Security as a CoAP resource: an optimized DTLS implementation for the IoT

Angelo Capossele* §, Valerio Cervo*, Gianluca De Cicco* and Chiara Petrioli* §
*Computer Science Department, University of Rome "La Sapienza"
§WSENSE S.r.l.
Rome, Italy
Email: {capossele, cervo, decicco, petrioli}@di.uniroma1.it

***Abstract*—The growing number of applications based on Internet of Things (IoT) technologies is pushing towards standardized protocol stacks for machine-to-machine (M2M) communication and the adoption of standard-based security solutions, such as the Datagram Transport Layer Security (DTLS). Despite the huge diffusion of DTLS, there is a lack of optimized implementations tailored to resource constrained devices. High energy consumption and long delays of current implementations limit their effective usage in real-life deployments. The aim of this paper is to explain how to integrate the DTLS protocol inside the Constrained Application Protocol (CoAP), exploiting Elliptic Curve Cryptography (ECC) optimizations and minimizing ROM occupancy. We have implemented our solution on an off-the-shelf mote platform and evaluated its performance. Results show that our ECC optimizations outperform priors scalar multiplication in state of the art for class 1 mote platforms, and improve network lifetime by a factor of up to 6.5 with respect to a standard-based not optimized implementation.**

I. Introduction

Given the widespread use of Internet of Things (IoT) technologies, Wireless Sensor Networks (WSNs) are likely to interact and exchange information with objects outside their own internal network. This interoperability requirement is pushing to migrate from proprietary protocol stacks to open and standardized solutions, leveraging on the use of IPv6 through the Low power Wireless Personal Area Network (6LoWPAN) and IEEE 802.15.4 protocols [1], [2]. This architecture enables sensor nodes to be directly addressed and seen as producers of information consumed by users on the Internet or by users in the IoT which opportunistically query the WSN. The nature of the WSN channel makes the data vulnerable to being modified, injected and eavesdropped. Therefore, security is often an important requirement, especially in application scenarios such as military, health-care or even home automation [3].

In order to find the right balance among security, energy-efficiency and interoperability, several works have investigated optimizations for Elliptic Curve Cryptography (ECC) [4] to address the design of a standardized security architecture suitable for embedded devices such as WSNs [5]–[7] or even Underwater Acoustic Sensor Networks (UASN) [8]. In [9]–[11], authors describe a solution based on the Transport Layer Security (TLS) protocol [12], which is extended to support ECC [13] to make it viable in a WSN environment. From a security and interoperability perspective, the flexibility provided by the TLS protocol is very appealing for IoT systems, because of its capability to support the negotiation of the cryptographic key and the symmetric cipher suites for message authentication and data encryption.

At the same time, the Constrained Application Protocol (CoAP) [14] is under standardization as an application layer protocol for the IoT. CoAP proposes to use Datagram Transport Layer Security (DTLS) [15], the UDP-based version of TLS, to provide end-to-end security. DTLS was initially designed for traditional networks. Therefore, porting the protocol as it is over resource constrained devices produces a heavyweight solution. DTLS headers are also too long to fit in a single IEEE 802.15.4 maximum transmission unit (MTU). Authors in [16], [17] have presented preliminary ideas on how to overcome these problems, highlighting the need to minimize communication overhead. To achieve this objective Raza et al. [18], [19] have proposed to adopt 6LoWPAN header compression for DTLS. They have linked compressed DTLS with the 6LoWPAN standard, achieving a 62% reduction in the number of additional security bits. Following the same approach, Kothmayr et al. [20], [21] have presented a security scheme based on RSA. Their implementation of DTLS is presented in the context of a system architecture achieving low overhead and high interoperability on a hardware platform suitable for IoT. However, computational overhead of their DTLS handshake introduces a high energy consumption due to the use of RSA-based cryptography. Other works [22], [23] have evaluated the performance of DTLS handshake for resource constrained environments using ECC-based cryptography, whose adoption is also proposed by the CoAP standard [14]. Their results still show high energy consumption.

Our contribution: The aim of this paper is to develop a fully optimized implementation of DTLS for CoAP, by combining existing and novel optimizations, minimizing computation and communication overhead. Our specific contributions are the following.

- We present the architecture of DTLS over CoAP, where security associations are created as CoAP resources, exploiting block wise transfer and message reordering provided by CoAP to minimize communication overhead and ROM occupation.
- We demonstrate the viability of our design by implementing it on an off-the-shelf mote platform. The implementation exploits several state of the art optimizations for

978-1-4673-6432-4/15/$31.00 ©2015 IEEE

DTLS as well as techniques to speed up computation of ECC-based operations.
- We experimentally assess the performance of our implementation in terms of energy consumption and overhead, showing that proposed optimizations can significantly improve overall performance.

The paper is organized as follows. In the next section we provide necessary background on CoAP and DTLS. Section III describes the architecture of our solution, detailing technical optimizations in Section IV. In Section V we evaluate performance of our DTLS implementation. Finally, Section VI concludes the paper.

II. CoAP and DTLS

A. CoAP

The IETF Constrained Application Protocol (CoAP) [14] is an application layer protocol tailored to resource constrained devices and M2M applications. It allows communication over the Internet among IoT objects that support UDP and 6lowPAN, achieving low overhead and supporting multicast. CoAP is an optimized implementation of the RESTful[1] specification [24], where a well-known URI specifies an entry point for requesting the resources hosted by a server. Similarly to the HTTP protocol, a typical URI can be: "coap://ipv6host:port/resource" CoAP architecture is divided into two layers: the lower *message layer* and the upper *request/response layer*. The *message layer* provides reliability and sequencing by means of a stop and wait protocol using the following types of messages: *confirmable* which requires an *acknowledgment* message as response, *non-confirmable* which does not require a response, and *reset* which is used in case a *confirmable* message cannot be processed. The *request/response layer* manages the mapping between requests and responses and their semantic. This layer offers basic request methods to provide a RESTful architecture: *GET*, *PUT*, *POST* and *DELETE*. The *GET* method retrieves information regarding the resource specified by the URI. *PUT* and *POST* methods both create or modify a target resource, with the difference that the former has the idempotent property. The *DELETE* method requests to delete a specified resource. Each request is associated to a response, which is identified by a code field in the CoAP header[2]. Figure 1 shows a simple CoAP interaction between a client and a temperature sensor acting as a server.

B. DTLS

The Datagram Transport Layer Security (DTLS) [15], is the UDP-based version of TLS, designed to provide end-to-end security association between pairs. It permits to flexibly negotiate security services and cryptographic mechanisms, selecting a specific cipher suite. An example of cipher suite is: TLS_ECDH_ECDSA_WITH_AES_128_CCM_8, which describes

[1]Representational state transfer (REST) is an abstraction of the architecture of the World Wide Web.

[2]See RFC 7252 for further details [14]

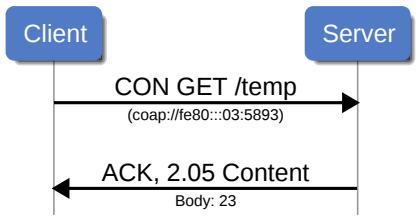

Fig. 1. Simple CoAP interaction.

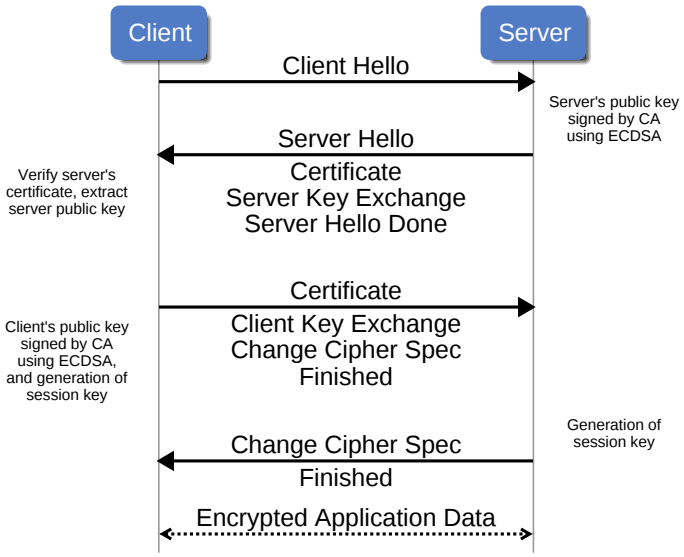

Fig. 2. DTLS key agreement with Fixed Diffie Hellman over Elliptic Curves.

the *fixed* Elliptic Curve Diffie-Hellman (ECDH) key agreement where a certificate contains the ECDH-capable public key signed by the Certification Authority with the Elliptic Curve Digital Signature Algorithm (ECDSA) [4]. Figure 2 shows an instance of the DTLS handshake protocol.

The client starts the DTLS handshake sending to the server a Client Hello message, containing the supported cipher suites (e.g., TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8). The server commits the decision of which cryptographic algorithms to use by sending as response a Server Hello message. The handshake then proceeds with an exchange of information, such as Certificates and Key Exchanges, required to establish a common secret, from which per-session keys are derived. A 1B long message, ChangeCipherSpec, then informs the other party to switch to authenticated/encrypted mode using the negotiated algorithms. Finally, the negotiation process ends verifying the previous messages exchanged with the Finished message, which contains a message authentication code (MAC) computed based on all the previous sent and received handshake messages as seen by each peer. The subsequent exchanged data are encrypted under the specified symmetric encryption algorithm (e.g., AES128 [25]) using the session key.

CoAP proposes to use DTLS protocol to provide end-to-end security, requiring the support of *NoSec* mode when DTLS is disabled, and *RawPublicKey*

mode with TLS_ECDH_ECDSA_WITH_AES_128_CCM_8 when an asymmetric key pair is present. Additonal not-mandatory options are *PreSharedKey* mode with TLS_PSK_WITH_AES_128_CCM_8 if a pre-shared key is available, or the cipher suite TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA if both *PreSharedKey* and *RawPublicKey* methods are supported. A secured resource is thus requested using: "coaps://", otherwise "coap://" if *NoSec* mode is employed.

III. DTLS as a CoAP resource

Since DTLS was initially designed to protect web application communication, an implementation *as is* results in heavy overhead in IoT scenarios, where sensor platforms have limited computational capabilities and internal (RAM/ROM) memory. Moreover, the complexity of DTLS implementation is increased by the fact that using it on top of the UDP protocol, requires to provide mechanisms to guarantee reliability and ordering of messages. To design a version of DTLS tailored to resource constrained devices, it is important to minimize both code size and amount of messages exchanged, resulting in an optimized handshake protocol. As also suggested in [17], we exploit CoAP capabilities to provide connection oriented communication offered by its *message layer*. More specifically, *Confirmable* messages require an *Acknowledgement* message as response, thus providing a reliable transmission. In addition, fragmentation can be performed relying on the block-wise transfer feature defined by CoAP, developed to support transmission of large payloads. The combination of both mechanisms allows to guarantee that DTLS is compliant to the standard but lighter. We develop RESTful DTLS connection as a CoAP resource, which is created when a new secure session is requested. This on one side allows large reuse of CoAP functionalities and code, and on the other side provides CoAP with the abilities to optimize use of resources, including what needed for security associations.

Denial of service (DoS) attacks are a critical aspect which needs to be addressed. Resources on the server can be consumed by an attacker initiating a series of handshake requests, resulting in possible expensive operations or flooding of large messages, such as certificates. In order to mitigate this attack, we used the stateless cookie technique presented in [26], where clients are forced to retransmit the *Client Hello* with the appended cookie. The server, based on the validation of the cookie, can continue the handshake.

Figure 3 shows how the handshake works using CoAP, where communication reliability is provided by *CON* and *ACK* messages which contain the DTLS handshake messages as payload. An IoT object acting as a client can request a secure connection to a server object using the URI "coaps://ipv6host:port/dtls" with the *PUT* method containing the *Client Hello* message as payload. As a result, a new DTLS session is created on the server and it can be updated using the *POST* method. The followings *HelloVerifyRequest* and *Client Hello* sent respectively by the server and the client, mitigate the DoS attack discussed above. Then, according to

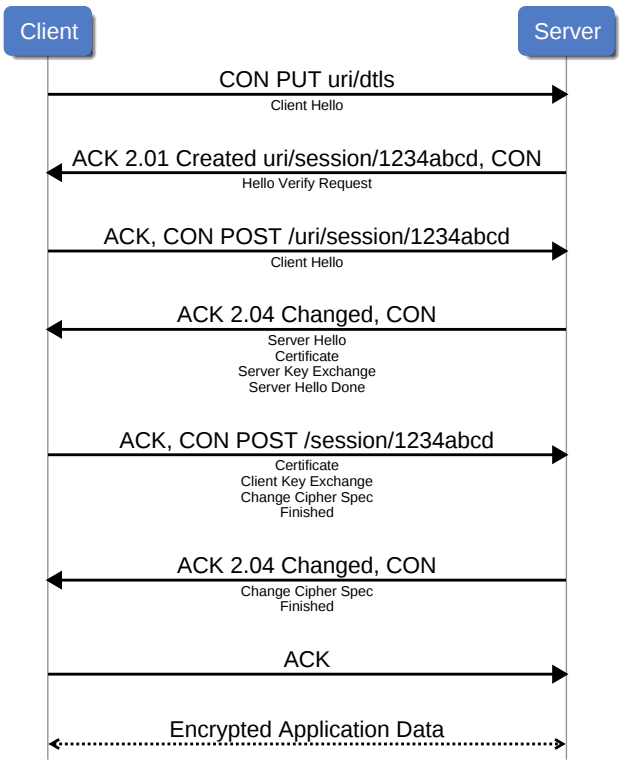

Fig. 3. DTLS key agreement over CoAP with Raw Public Key.

the negotiated cipher suite, they exchange information needed to establish a common secret and derive a session key. Note that, based on the cipher suite, messages of this phase can be quite large (e.g., certificate). Our DTLS implementation leaves all the fragmentation tasks to CoAP, through its efficient block-wise transfer. Finally, after both client and server have received the *Finished* message, handshake is concluded and the secure session is established.

Since the resources available on a sensor node are clearly limited, devices can close a DTLS connection, thus freeing RAM space. However, considering that DTLS handshakes are by far the most expensive task of the DTLS protocol, frequently closing and re-establishing connections is very inefficient. Therefore, sensor nodes should keep the security association up for as long as possible. For this reason, if the memory available on a sensor node is unable to store parameters for all the security associations, we adopt a caching strategy to store on the flash memory the set of security associations that are frequently used.

IV. Implementation and Optimizations

To demonstrate the viability of our solution, we have implemented it on the MagoNode [27], a mote platform that has been designed by our spin-off Wsense.[3] The MagoNode features an 8bit ultra low power 16MHz microcontroller, the Atmel's Atmega128RFA1 (RFA1), with an integrated low-power transceiver 802.15.4. Thanks to the 2.4Ghz RF front-

[3]http://www.wsense.it

end, the MagoNode platform is able to communicate over long-distances still keeping very high energy efficiency. It is equipped with 16KB of RAM and 128KB of ROM, which is enough to store TinyOS stack which includes 6LoWPAN, RPL, UDP, CoAP and our DTLS implementation. We used TinyOS 2.x [28] as the operating system and we developed our solution in nesC.

From a performance perspective, implementing a cryptographic library on resource constrained devices is very challenging. Motes are usually equipped with simple and cheap microcontrollers, few KB dozens of ROM and just 10-16KB of RAM. Performing heavy cryptographic operations typically requires long execution times, impairing communication delay and energy consumption performance. It is crucial to implement efficiently these operations, optimizing them at the lowest level, in order to achieve good performance while providing an acceptable security level. The following subsections describe how we have optimized implementation of basic operations on which many security protocols such as ECDH, ECDSA rely upon.

A. Modular Arithmetic on Large Integer

Large integers on devices with 8bit registers are implemented using arrays. When operations such as multiplication and squaring are used, which require a large amount of memory accesses to store results, an efficient use of registers is needed. We developed assembly code routines based on [29], specifically optimized for the MagoNode platform, which allow an improved use of registers, thus reducing the number of memory operations.

B. ECC

We developed our ECC library based on a combination of TinyECC[4] and Relic[5] libraries, implementing the elliptic curve as a MNT curve, which can be described in the simplified Weierstrass form as

$$E(\mathbb{F}_p) : y^2 = x^3 + ax + b. \quad (1)$$

The elliptic curve E is defined over a prime field $\mathbb{F}_p$ where $p = 2^{160} - 2^{31} - 1$ according to SECG [30] recommendations. This curve provides a security level of 80 bits. In addition, our group size p is a pseudo Mersenne prime, therefore we can speed up modular multiplication and squaring by adopting curve-specific optimizations. Moreover, the elliptic curve E can be converted from affine coordinates to Jacobian ones as follows:

$$E(\mathbb{F}_p) : Y^2 = X^3 + aXZ^4 + bZ^6. \quad (2)$$

where $X = xZ^2$, $Y = yZ^3$. Adding a third element to represent a point (X, Y, Z), allows to separately calculate the numerator and the denominator during computational costly operations such as the modular inversion. As a result, we can further reduce the execution time to perform such operations.

[4]http://discovery.csc.ncsu.edu/software/TinyECC/

[5]http://code.google.com/p/relic-toolkit

Let $G \in \mathbb{G}_q$ be a generator of a cyclic subgroup of order q. The elliptic curve scalar multiplication $P = kG$ is defined as the addition of the point G along the curve E repeated k times. The problem of finding k, given P and G is called the elliptic curve discrete logarithm problem (ECDLP), whose hardness is the fundamental assumption of the security of various cryptographic protocols, such as the Elliptic Curve Diffie-Hellman (ECDH) and the Elliptic Curve Digital Signature Algorithm (ECDSA). Scalar multiplication is often the most expensive operation in EC based cryptography, therefore optimizing it can drastically improve overall performance of the specific cryptographic protocol. In order to speed up the computation of scalar multiplication, we used the technique proposed in [6], hereafter referred to as *IBPV*. Roughly speaking, it randomly generates a number n of k_i elements of order q, precomputes $P_i = Gk_i$ for all $i \in n$ and stores in a table all the pairs (k_i, P_i). When we need to perform a scalar multiplication such as $R = Gr$, where r is a random element of order q, *IBPV* randomly selects l out of the n precomputed pairs, computes $r = \sum k_i$ and $R = \sum P_i$ for all $i \in l$ terms respectively, obtaining the corresponding $R = Gr$. We exploit *IBPV* to improve performance of ECDSA signature, moreover we extend it to the ECDH protocol. We also used the *Shamir trick* [4] which allows to perform the sum of two scalar multiplications faster than performing two independent scalar multiplications, thus reducing the execution time of the ECDSA signature verification.

V. Performance Evaluation

To evaluate the impact of our optimized implementation of DTLS, we assess the computational overhead of its most frequently used operations, defined in terms of the time needed to perform such operations. We also considered the energy impact of performing our DTLS handshake with different cipher suites. Furthermore, to highlight benefits of designing the DTLS as a CoAP resource (hereafter referred as DTLS over CoAP), we compared our solution with the standard implementation of DTLS in terms of RAM and ROM occupancy. Since there was not an available implementation of DTLS for TinyOS, we have also implemented the standard version, exploiting some parts of TinyDTLS, a Contiki [31] version of DTLS. Table I shows RAM and ROM occupancy of CoAP and Blip protocols, which are the stack implementation composed of the 802.15.4 MAC layer, RPL (RFC 6550) as routing protocol, 6LoWPAN, UDP and CoAP. The whole stack occupancy is respectively about 41% and 40% of the RAM and ROM available on the MagoNode platform. Table I shows also the comparison of the amount of resources required by the standard implementation of DTLS with respect to DTLS over CoAP. Results show that leaving reliability and fragmentation tasks to CoAP, and implementig DTLS as a CoAP resource, ROM occupancy of DTLS can be reduced by almost 23% with respect to the standard implementation of DTLS which is also more expensive in terms of RAM resources, as it requires an extra 236B overhead.

Table II shows the computational overhead and energy cost of the EC scalar multiplication over the MagoNode mote, which we recall to be the most expensive operation performed by cryptographic protocols during the DTLS handshake. We separately display performance results obtained by enabling the different optimizations as this allows to understand the impact on performance of each introduced optimization. Performance are measured in terms of overhead and ROM consumption. *Base* identifies the implementation without optimizations. *Assembly* introduces low-level routines for modular arithmetic on large integer, *curv. opt.* adds to *base* Mersenne specific elliptic curve optimization, *proj. coord.* enhances *base* with the use of Jacobian coordinates to represent a point on the curve. *All* refers to the case where all optimizations, except *IBPV* are enabled. *All + IBPV* enables also the *IBPV* generator. The *base* version is basically composed of the ECC library providing modular arithmetic for large integers and for points on elliptic curves. Its ROM occupation is about 7KB, and we used it as the benchmark to show the extra overhead introduced by each optimization. Enabling *all* as optimization level, performance of scalar multiplication improves by a factor of almost 14 with respect to *base*, at the cost of an extra 4.9KB of ROM. This is an additional 70% of the ROM with respect to what needed by the ECC library. By adding another 13.6% of code, we enable *all + IBPV* which allows to reduce the execution time and the corresponding energy consumption by a factor of 100, the most performing solution in state of the art for class 1 mote platforms [6].

Table III explains the impact of our optimizations from a cryptographic protocols perspective, showing the execution time and the energy consumption needed to perform a digital signature and a signature verification using ECDSA and an instance of the ECDH protocol. Results compare *base*, *all* and *all + IBPV* settings. Performing a signature, using *all* improves performance by a factor of 13 while *all + IBPV* improves performance by a factor of almost 91. For the signature verification, enabling IBPV does not introduce any improvement with respect to *all* setting because the operations involved do not require any scalar multiplication in the form Gr where r is a large integer randomly chosen. Thus the main optimization involved is the *Shamir trick*, which allows to reduce execution time and energy cost by a factor of 25. Finally, ECDH is improved by a factor of 13.8 and 24 respectively with *all* and *all + IBPV* setting.

In Figure 4 we show the energy impact, in terms of mote lifetime, of establishing several DTLS sessions per hour, comparing our proposed solution with the standard implementation of DTLS (e.g. not optimized). For this experiment, we used a mote acting as a server and several motes acting as a client. Each mote (MagoNode platform) was equipped with two alkaline AA 1.5 V batteries. Since the most energy costly operations during an handshake are ECC operations, we compare the energy consumption adopting two different cipher suites based on ECC: TLS_ECDH_ECDSA_WITH_AES_128_CCM_8 and TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA. The former is based on the *fixed* ECDH key agreement where a certificate contains the ECDH-capable public key signed by the Certification Authority with ECDSA. Therefore the main operations involved are scalar multiplication and signature verification. In the latter, public keys are ephemerals, so peers authentication is not provided. Cipher suites are implemented both in their standard version (e.g. not optimized) and in their optimized version. Results show that significant lifetime improvements are obtained when using our optimizations. Lifetime improves by a factor ranging from 2.8 to 6.5 using TLS_ECDH_ECDSA_WITH_AES_128_CCM_8 and by a factor ranging from 2 to 4.4 using TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA .

Protocol	ROM	RAM
CoAP + Blip	51410 B	6653 B
standard DTLS	10983 B	7380 B
DTLS over CoAP	8936 B	7144 B

TABLE I
ROM & RAM OCCUPANCY OF OUR DTLS OVER COAP IMPLEMENTATION

Scalar Multiplication			
Optimization	Time	Energy	ROM
base	13487 ms	190.17 mJ	/
assembly	13057 ms	184.1 mJ	2318 B
curve opt.	12591 ms	177.53 mJ	578 B
proj. coord.	3896 ms	54.93 mJ	1994 B
all	976 ms	13.76 mJ	4890 B
all + IBPV	135 ms	1.9 mJ	5852 B

TABLE II
COMPUTATIONAL OVERHEAD, ENERGY CONSUMPTION AND ROM OCCUPANCY ON MAGONODE PLATFORM.

VI. CONCLUSION

We have presented the architecture of our implementation of DTLS over CoAP, where security associations are created as CoAP resources, exploiting reliability and the block wise transfer provided by CoAP. Our implementation exploits specific optimizations, from a low level (e.g., assembly routines) up to a protocol level, thus minimizing computation overhead and ROM occupation. The implementation and feasibility of the proposed solution has been assessed through experiments on our MagoNode platform, showing that our optimizations allows to achieve the most performing scalar multiplication in state of the art for class 1 mote platforms. Moreover, we have detailed each implemented optimization showing advantages and disadvantages of enabling them, in terms of computational overhead and ROM occupation. Experimental results show

	base (ms/mJ)	all (ms/mJ)	all + IBPV (ms/mJ)
ECDSA sign	13635 / 192.25	1045 / 14.73	150 / 2.11
ECDSA verify	27476 / 387.41	1076 / 15.17	1076 / 15.17
ECDH	26974 / 380.33	1952 / 27.52	1111 / 15,67

TABLE III
COMPUTATIONAL OVERHEAD AND ENERGY CONSUMPTION OF ECDSA AND ECDH PROTOCOLS FOR DIFFERENT OPTIMIZATIONS ON MAGONODE PLATFORM.

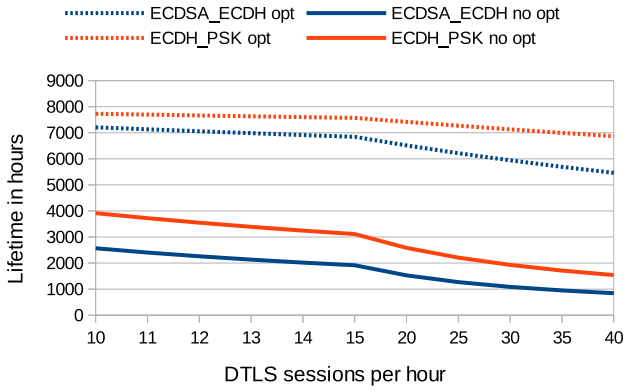

Fig. 4. Lifetime of MagoNode platform depending on the number of DTLS opened sessions per hour.

that network lifetime improves by a factor of up to 6.5 using our solution with respect to a standard-based not optimized implementation.

Acknowledgment

This paper has been partially supported by the PRIN project TENACE.

References

[1] N. Kushalnagar, G. Montenegro, and C. Schumacher, "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals," *Network Working Group RFC 4919*, August 2007. [Online]. Available: http://tools.ietf.org/html/rfc4919

[2] "Approved Draft Amendment to IEEE Standard for Information technology-Telecommunications and information exchange between systems-PART 15.4:Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs): Amendment to add alternate PHY (Amendment of IEEE Std 802.15.4)," *IEEE Approved Std P802.15.4a/D7, Jan 2007*, pp. –, 2007.

[3] S. Gerdes and O. Bergmann, "Security Requirements for Managing Smart Objects in Home Automation," in *Mobile Networks and Management*. Springer Berlin Heidelberg, 2013, vol. 58, pp. 231–243.

[4] D. Hankerson, A. J. Menezes, and S. Vanstone, *Guide to Elliptic Curve Cryptography*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2003.

[5] G. Bianchi, A. T. Capossele, C. Petrioli, and D. Spenza, "AGREE: exploiting energy harvesting to support data-centric access control in WSNs," *Ad hoc networks*, vol. 11, no. 8, pp. 2625–2636, 2013.

[6] G. Ateniese, G. Bianchi, A. T. Capossele, and C. Petrioli, "Low-cost Standard Signatures in Wireless Sensor Networks: A Case for Reviving Pre-computation Techniques?" in *Proceedings of the 20th Annual Network & Distributed System Security Symposium, NDSS'13*, San Diego, CA, USA, 2013.

[7] L. Marin, A. Jara, and A. S. Gomez, "Shifting primes: Optimizing elliptic curve cryptography for 16-bit devices without hardware multiplier," *Mathematical and Computer Modelling*, vol. 58, no. 56, pp. 1155 – 1174, 2013.

[8] G. Ateniese, A. T. Capossele, P. Gjanci, C. Petrioli, and D. Spaccini, "SecFUN: Security Framework for Underwater acoustic sensor Networks," in *Proceedings of OCEANS - Genova, 2015 MTS/IEEE*, May 2015, pp. 1–9.

[9] V. Gupta, M. Wurm, Y. Zhu, M. Millard, S. Fung, N. Gura, H. Eberle, and S. C. Shantz, "Sizzle: a standards-based end-to-end security architecture for the embedded internet," Sun Microsystems, Inc., Technical Reports, SERIES 13103, Tech. Rep., 2005.

[10] S. Fouladgar, B. Mainaud, K. Masmoudi, and H. Afifi, "Tiny 3-TLS: A Trust Delegation Protocol for Wireless Sensor Networks," *Springer LNCS, Security and Privacy in Ad-Hoc and Sensor Networks*, pp. 32–42, March 2007.

[11] G. Bianchi, A. T. Capossele, A. Mei, and C. Petrioli, "Flexible Key Exchange Negotiation for Wireless Sensor Networks," in *Proceedings of the fifth ACM international workshop on Wireless network testbeds, experimental evaluation and characterization*, ser. WiNTECH '10. Chicago, Illinois, USA: ACM, 2010, pp. 55–62.

[12] T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) Protocol, Version 1.2," *IETF RFC*, vol. 5246, August 2008.

[13] S. Blake-Wilson, N. Bolyard, V. Gupta, C. Hawk, and B. Moeller, "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)," *IETF RFC*, vol. 4492, May 2006.

[14] Z. Shelby, K. Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP)," *IETF RFC 7252*, June 2014. [Online]. Available: http://tools.ietf.org/html/rfc7252

[15] E. Rescorla and N. Modadugu, "Datagram Transport Layer Security," *IETF RFC*, vol. 4347, April 2006.

[16] K. Hartke, "Practical Issues with Datagram Transport Layer Security in Constrained Environments draft-hartke-dice-practical-issues-01," *DICE Internet-Draft*, April 2014. [Online]. Available: http://tools.ietf.org/html/draft-hartke-dice-practical-issues-01

[17] S. Keoh, S. Kumar, and Z. Shelby, "Profiling of DTLS for CoAP-based IoT Applications draft-keoh-dice-dtls-profile-iot-00," *Internet-Draft*, November 2013. [Online]. Available: http://tools.ietf.org/html/draft-keoh-dice-dtls-profile-iot-00

[18] S. Raza, D. Trabalza, and T. Voigt, "6LoWPAN Compressed DTLS for CoAP," in *Proceedings of the IEEE DCOSS 2012*. IEEE, pp. 287–289.

[19] S. Raza, H. Shafagh, K. Hewage, R. Hummen, and T. Voigt, "Lithe: Lightweight secure CoAP for the Internet of Things," *Sensors Journal, IEEE*, vol. 13, no. 10, pp. 3711–3720, Oct 2013.

[20] T. Kothmayr, "A Security Architecture for Wireless Sensor Networks based on DTLS," Master's thesis, Universitat Augsburg, December 2011.

[21] T. Kothmayr, C. Schmitt, W. Hu, M. Brnig, and G. Carle, "DTLS based security and two-way authentication for the Internet of Things," *Elsevier, Ad Hoc Networks*, vol. 11, no. 8, pp. 2710 – 2723, 2013.

[22] J. Granjal, E. Monteiro, and J. Silva, "On the effectiveness of end-to-end security for internet-integrated sensing applications," in *Green Computing and Communications (GreenCom), 2012 IEEE International Conference on*, Nov 2012, pp. 87–93.

[23] J. Granjal, E. Monteiro, and J. Sa Silva, "End-to-end transport-layer security for internet-integrated sensing applications with mutual and delegated ECC public-key authentication," in *IFIP Networking Conference, 2013*, May 2013, pp. 1–9.

[24] R. T. Fielding and R. N. Taylor, "Principled design of the modern web architecture," *ACM Trans. Internet Technol.*, vol. 2, no. 2, pp. 115–150, May 2002.

[25] N. FIPS, "197: Announcing the Advanced Encryption Standard (AES)," *Information Technology Laboratory, National Institute of Standards and Technology*, vol. 5 (4), November 2001.

[26] P. Karn and W. Simpson, "Photuris: Session-Key Management Protocol," *Network Working Group RFC 2522*, March 1999. [Online]. Available: http://tools.ietf.org/html/rfc2522

[27] U. M. Colesanti, A. Lo Russo, M. Paoli, C. Petrioli, and A. Vitaletti, "Introducing the magonode platform," in *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, ser. SenSys '13. New York, NY, USA: ACM, 2013, pp. 79:1–79:2.

[28] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler, "Tinyos: An operating system for sensor networks," in *Ambient Intelligence*, W. Weber, J. Rabaey, and E. Aarts, Eds. Springer Berlin Heidelberg, 2005, pp. 115–148.

[29] N. Gura, A. Patel, A. W, H. Eberle, and S. C. Shantz, "Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs," in *Proceedings of the 6th International Workshop on Cryptographic Hardware and Embedded Systems*, Boston, Massachusetts, USA, August 2004, pp. 119–132.

[30] SECG, "SEC 2: Recommended Elliptic Curve Domain Parameters version 2.0."

[31] A. Dunkels, B. Grnvall, and T. Voigt, "Contiki - a lightweight and flexible operating system for tiny networked sensors," in *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*, Nov 2004, pp. 455–462.