even more security features. There is no need to purchase a more powerful processor in the future.

The measurements indicate that the minimum processor memory requirements are: 49.7 kB of flash memory and 10 kB of RAM. When using AES encryption with a 256-bit key, the time to load the firmware increases twofold. As the devices only need to be updated once in a while, speed is not critical. Having implemented a SW solution only, it may be interesting to try a HW solution in the future and do a cost-benefit analysis. We would also like to conduct experiments with OTP to evaluate the safety claims of the producer. Whether we extend the functionality of the firmware or use dual banked partitioning for firmware backup, an external memory will have to be integrated into the device. We would like to experiment with other cryptographic techniques such as AES with CCM (Counter with CBC-MAC), RSA and integrity verification methods.

Acknowledgment

This paper is supported by the grant no. SGS16/159/OHK3/2T/13 and by the grant in EUREKA Cluster program with EUREKA7592 AutoEPCIS project.

References

[1] KOLAROVSZKI, P. and V. DUBRAVKA. The presentation of production line and warehouse management based on RFID technology through 3D modelling and animation. *Transport and telecommunication*. 2010, vol. 11, iss. 3, pp. 26–36. ISSN 1407-6160.

[2] KEBO, V., P. STASA, F. BENES and J. SVUB. Auto-identification in mining industry. *Inzynieria Mineralna*. 2015, vol. 16, iss. 1, pp. 7–12. ISSN 1640-4920.

[3] VACULIK J., P. KOLAROVSYKI and J. TENGLER. Possibility of RFID in conditions of postal operators. In: *Radio frequency identification from system to applications*. Rijeka: InTech, 2015, pp. 7–12. ISBN 978-953-51-1143-6. DOI: 10.5772/46210.

[4] Gartner Says 6.4 Billion Connected "Things" Will Be in Use in 2016. *Gartner* [online]. 2015. Available at: `http://www.gartner.com/newsroom/id/3165317`.

[5] JURKOVIC, G. and V. SRUK. Remote firmware update for constrained embedded systems. In: *2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. Opatija: IEEE, 2014, pp. 1019–1023. ISBN 978-953-233-077-9. DOI: 10.1109/MIPRO.2014.6859718.

[6] YANG, M. and F. ZHU. The Design of Remote Update System Based on GPRS Technology. In: *2010 International Conference on Management and Service Science*. Wuhan: IEEE, 2010, pp. 1–4. ISBN 978-1-4244-5325-2. DOI: 10.1109/ICMSS.2010.5575699

[7] DALAI, S., B. CHATTERJEE, D. DEY, S. CHAKRAVORTI and K. BHATTACHARYA. Microcontroller based remote updating system using voice channel of cellular network. In: *2015 IEEE Power, Communication and Information Technology Conference (PC-ITC)*. Bhubaneswar: IEEE, 2015, pp. 11–16. ISBN 978-1-4799-7455-9. DOI: 10.1109/PCITC.2015.7438154.

[8] CHOI, B. C., S. H. LEE, J. C. NA and J. H. LEE Secure firmware validation and update for consumer devices in home networking. *IEEE Transactions on Consumer Electronics*. 2016, vol. 62, iss. 1, pp. 39–44. ISSN 0098-3063. DOI: 10.1109/TCE.2016.7448561.

[9] ZAWARE, P. G. and S. V. SHINDE. Wireless monitoring, controlling and firmware upgradation of embedded devices using Wi-Fi. In: *2014 International Conference on Advances in Communication and Computing Technologies (ICA-CACT 2014)*. Mumbai: IEEE, 2014, pp. 1–6. ISBN 978-1-4799-7318-7. DOI: 10.1109/EIC.2015.7230742.

[10] HONG, S. G., N. S. KIM and T. HEO. A smartphone connected software updating framework for IoT devices. In: *2015 International Symposium on Consumer Electronics (ISCE)*. Madrid: IEEE, 2015, pp. 1–2. ISBN 978-1-4673-7365-4. DOI: 10.1109/ISCE.2015.7177805.

[11] THANH, T., T. H. VU, N. V. CUONG and P. N. NAM. A protocol for secure remote update of run-time partially reconfigurable systems based on FPGA. In: *2013 International Conference on Control, Automation and Information Sciences (ICCAIS)*. Nha Trang: IEEE, 2013, pp. 295–299. ISBN 978-1-4799-0572-0. DOI: 10.1109/ICCAIS.2013.6720571.

[12] Atmel AT02333: Safe and Secure Bootloader Implementation for SAM3/4. *Atmel* [online]. 2015. Available at: `http://www.atmel.com`.

[13] Brad Conte Computing, math, and other hobbies: Implementation of AES in C. *Bradconte* [online]. 2006. Available at: `http://bradconte.com/aes_c`.

[14] Un-Attackable (Non-Breakable) of W7500 and W5500. *SOS Electronic* [online]. 2015. Available at: `http://wiznetmuseum.com`.

[15] XueMing. Designing bootloader for Microchip dsPIC33E/PIC24E micro-controller. *Microchip* [online]. 2016. Available at: `https://zavax.wordpress.com`.

About Authors

Lukas KVARDA was born in Jilemnice, Czech Republic in 1986. He earned his M.Sc. degree in Electrical Engineering from the Czech Technical University in Prague in 2013. His research interests include HW and SW design, RFID technology and cryptography.

Pavel HNYK was born in Jilemnice, Czech Republic in 1987. He earned his M.Sc. degree in Electrical Engineering from the Czech Technical University in Prague in 2013. His research interests include HW and SW design, RFID technology and cryptography.

Lukas VOJTECH received M.Sc. and Ph.D. at Telecommunication Engineering at the Czech Technical University in Prague, Czech Republic in 2003 and 2010. He has been actively involved in several national and international projects. He is a leader of RFID laboratory at the Czech Technical University in Prague since 2010. His research interests are hardware prototyping and measurement especially in the field of RFID technology, textile antenna design and localization.

Zdenek LOKAJ received M.Sc. and Ph.D. at Engineering Informatics at Faculty of Transportation Sciences of Czech Technical University in Prague, Czech Republic in 2005 and 2011. In 2015 he achieved Associate Professor grade. His research interests are system design and integration mainly in ITS area, IT security and telecommunications solutions for intelligent transport systems.

Marek NERUDA received the M.Sc. and Ph.D. degree in Electrical Engineering from the Czech Technical University in Prague, Faculty of Electrical Engineering, Czech Republic in 2007 and in 2014, respectively. His research interests include RFID technology and electrically conductive textile materials.

Tomas ZITTA is a Ph.D. student of a study program at Telecommunication Engineering at the Czech Technical University in Prague, Czech Republic. He is focused on mobile application development and security in IoT ecosystem.