ELLIPTIC CURVE CRYPTOGRAPHY FOR CONSTRAINED DEVICES IN INTERNET OF THINGS AND INDUSTRY 4.0

Tadeáš Cvrček
Bachelor Degree Programme, FEEC BUT
E-mail: tadeas.cvrcek@vutbr.cz

Supervised by: Petr Dzurenda
E-mail: dzurenda@feec.vutbr.cz

Abstract: The article summarizes the usability of devices (which are usually equipped with AVR and ARM Cortex-M microcontrollers) with low-power consumption in security solutions for IoT and Industrial 4.0 ecosystems. These devices have limited processing and storage capabilities and they often run on batteries. On the other hand, cryptographic algorithms are usually computationally demanding and, therefore, it is hard to implement them on constrained devices. This work studies the capabilities of current IoT devices to perform elliptic curve cryptographic protocols and the possibilities to design and implement own cryptographic protocols on constrained devices.

Keywords: Arduino, Cryptography, Low Performance Devices, Industry 4.0, Internet of Things

1 INTRODUCTION

Nowadays, there are many devices with low-power consumption which are part of various embedded systems and play an important role in our daily life. High growth of these devices is given mainly by the fast increasing market of IoT (Internet of Things) and Industry 4.0 [1]. The trend of IoT focuses on integration of real world devices into cyberspace. Embedded chips are almost in any device and these devices are connected to the Internet. All these connected (so-called smart devices) make our life easier and more comfortable. However, this connectivity brings a need of higher security, i.e. provide secure communication and access control, as well as economic aspects such as low price and operating costs.

The low power consumption is typical for many of IoT devices. These devices are usually based on ARM-Cortex-M (32-bit) or AVR (8-bit) processor families. Both processor families can be found on current Arduino single-board microcontrollers. For example, AVR processors are used on Arduino UNO and Arduino NANO (ATmega328 runs at 16 MHz) and ARM-Cortex-M powers Arduino DUE (Atmel SAM3X8E runs at 84 MHz).

Asymmetric cryptography is based on modular arithmetic operations (i.e., modular addition, multiplication, inversion and exponentiation) and operations over elliptic curve points (addition of two points and scalar multiplication of elliptic curve point). Furthermore, required modular arithmetic works with big numbers, e.g., RSA uses at least 2048 bits long keys by NIST (National Institute of Standards and Technology) recommendations. On the other hand, elliptic curve cryptography requires smaller keys (but still sufficiently big and secure) which are around 256 bits. However, Arduino natively supports only integer numbers up to 32 bits long.

2 LIBRARIES FOR ARDUINO PLATFORM

Currently, there are many available libraries for Arduino. These libraries extend Arduino capabilities such as network connection, cryptography and security support, and other. Many of these libraries

are freely available thanks to communities of developers that make them open source. First of all, we can use official Arduino repository, which can be accessed directly from Arduino IDE (Integrated Development Environment). In this repository, the libraries and their functionalities are well-tested and surely compatible with Arduino devices. The next great source of Arduino libraries and applications is GitHub, where many developers voluntarily and worldwidely upload their source codes. It is difficult to find such libraries, that support modular arithmetic operations with big numbers (bigger than 512 bits) or elliptic curve operations in general. One option is to use *BigNumber* [2] library supporting modular arithmetic operations. Second option is to use *micro-ecc* [3] library, which implements ECDH (Elliptic Curve Diffie-Hellman) and ECDSA (Elliptic Curve Digital Signature Algorithm) protocols including underlying mathematical operations .

2.1 MODULAR ARITHMETIC OPERATIONS SUPPORT

BigNumber library supports all crucial modular operations: multiplication, addition and exponentiation. We provide benchmark tests of all operations on both Arduino UNO and Arduino DUE devices in order to see differences of their computational complexity based on used architecture, see Table 1 and Table 2 for more details.

Table 1: Benchmark results of modular arithmetic on AVR (BigNumber library)

Operation	16 bits	32 bits	64 bits	128 bits	256 bits	512 bits
Mod. addition	3 ms	4 ms	6 ms	8 ms	12 ms	-
Mod. multiplication	1 ms	5 ms	11 ms	35 ms	119 ms	-
Mod. exponentiation	45 ms	214 ms	1170 ms	7627 ms	-	-

Table 2: Benchmark results of modular arithmetic on ARM Cortex-M (BigNumber library)

Operation	16 bits	32 bits	64 bits	128 bits	256 bits	512 bits
Mod. addition	0 ms	0 ms	0 ms	0 ms	0 ms	1 ms
Mod. multiplication	0 ms	0 ms	0 ms	3 ms	8 ms	26 ms
Mod. exponentiation	5 ms	17 ms	78 ms	492 ms	3646 ms	25712 ms

Comparison of Table 1 and Table 2 shows that ARM Cortex-M offers 10 to 20 times faster operations processing than AVR architecture. Furthermore, these tests also discovered that AVR architecture is able to process only big numbers up to 256 bits length. Based on this fact, Arduino DUE and, therefore, ARM Cortex-M architecture is more suitable for cryptographic protocol implementation and integration to security solutions.

2.2 ELLIPTIC CURVE POINT OPERATIONS SUPPORT

The *micro-ecc* library offers a small and fast ECDH and ECDSA implementation for 8-bit, 32-bit, and 64-bit processors. Unfortunately, this library does not implement interface for modular and elliptic curve external calls. However, the existence of ECDH and ECDSA implementations means that there is already internal supports of modular and elliptic curve arithmetic operations. In our implementation we make these crucial operations available from outside to developers. The library implements by default the following elliptic curves: *secp160r1*, *secp192r1*, *secp224r1*, *secp256r1* a *secp256k1*.

Benchmark tests showed performance differences between AVR (Arduino UNO) and ARM (Arduino DUE), see Figure 1, where ARM architecture is ca. 12 to 20 times faster than AVR in executing ECDH protocol. In case of AVR, the time for ECDH key negotiation is between 5 to 18 seconds depending on the elliptic curve used. This time can be considered as very ineffective in real implementations.

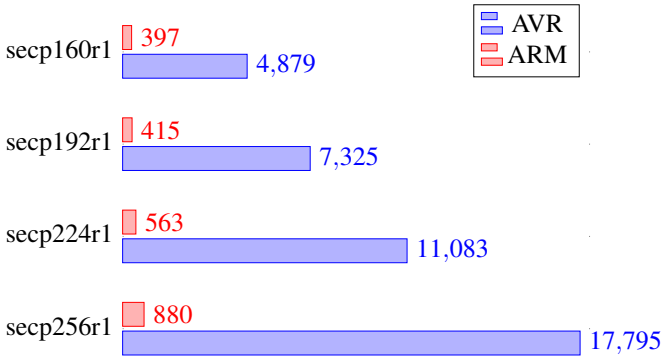

Figure 1: Performance comparison of AVR and ARM Cortex-M for ECDH protocol (Time [ms])

In order to make the *micro-ecc* library usable for implementation of our cryptographic protocols, the library functions needed to be modified. First of all, internal functions for modular arithmetic and point multiplication were made accessible from outside of library. Afterwards, a function for point addition was created.

Benchmark tests of modular arithmetic and elliptic curve point operations were performed only on Arduino DUE, due to its stability and fast processing even with big numbers. The results for all supported elliptic curves are depicted in Table 3. All results are in milliseconds and, therefore, they can be considered as negligible. The most crucial operation is scalar multiplication of elliptic curve point. This operation requires ca. from 60 to 160 ms to process.

Table 3: Benchmark results of micro-ecc on ARM Cortex-M

Operation	secp160r1	secp192r1	secp224r1	secp256r1	secp256k1
Mod. addition	0 ms	0 ms	0 ms	0 ms	0 ms
Mod. multiplication	3 ms	2 ms	2 ms	3 ms	4 ms
Mod. inversion	2 ms	2 ms	3 ms	4 ms	3 ms
Point addition	0 ms	0 ms	0 ms	0 ms	0 ms
Point multiplicaiton	67 ms	79 ms	99 ms	189 ms	161 ms

3 ECDH PROTOCOL IMPLEMENTATION

We use standard *micro-ecc* library, the ECDH protocol was implemented and run between two Arduino NANO devices. Arduino NANO devices have same performance capabilities as Arduino UNO, but are significantly smaller. The final implementation uses I^2C bus and it is controlled by *Wire* library. Communication itself took place over bridges of pins A4↔A4 and A5↔A5. Connection diagram of both devices is depicted in Figure 2. The final implementation was time consuming. In fact, the whole process of key negotiation took from 2 to 5 s depending on the selected elliptic curve.

4 PROOF OF KNOWLEDGE PROTOCOL IMPLEMENTATION

The implementation of proof of knowledge protocol of algebraic message authentication code MAC_{wBB} [4] was run on Arduino DUE. The authentication code MAC_{wBB} and the message m stay hidden during the proving phase. The prover only shows the possession of both these values. These kind of protocols are often used in privacy-enhancing protocols, such as anonymous credentials and group signatures). The protocol includes operations with elliptic curve points and modular arithmetic operations in the field of elliptic curve. Benchmark tests of the implementation are depicted in Figure 3. This protocol implementation is not compatible with AVR architecture.

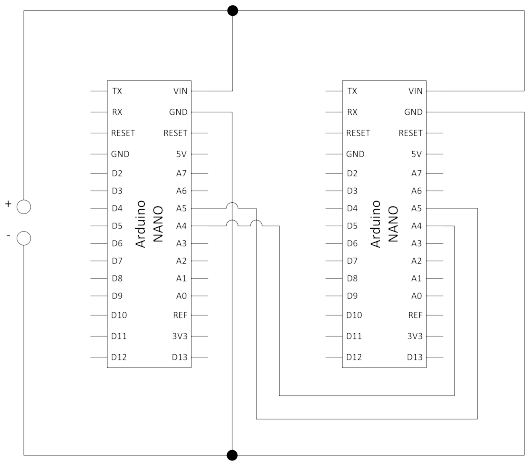

Figure 2: Basic connection of Arduino NANO devices through I^2C bus

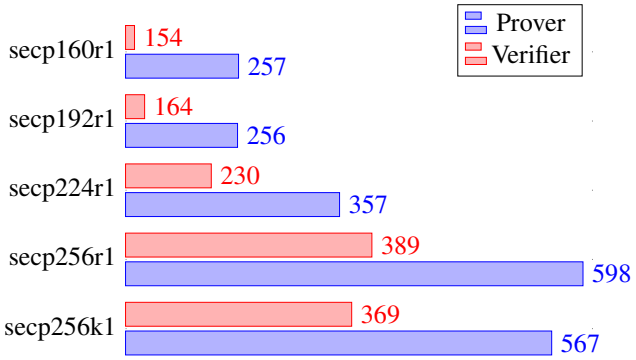

Figure 3: Performance comparison of Proof of Knowledge MAC_{wBB} on Arduino DUE (Time [ms])

5 CONCLUSION

Based on the provided benchmark tests, it is possible to come out with the conclusion that Arduino with ARM Cortex-M processors are the most suitable devices for the implementations of advanced cryptographic protocols with privacy protection features. In fact, the upcoming research follows this direction. Devices powered by AVR architecture are suitable only for less performance demanding projects. Moreover, Arduino DUE offers many possible ways how to develop a final application for a specific use case. In fact, Arduino DUE has sufficient processing power and memory resources, many I/O pins and interfaces for peripheral devices, such as smart cards (i.e., secure HW keystore) and wireless network modules.

REFERENCES

[1] Ministerstvo průmyslu a obchodu České republiky. [online]. [cit. 12. 03. 2020]. URL: https://www.mpo.cz/assets/dokumenty/53723/64358/658713/priloha001.pdf.

[2] Bignumber. URL:https://github.com/nickgammon/BigNumber.

[3] Ken MacKay. micro-ecc. URL:http://kmackay.ca/micro-ecc/.

[4] Camenisch J., Drijvers M., Dzurenda P., Hajny J. (2019) Fast keyed-verification anonymous credentials on standard smart cards. InIFIP International Conference on ICT Systems Security and Privacy Protection 2019 Jun 25 (pp. 286-298). Springer, Cham.