Decoupling security from applications in CoAP-based IoT devices

Jorge David de Hoz, *Member, IEEE*, Jose Saldana, *Member, IEEE*,
Julián Fernández-Navajas, José Ruiz-Mas

***Abstract*— The complex and ever changing Internet of Things (IoT) domain could benefit from standardization and a higher degree of autonomy between different layers: standard approaches defining the relationship between security communication software functionalities, hardware and applications would allow a more efficient, flexible and secure communication. To this end, techniques in which the security of IoT devices is decoupled from the applications they run, can provide significant benefits and enable the development of new standardization strategies. This paper presents a study of the benefits provided by IoTsafe, a security decoupling approach, when used in combination with the Constrained Application Protocol (CoAP). Whereas previous work relied on HTTP/HTTP2 protocols, the present paper is focused on the analysis of the feasibility of IoTsafe in more constrained devices in channels with high interference levels. The benefits of this technique are illustrated by means of a battery of tests to evaluate the impact of this scheme. The results show no performance penalty (taking CoAP with security as a baseline) in lossless channels, even when an overhead increment of 38% is borne. Furthermore, in lossier channels, a transfer time reduction of 36% is achieved, a figure that increases significantly if traffic compression is enabled.**

***Index Terms*—IoT, CoAP, SSH, DTLS, Security decoupling**

I. Introduction

The continuous increase in the number of the IoT devices is a well-known phenomenon. Some predictions forecast a total amount of connected devices around 500 billion by the end of 2030 [1]. As a counterpart, the concerns about the security threats that may affect these devices are growing. Extensive literature has already defined different taxonomies of potential threats for these devices, and has also suggested some possible strategies to overcome them and limit their scope [2] [3] [4] [5].

IoT security maintenance is usually addressed following an "ad-hoc" reactive approach, where temporary measures (patches) are provided whenever a vulnerability is discovered by an attack or just by public disclosure. However, the lack of incentives for the final user to deploy these patches may render this security paradigm insufficient [6]. To overcome this problem, a more proactive methodology would need to include an improved approach where security-by-design is seriously considered [7].

Each IoT market is defined by specific needs addressed by technological solutions that usually cover all concerning aspects and layers of the IoT project in a vertical way: from devices to services. Thus, IoT vertical markets' demands usually result in IoT vertical solutions. These technologies are devised only with the particular market problems in mind, and the absence of globally accepted standards at each layer hinders future interoperability between verticals. Advanced security technologies will further isolate the different solutions, requiring more complex middleware to keep connectivity but preventing new business opportunities from arising. For the sake of productivity, and also to benefit from new third parties' added value, vertical solutions should be encouraged to increase their interoperability and potential collaboration [8].

Regulators are also addressing this problem. As an example, the European Union is concerned about all these cybersecurity issues and truly committed to its future regulation. Ongoing work in this area suggests that, in some cases, all IoT security efforts will have to be externally certified in accordance with the Cybersecurity Act [9]. These security efforts affect any piece of software, hardware, a device or set of devices, a technique, or a combination of any of the previous, according to the definition of the Target of Evaluation (TOE) by the Common Criteria for IT Security Evaluation (ISO/IEC 15408)[1]. In order to achieve an Evaluation Assurance Level 7 (EAL7), the current *complexity of the design must be minimized* and also *a "white box" testing and complete independent confirmation of developer test results are required* [10].

This suggests that a simplification of the certification process is feasible if more independence between the parts of the system is achieved. Security should be guaranteed and managed by well-defined *items* (hardware, software or both), which should have to be certified (for security purposes) according to their use, as it similarly happens with i.e., Payment Card Industry (PCI)[2] certifications in points of sale, cashiers, etc. Those devices might be complex but, as "*payment security decoupling*" is successfully implemented, only the elements that are involved in payment procedures have to be certified by PCI.

Thus, solutions able to facilitate the future certification processes of IoT devices are needed. This problem is being dealt with in different works usually through a middleware approach [11] offering hardware abstraction [12] to increase interoperability and to improve security [13], but the Application

Jorge David de Hoz, Jose Saldana, Julián Fernández-Navajas and José Ruiz-Mas are with the I3A Aragon Institute of Engineering Research, University of Zaragoza, Ada Byron Building, 50018 Zaragoza (e-mail: dhoz@unizar.es; jsaldana@unizar.es; navajas@unizar.es; jruiz@unizar.es).

[1] Common Criteria for IT Security Evaluation, https://www.commoncriteriaportal.org/files/ccfiles/CCPART1V3.1R5.pdf

[2] Payment Card Industry certifications https://www.pcisecuritystandards.org/

Programming Interfaces (APIs) required to be used by the IoT application are wont to prevent a full decoupling from succeeding, as highlighted in [14]. In that paper, and also in [15], a system called IoTsafe was presented, where the security was decoupled from the applications using plain sockets, Secure Socket Shell (SSH) protocol and Linux kernel features. Such an approach does not guarantee any certification compliance by this alone, but simplifies the way to address security and certification issues separating security from applications [15]. However, its scope was limited because it was only implemented and tested with long-standing protocols such as HTTP. In contrast, more recent protocols specifically designed for IoT environments as i.e., CoAP, were not studied.

In the present paper, we perform an analysis of the security decoupling feasibility in IoT environments using CoAP. An overall study of its advantages is conducted and it is also discussed how CoAP environments could easily benefit from security decoupling features. Taking into account the resource limitations of these environments, the performance impact is also studied, by means of empirical measurements of the penalty cost the security decoupling approach could have in terms of transfer time, which by extension affects the energy consumption and hardware resources required [15]. The tests will primarily evaluate the feasibility of the security decoupling strategy for CoAP applications and its performance in lossy environments due to existing interferences in the channel used.

The remainder of the article is organized as follows: in the next section we summarize the security decoupling approaches existing today, with a particular focus in communications. In this section are introduced as well some capital features of constrained devices and protocols, and some ongoing efforts which would enable such security decoupling strategy in these devices. Section III details the Test Scenarios, the Results are presented in section IV and the paper ends with the Conclusions.

II. Related work

A. Security decoupling: a "Divide and rule" policy

The security concerns, the verticals' interoperability and the regulation efforts previously introduced represent competing interests from different parties, all together intertwined into a Gordian knot that stalls most of the initiatives towards any direction. This has motivated a differentiated handling of the security in a special way, apart from the rest of functionalities and traditional features IoT device's applications may offer.

Modern processors and microcontrollers provide special functionalities that can help to achieve this objective. The tasks related to security management, integrity and trust between applications in a device can be addressed with specific security hardware entities, conforming a Trusted Execution Environment (TEE) [16]. This can be understood as a security decoupling of sorts, as the applications are relying main security responsibilities to these entities, so as to gain a proper trust. Applications running under this paradigm are considered under a TEE [17].

[3] Global Platform, The standard for secure digital services and devices, https://globalplatform.org/

Nevertheless, this approach makes the final product highly dependent on the chosen technical scheme (the TEE solution and hardware architecture). Thus, some initiatives have arisen such as Globalplatform[3], gathering about 90 organizations worldwide to devise the basis of a set of standards that any solution should follow so as to guarantee interoperability. Such standards define multiple interfaces according to each technical area. Network communications also have their own, defined by the TEE Sockets API [18].

This API enables Trusted Applications (TAs) to establish and use their communications through a new kind of network socket, known as the *iSocket.* This entity acts as a wrapper of a regular socket for the TA (Fig. 1). Rich Execution Environment Applications (GlobalPlatform uses the term "REE" to talk about applications in which the user interacts with the OS) have to rely on the TEE Client API in order to secure their communications through a TA and thus, this scheme is indeed following a security decoupling approach. The *iSocket* can be configured to establish a secure communication using Transport Layer Security (TLS) or Datagram Transport Layer Security (DTLS) protocols. In the case TLS/DTLS became outdated, a new TA should be devised and upgraded to support the new TLS version. This might not be straightforward, but it would imply a much easier procedure than upgrading the whole REE Application for every kind of device.

This particular item (the TA) is supposed to become common, under heavy development and independent from REE Applications. However, if this upgrade (of the TA) required a new version of the TEE Client API, the REE Client Application should have to be adapted as well, and its binary code replaced, though this situation should become rather anecdotal (only major upgrades should require API changes).

As a summary, it can be said that an IoT solution which is to follow this scheme to the letter would benefit from certifiable security, although this may imply more complexity and a certain loss of interoperability. Regarding networking issues in particular, when the application and no TEE-compliant sources have to communicate, then TLS/DTLS must be included in the REE application, preventing a security decoupling approach from being followed in practice. Besides, using TEE network security features also requires the application to be *specifically* designed for such a communication API, forcing already existing applications to be redesigned to get adapted to this requirement.

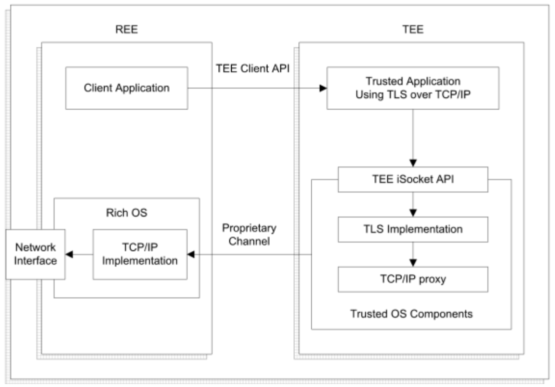

Fig. 1. Separation of Security Protocols and Pure Transport Protocols [18].

OPEN-TEE [19] is a related technology that provides a slightly different approach: it offers simplicity and compatibility between architectures in exchange for renouncing to the use of dedicated hardware features and their performance. Further bold approaches include virtualization techniques adapted to IoT edge devices. The virtualization system may or may not comply with Globalplatform standards, but it can benefit either way from dedicated security hardware features without the virtualized applications noticing. This type of system is transparent to the applications and can be more convenient for developers, as it simplifies the application design by truly decoupling security issues. Successful secure virtualization has even been carried out in devices with 2 mbytes of ROM and 512 kbytes of RAM [13].

This virtualization scheme could be complemented with a security decoupling technique for communications that would enhance an overall security solution without affecting either applications' portability or simplicity.

B. IoT verticals: a connectivity barrier to overcome

The great number of existing verticals for different IoT areas makes it imperative the creation of supra entities called *federations* [20]. Interconnecting different verticals usually requires complex semantic transversal middleware environments [21], but the European Telecommunications Standards Institute (ETSI), member of the oneM2M Global Initiative[4], is working towards "horizontalizing" the pipes [22] to simplify this feat.

To that end, it is remarked the necessity of a standardized horizontal middleware of sorts that could homogenize all underlying devices and technologies, simplifying intercommunication between services of different vertical approaches, as presented in Fig. 2.

RERUM[5] can be considered as a fine example of this effort: a middleware that helps to improve privacy and security in the IoT, but also promoting interoperability [23]. This approach intends to become the middleware suggested by ETSI (Common Application Infrastructure, Fig. 2) to simplify horizontal deployments across many IoT markets. However, instead of presenting a truly way to devise IoT federations through horizontal deployment, RERUM ends up becoming itself into an extraordinary flexible but vertical solution in practice, because of its mandatory IoT device abstraction scheme, a common issue in middleware-based approaches.

This abstraction is held by a local IoT device middleware (Fig. 3), which converts the IoT device (the physical device) into a RERUM Device (RD), i.e., a virtual entity easier to handle. This allows the main middleware modules (in servers) to benefit from a translation of all the resources and services that the IoT devices may provide, but as RD resources instead. Thus, this "virtualization" cannot be truly considered a successful security decoupling-based approach, since the IoT device software abstraction and the communication security features are not independent: the RD Adaptor is a part of the RD Middleware environment that lays in each IoT device and homogenizes them all, but also assuming all security communication responsibilities.

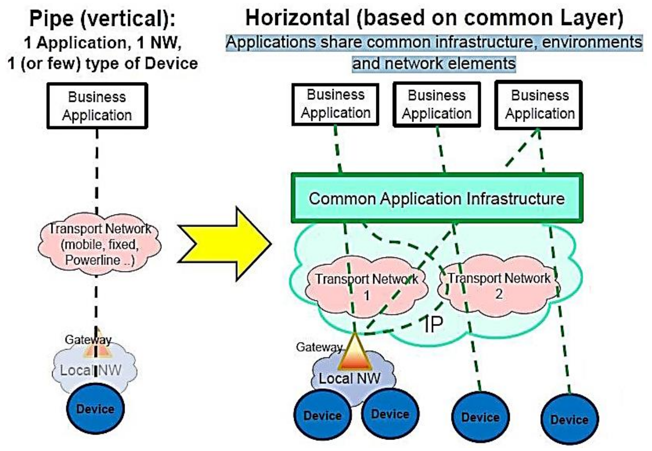

Fig. 2. Vertical and horizontal pipe standardization scenarios [22].

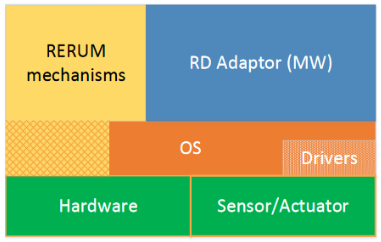

Fig. 3. RERUM device functional layers [23].

Thus, it may prove difficult for already designed devices to join the Common Application Infrastructure that RERUM provides, as this would require major changes to each IoT device's software design. Besides, this hardware abstraction technique also happens to result hardware dependent: it is only devised for ARM technologies, limiting the scope of the federation architecture to specific manufacturers [24]. Furthermore, security upgrades are to be made altogether with the middleware (not independent).

Anyhow, including a security decoupling approach in this scheme is feasible and can provide all its intrinsic benefits. This would only require minimal modifications of the proposed RERUM design, allowing a simpler IoT device abstraction approach and also different IoT device's architectures to join.

There exist other different approaches based on the use of middleware which usually simplifies its implementation through frameworks [25]. However, the majority of the solutions suffer from similar difficulties when handling security upgrades because the IoT device software is not fully independent from the middleware and its APIs.

C. IoTsafe: an IoT security decoupling approach

IoTsafe is a security decoupling scheme deployed in the communication layers. It is able to simplify some security problems (particularly when upgrading, see Fig. 4) and to reduce possible threats whilst improving interoperability and helping to comply with certification procedures [14].

This security decoupling technique is only straightforwardly deployed when an embedded Linux variant and the SSH protocol are present in the devices. Nevertheless, this solution should be understood as a possible realization of a general method to enable

[4] One M2M, standards for M2M and the Internet of Things, http://www.onem2m.org/

[5] RERUM: REliable, Resilient and secUre IoT for sMart city applications, https://ict-rerum.eu/

the "horizontalization of verticals," not necessarily requiring Linux or SSH [26]. Security decoupling can be presented as an "extension" of OSI communication layer in between the 4th and the 5th layers: IoTsafe uses sockets as interface (layer 4) but also operating system driven security contexts (layer 5) provided only through standard Linux kernel features.

IoTsafe proposes simplifying all IoT application software in the device by delegating all security and authentication issues to a complimentary stand-alone software module in charge of establishing a secure connection between the device and the gateway/server (Fig. 5). This piece of software should be the only one able to communicate, providing *transparent proxification* of all communication sockets to/from the IoT gateway/server.

Conversely, the local communications between sockets within the server (Fig. 6) are securely established from one socket to another following a virtual circuit commutation approach conceptually similar to the one illustrated in Fig. 7. This local communication inside the IoT gateway/server is fashioned and protected transparently through security contexts [14]. Linux kernel features there as a "communication hypervisor", transparent to the proper communications, as regular hypervisors deal with virtual machines. Such a security decoupling approach inherently offers interesting benefits [14]:

- Efficient application development: Security concerns are out of the IoT device application scope and can be delegated to complimentary software maintained by 3rd business parties specialized in security.

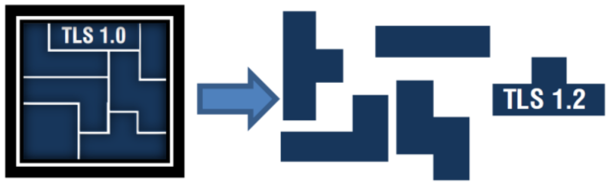

Fig. 4. Upgrading TLS on an IoT Core Application of an IoT device: It may require modifying the IoT Core Application in the device and rebuild it [15].

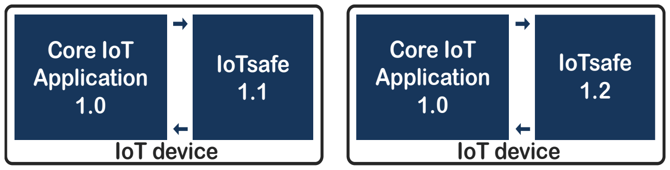

Fig. 5. Upgrading IoTsafe on an IoT device: IoTsafe stand-alone security modules interface with the core IoT Application through local sockets, easing security upgrading processes thanks to software independence [15].

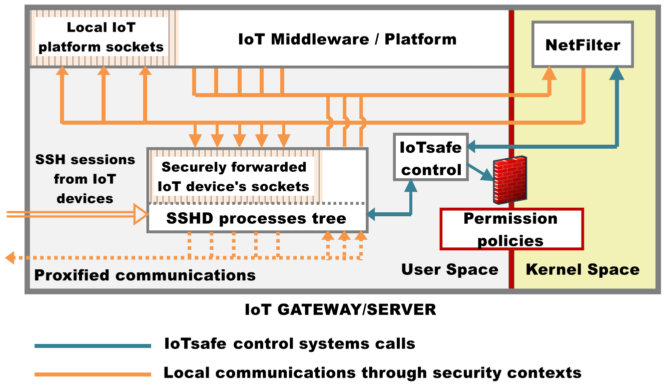
Local IoT platform sockets
IoT Middleware / Platform
NetFilter
SSH sessions from IoT devices
Securely forwarded IoT device's sockets
SSHD processes tree
IoTsafe control
Permission policies
Proxified communications
User Space
Kernel Space
IoT GATEWAY/SERVER
IoTsafe control systems calls
Local communications through security contexts

Fig. 6. IoTsafe conceptual scheme for local connections in the gateway/server.

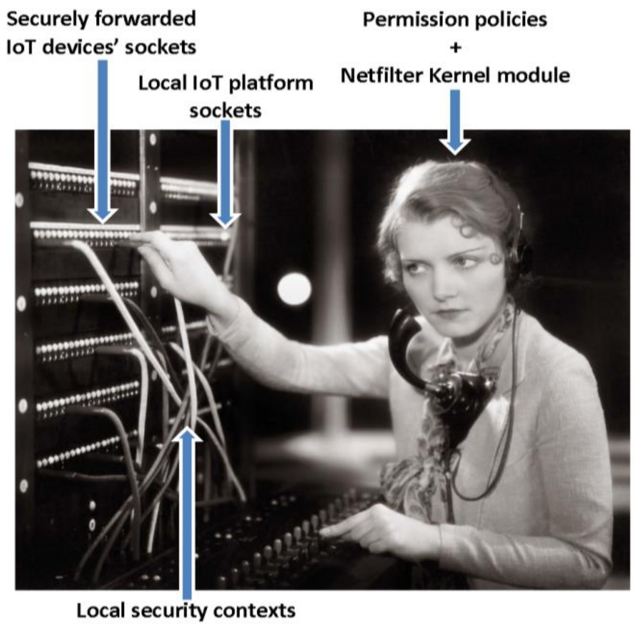
Securely forwarded IoT devices' sockets
Local IoT platform sockets
Permission policies + Netfilter Kernel module
Local security contexts

Fig. 7. IoTsafe gateway/server simplified communication scheme, rendered as an old-style manual circuit-switched telephone network.

- Simpler application's communication: End-to-end data is generated by insecure applications over a secure channel.
- Technology-independent communication: As secure digital communication may be transparently transmitted through insecure analog channels, even raw IoT devices' resources can be securely, transparently and independently transmitted through the underlying communication technology by using security decoupling techniques. Thus, local IoT device resources can be *forwarded* as plain network sockets (a much simpler interface than used by most APIs) into an IoT gateway/server, and remote resources can be securely *forwarded back* into the IoT device as well.

As a result, all the security concerns can be delegated nearly in full to this IoTsafe lower communication layer, and be upgraded and maintained by external 3rd parties offering an analogous service to, i.e., the one provided by antivirus security firms for computers and servers. This could allow a white-box testing procedure that would enable compliance for the highest security certification levels more affordably through the entire life cycle of the device. Simultaneously, interoperability would also be simplified, as related communication layer interfaces consist of plain local sockets handled transparently through security contexts: IoT server applications perceive all IoT devices' resources as local entities (plain sockets inside the machine) and so do the IoT devices with the ones provided by the IoT platform/middleware of the IoT gateway/server [14].

D. Security decoupling for constrained devices

Recent surveys [27] [28] [29] conducted among developers suggest that IoT solutions deployed into moderately constrained devices rely on Operating Systems (OS) or Real Time Operating Systems (RTOS) rather than on bare metal firmware. According to the most recent of those surveys [29], FreeRTOS[6] seems to be

[6] https://www.freertos.org/

one of the RTOS preferred for such projects by the developers, thanks to its wide hardware support and also by its available Portable Operating System Interface (POSIX) compliance.

IoTsafe's direct approach requires at least a POSIX environment and also SSH features available in the devices, something that the most constrained ones were unable to achieve. Nevertheless, novel initiatives interested in boosting SSH development into constrained equipment have arisen. As an example, WolfSSH[7] is able to provide SSH server v2 implementations for FreeRTOS with a minimal footprint of 32 kbytes and only requiring between 1.4 and 2 kbytes of RAM. This would enable IoTsafe to be run in more constrained devices straightforwardly. According to a recent IETF draft classification [30], such abled devices would be ranging from class 15 (0.5 GB to 1 GB RAM) to virtually class 1 (10 KiB of RAM, 100 KiB of ROM), as FreeRTOS hardware requirements are also rather slim.

Communication protocols used in constrained devices are diverse, and sometimes long-standing ones are included, such as HTTP, HTTP/2 and Message Queuing Telemetry Transport protocol (MQTT). On the other hand, CoAP is relatively new, particularly devised for constrained environments, and complies as a REpresentational State Transfer protocol (RESTful). It works over UDP and it is primarily aimed for small devices' payloads like sensors' and actuators'. Nevertheless, its bitwise extension also provides support for occasionally larger payload transfers, primarily for firmware upgrades [31].

As explained in the Introduction, the security decoupling approach should be independent from the underlying technology and thus, it is feasible to design different stand-alone modules (based for example on CoAP rather than on SSH), allowing this security-decoupling scheme also to use UDP. Such an approach would help to deal with security concerns, thanks to IoTsafe general security decoupling procedures, once they are found feasible in CoAP deployments. Besides, this decoupling functionality would help to separately address other problems, such as improving the behavior of the communications in lossy environments without modifying the core IoT CoAP application binaries.

Anyhow, in the pursuit of simplicity, the straightforward approach (based on Linux and SSH) is chosen for the tests presented in the next section, in order to simplify the procedure and also to allow cross-analyses with existing results for other protocols found in the literature [15].

III. Test Scenarios

The comparison of a security-decoupled approach and the standard one involves multiple test series where the IoT device performs several GET petitions to a gateway/server, requesting a resource. Both server and client pieces of software supporting CoAP are devised in Java using the Californium framework[8]. The resource offered is dynamic and provides random text. The figures of merit to be evaluated in the tests are the overhead, the loss rate of 802.15.4 frames and the required transfer time. These measurements are devised to feature the possible performance impact that a common lossy environment may have in each configuration. Other concerns left aside such as payload behavior dependence, 802.15.4 fragmentation or server security and scalability, have already been studied in depth [14].

The test series are run in two scenarios: the first one uses an almost lossless channel, and the second one is lossier with ~1% UDP's lost 802.15.4 frames. The test procedure in the lossy environment is devised to affect similarly the four cases and evidence of this is thoroughly elaborated in the Results. In both scenarios, 100 interleaved sets of 100 GET requests each with 1024 bytes of payload, are performed for each protocol configuration to analyze (a *case*). Once a set of 100 GET requests of one of the cases is finished, the following one proceeds until the four cases are completed (an iteration). The whole test battery comprises 100 iterations for each scenario.

Each set of 100 GET requests is performed in a row, i.e., serial, to avoid penalizing DTLS-based cases. In general, DTLS should perform a full secure handshake for each request while SSH approaches just establish a connection reusable for all the 100 requests. Thus, when CoAPs (CoAP with DTLS support) is run, DTLS can benefit from this, reusing the cryptographic established configuration for the 99 remaining requests. The four cases to consider at each scenario are as follows:

- CoAP, used straightforwardly, including a server and a client without any other special feature.
- CoAPs, similar to the previous one, but with DTLS.
- CoAP with SSH (denoted as CoAP_SSH in the graphs). It uses security decoupling based on an OpenSSH embodiment. This protocol provides security to CoAP server and client by socket proxification. To allow this, firstly all CoAP bidirectional traffic should be encapsulated into a TCP stream using Linux *socat* command, and then forwarded to the server trough an SSH established connection.

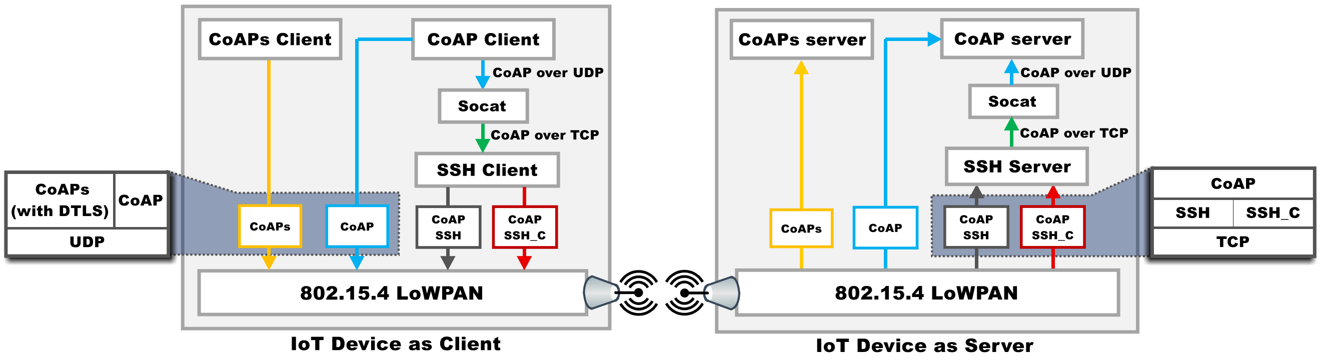

Fig. 8. IoT device test diagram to compare SSH vs. DTLS performance in CoAP communications.

[7] https://www.wolfssl.com/products/wolfssh/

[8] https://www.eclipse.org/californium/

- CoAP with SSH with compression (CoAP_SSH_C). It is similar to the previous case, but enabling the compression features provided by the ZLIB library [32] used by the SSH implementation.

The proposed scenarios (depicted in Fig. 8) comprise one device acting as a client and another one as a gateway/server. Both devices are Raspberry Pi 3 Model B V1.2 equipped with 802.15.4 Openlabs interfaces (based on the at86rf233 Amtel transceiver). Communications are established between the two Raspberries over WPAN.

The first scenario uses channel 26 (2480 MHz) so as to avoid 802.11 interferences. The second scenario employs channel 22 (2460 MHz) to benchmark a lossy channel (a common case in IoT scenarios), because of the existing interferences with the 802.11 channel 11 (in our case, 34 WiFi networks ranging from -59 to -75 dBm were present). Both 802.15.4 interfaces are configured at -2 dBm, placed one meter away from each other. A virtual LoWPAN interface is configured on both devices to fragment packets when needed: IPv6 requires a minimum MTU of 1280 bytes, while 802.15.4's MTU is 127 bytes (working with LoWPAN fragmentation, it only offers 96 bytes of payload). Linux kernel 4.7.4 is used, which includes a stable IEEE 802.15.4 LoWPAN implementation. The transfer rate of this device is set to 250 kbps, i.e. the maximum specified in the 802.15.4 standard. CoAPs implementation uses DTLS 1.2. During the tests, the 'TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256' algorithm is used, one of those recommended as Extensible Authentication Protocol-Transport Layer Security (EAP-TLS) Approved Algorithms [33]. This suite implements the Elliptic-curve Diffie-Hellman Ephemeral key exchange, using the Elliptic-curve Digital Signature Algorithm (ECDSA), with AES-128 as the block cipher, and SHA-256 for the hash message authentication code (HMAC).

SSH is provided by OpenSSH, configured to work with close (but safer) algorithms to establish connection: it uses curve25519-sha256 as an ECDSA key exchange algorithm, aes256-gcm as block cypher and hmac-512 as HMAC (this last algorithm is stronger than the one used in TLS because hmac-sha384 is no longer available as an HMAC algorithm in current OpenSSH default implementations). SSH compression is provided by ZLIB and can be activated by default editing the *ssh_config* file. TCP standard Raspbian parameters are used, and CoAP default values are slightly modified to be able to bear with the tests. The only modified variables are the following ones: *blockwise_status_lifetime*, *max_resource_body_size* and *max_peer_inactivity_period*.

IV. Results

The DTLS vs SSH comparison is performed using CoAPs directly and CoAP over SSH. The CoAP case (without security) is evaluated just to provide a reference. The server offers the CoAP/CoAPs text resource explained in the previous section, as IoT payloads are usually plain text with semantic-based descriptions [34]. Although this resource does not include binary executable data for payload testing, it is assumed it would have also moderate compressibility [35]. The tests are devised to measure several traffic features:

- T_{req}, the time required to fulfill 100 consecutive GET requests of 1024 bytes of random text each. Other payload values (lower and higher ones) and 802.15.4 fragmentation related issues have already been studied in depth in other TLS/SSH works [14], confirming that, the higher the payload is, the better SSH solutions render when its compression features are enabled. The value of 1024 has been chosen for the payload as it is low enough to avoid favoring SSH alternatives, whilst it is also large enough to generate 802.15.4 fragmentation.
- Overhead: percentage of extra information received at the network interface (beyond 1024*100 bytes) due to communication protocols, retransmissions, etc.
- Error rate: percentage of lost frames (sent or to be received).

A. Lossless scenario

Fig. 9 and Fig. 10 present the results of the tests in a nearly lossless environment. The scenario is static with just one 802.15.4 sender and one receiver in a channel free of 802.11 transmissions and thus, this assumption can be considered as valid. In these tests' results, it can be observed the effect that the default TCP/IP congestion control algorithm (Reno) has in the missing 802.15.4 frames metric for CoAP_SSH and CoAP_SSH_C cases. This algorithm tries to fully leverage the channel capacity potential at the cost of causing such missing frames. On the other hand, the more conservative approach followed by CoAP and CoAPs does not appreciably increase the frame loss rate.

These results show that the missing frame rate in CoAP_SSH and CoAP_SSH_C is not valid to clearly measure the quality of the channel. On the contrary, CoAP and CoAPs metrics in this field are not being affected by such a congestion control algorithm and thus, can be taken into consideration.

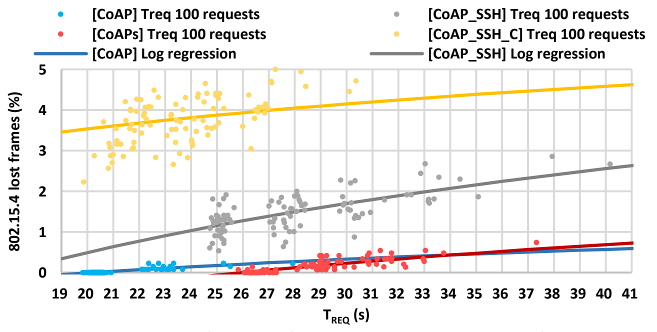

Fig. 9. T_{req} dispersion for sets of 100 CoAP GET requests of 1024 bytes of payload each with an avg. LQI causing ~0.1% UDP's lost 802.15.4 frames.

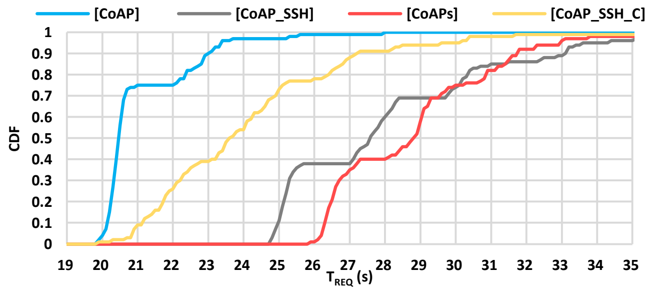

Fig. 10. CDF T_{req} of 100 GET requests of 1024 bytes of payload each with an avg. LQI causing ~0.1% UDP's lost 802.15.4 frames.

A summary of these results is presented in Table I. It is worth remarking that, although CoAP_SSH has an increase of 38% of overhead over CoAPs, it slightly outperforms the latter (2.29%) thanks to a better use of the channel; and when SSH compression features are enabled, this performance boost rises up to 14.26%.

During all the tests, the possible interferences are handled as interfering noise [36]. Such is the default configuration of the chip (*cca_mode* = 1) [9] for all interfering signals over -77 dBm (*cca_ed_level* = -77). An accepted approach to deal with interferences caused by 802.11 transmissions in 802.15.4 links is to consider them as noise causing channel fadings [37].

In 802.15.4 links, the Link Quality Indicator[9] (LQI) is presented as a better metric to assess the true quality state of the channel rather than the traditional Received Signal Strength Indicator (RSSI). This is mainly due to the effect that a narrow-band interferer inside the channel bandwidth may have in the transmissions (a fading), but with an RSSI unaltered or even higher in value [38]. Thus, the LQI may be better worked out considering *chip error rate* metrics [38]. Likewise, the 802.15.4 missing frames could serve as a valid indicator of the LQI if there is no other cause of its variability, as in CoAP and CoAPs cases.

B. Lossy scenario

Fig. 11 and 12 present the tests' results of a lossy scenario: a real channel with 802.11 interferences, where the sources of such interferences are not under control.

The nature of other protocols causing cross interferences with 802.15.4 has been studied in depth already using anechoic chambers [39]. However, we are considering a realistic scenario with multiple 802.11 interferences impractical to model. This poses a challenge, as we need to guarantee that the channel presents the same statistics in the four cases, so as not to draw skewed conclusions.

To address this question, the effect of the 802.11 interferences on the channel has been analyzed in each case by comparing the correlation of the required time (T_{req}) for each test, with the percentage of 802.15.4 missed frames. These results are plotted in Fig. 11 as a dispersion graph, and then a logarithmic regression is obtained for each result series.

The resilience to frame loss of each case can be featured using the frame loss metric in place of the LQI as error rate metric when the major cause of the T_{req} variability is the lossy effect of the channel. In our scenario configuration (a static sender and receiver) this could be confirmed if a tight correlation between the loss frame rate and T_{req} was observed. This fact is validated through the F-test [40] only in the cases where a conservative congestion control algorithm is implemented. This outcome

TABLE I
SUMMARY RESULTS 1ST SCENARIO

Case	Avg. frame error (%)	Avg. Treq (s)	ΔP* (%)	Avg. overhead (%)
CoAP	0.03	21.07	27.00	11.92
CoAPs	0.16	28.86	0.00	20.23
CoAP_SSH	1.49	28.20	2.29	58.92
CoAP_SSH_C	3.82	24.74	14.26	-8.14

* ΔP (%) = 100*(T_{req}_CoAPs-T_{req})/T_{req}_CoAPs

[9]http://ww1.microchip.com/downloads/en/devicedoc/atmel-8351-mcu_wireless-at86rf233_datasheet.pdf

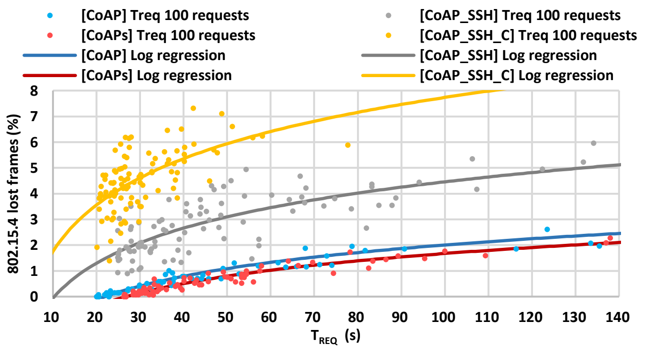

Fig. 11. T_{req} dispersion for sets of 100 CoAP GET requests of 1024 bytes of payload each with an avg. LQI causing ~1% UDP's lost 802.15.4 frames.

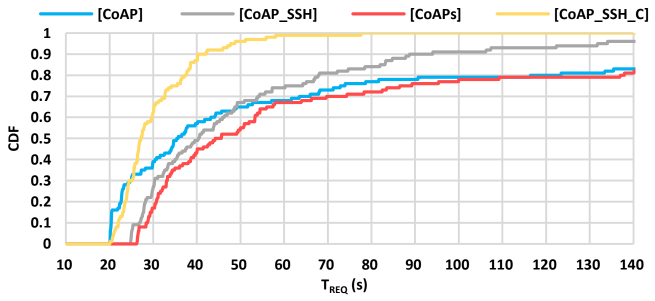

Fig. 12. CDF T_{req} of 100 GET requests of 1024 bytes of payload each with an avg. LQI causing ~1% UDP's lost 802.15.4 frames.

confirms that CoAP and CoAPs cases' results (missing frames and T_{req}) are not randomly scattered with a 90-95% of certainty and thus, due to such a high correlation it is valid to assume that in this particular testbed, *for a specific LQI value (unknown) there are close values of 'percentage of 802.15.4 missing frames' as well*. Due to the relationship between T_{req} and the missing frames metric, it can be stated as well that *for a specific 'percentage of 802.15.4 missing frames' there is a close range of the channel capacity*. In our case, this behavior can be easily appreciated in Fig. 11 but also in Fig. 9, where the affinity that CoAP and CoAPs have to their respective regression functions is observed.

The channel used can be considered a stochastic process of unknown properties and it is not expected to be stationary, nor even in a wide sense. Thus, it is not feasible to calculate long-term statistical properties of the channel such as its capacity (ergodic capacity), even after averaging enough channel fading episodes [41]. Nevertheless, if the channel's statistical features (though time dependent) progress slowly enough, it can be considered as a *local sense stationary process* [42]. Thus, it is possible to devise a testbed where the four cases are interleaved in a way that each case suffers similar interferences (fading) statistically, and at the same time, that the evolution of such a channel is slow enough so as to allow statistic similar interferences affect nearly equally to the four cases during the multiple (100 in our case) iterations, that is, the long term.

This behavior is presented in Fig. 13, where it is shown the analogous behavior the cumulative loss frame average has on CoAP and CoAPs cases. This explains why the percentage of lost 802.15.4 frames (inherent to the channel quality and capacity) is nearly the same in both CoAP and CoAPs cases after 100

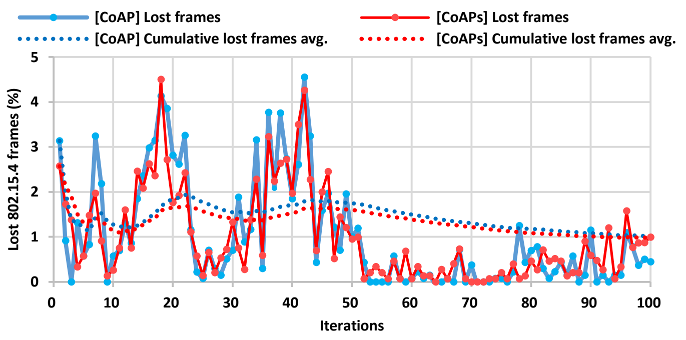

Fig. 13. Cumulative average of the percentage of lost 802.15.4 frames through the 100 set of tests in the second scenario.

iterations ([CoAP lost frames avg.]: 1.014% and [CoAPs lost frames avg.]: 0.986%). Furthermore, as the four cases are interleaved, it is reasonable to assume that CoAP_SSH and CoAP_SSH_C were also affected in the same way by the channel during the entirety of the tests. Therefore, it is feasible to statistically compare the behavior of the four different cases with different values of LQI, by considering the missing frame rate mark of CoAP or CoAPs as indirect references of the LQI status of the channel or its capacity.

On the other hand, the two cases in which SSH is used do not present such a correlation and thus, there exist multiple tests with similar T_{req} marks that suffer a wide range of frame error rate values (dispersion). This indicates that those cases (CoAP_SSH and CoAP_SSH_C) are more resilient to interferences than just plain CoAP/CoAPs: even though the LQI decreases (increase of missing frames for CoAP and CoAPs), SSH-based cases are able to adapt and make a better usage of the channel thanks to TCP default congestion control algorithm (Reno) and its ability to better adapt to the capacity of the channel.

C. Interference effects analysis

In the results displayed in Fig. 11, the CoAPs' lack of resilience to interference or noise is clearly made manifest. In addition, the left-aligned swarms of dots of the SSH cases, not only present a better harnessing of the channel capacity, but also a better resilience to interferences. In Fig. 12, it is shown how CoAP_SSH_C's ability for slashing the overhead and payload through compression proves to be instrumental this time: a decrease of the amount of information to be sent reduces the loss probability and also the time required to complete each request.

Compression can also contribute to reduce the components (software and hardware) work load for longer periods of time. The extra energy needed by these compressing features reduces transmission times, which saves a greater amount of energy that would be required otherwise by the application and the entire device's communication system [43] [14].

A summary of the average results is displayed in Table II. In this scenario, the overhead is higher: more retransmissions are requested in the four cases, because of the lossier behavior of the channel. Remarkably, CoAP_SSH significantly improves its transfer time when compared to CoAPs (up to 36%) and, when the compression feature is enabled, this figure rises up to 63%.

TABLE II
SUMMARY RESULTS 2ND SCENARIO

Case	Avg. frame error (%)	Avg. T_{req} (s)	ΔP* (%)	Avg. overhead (%)
CoAP	1.02	76	8.42	25.72
CoAPs	0.99	83	0	34.81
CoAP_SSH	2.91	53	36.30	89.71
CoAP_SSH_C	4.55	30	63.33	-4.54

* ΔP (%) = 100*(T_{req}_CoAPs-Treq)/T_{req}_CoAPs

V. CONCLUSIONS

IoT devices are in constant growth and ad-hoc vertical solutions are arising as responses to the foremost challenges our world faces today. New opportunities may appear from synergies between verticals, but connectivity between them is usually complex. Besides, security and privacy have become major concerns, not only for IoT developers, but between regulatory bodies as well. CoAP is frequently used as an application level protocol in constrained devices, as a substitute for HTTP to give support to IoT services of different kinds; and security decoupling proves to be an instrumental technique to enable the rise of new certification frameworks that grant a more standardized management of security and privacy.

This paper has provided evidence of the feasibility of security decoupling in applications using CoAP, and has also documented beneficial side effects that this approach can provide to 802.15.4 IoT devices in presence of 802.11 interference sources. Particularly, the straightforward IoTsafe approach has been evaluated against traditional CoAP deployments. The results show that this implementation is feasible allowing the IoT device application and the server software to use the unsecure version of CoAP (without DTLS) while seamlessly entrust security concerns to IoTsafe without requiring any other particular feature to be included in their design.

The performance impact of this straightforward approach does not penalize the communication performance in lossless environments, despite a 39% of overhead increase compared to CoAPs standard deployment.

In lossier scenarios because of 802.11 interferences, the decoupled scheme can reduce transfer time by 36% with respect to CoAPs, even though the overhead difference between CoAP_SSH and CoAPs increases (54.9%). If SSH compressing features are also put into action, the global overhead is the lowest of all cases (even negative, thanks to the compressibility of the payload) and the performance in transfer time boosts up to 14% in interference-free scenarios, and up to 63% when the channel experiments fading events causing an average of ~1% of CoAP or CoAPs 802.15.4 frames to be lost.

These results remark the importance of a congestion control algorithm and how decoupled solutions based on SSH/TCP can benefit from it without altering IoT core software and/or protocols. As future work, further optimized security decoupling approaches should be taken into consideration, such as new SSH UDP-based deployments or full security decoupling CoAP solution with more advanced congestion control algorithms, which may provide room for further improvement.

VI. ACKNOWLEDGEMENTS

This work was supported in part by Fondo Europeo de Desarrollo Regional Aragón Operative Programme 2014-2020 "Building Europe from Aragon" (Research Group T31_17R).

VII. References

[1] Cisco, "Internet of Things at a Glance" [Online]. Available: https://www.cisco.com/c/dam/en/us/products/collateral/se/internet-of-things/at-a-glance-c45-731471.pdf [Accessed 30 09 2019].

[2] J. Lin et. al., "A Survey on Internet of Things: Architecture, Enabling Technologies, Security and Privacy, and Applications", IEEE Internet of Things Journal, vol. 4, no. 5, pp. 1125 - 1142, 2017.

[3] I. Stellios, P. Kotzanikolaou, M. Psarakis, C. Alcaraz and J. Lopez, "A Survey of IoT-Enabled Cyberattacks: Assessing Attack Paths to Critical Infrastructures and Services", IEEE Communications Surveys & Tutorials, vol. 20, no. 4, pp. 3453 - 3495, 2018.

[4] M. Al-Fuqaha et al., "Internet of Things: A survey on enabling technologies protocols and applications", IEEE Communications Surveys & Tutorials, vol. 17, no. 4, pp. 2347-2376, 2015.

[5] I. Yaqoob et al., ", Internet of things architecture: Recent advances taxonomy requirements and open challenges", IEEE wireless communications, vol. 4, no. 3, pp. 10-16, 2017.

[6] B. Schneier, "Patching is failing as a security paradigm", in Click Here to Kill Everybody: Security and Survival in a Hyper-connected World, New York, W. W. Norton & Company, 2018.

[7] F. Restuccia, S. D'Oro and T. Melodia, "Securing the Internet of Things in the Age of Machine Learning and Software-Defined Networking", IEEE Internet of Things Journal, vol. 5, no. 6, pp. 4829 - 4842, 2018.

[8] A. Ghanbari et al., "Business Development in the Internet of Things: A Matter of Vertical Cooperation", IEEE Communications Magazine, vol. 55, no. 2, pp. 135 - 141, 2017.

[9] Directorate-General for Communications Networks, Content and Technology, "Cybersecurity Act", 19 September 2017. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/ALL/?uri=COM:2017:477:FIN [Accessed 18 07 2019]

[10] G. Gianmarco Baldini and A. L. Giannopoulos, "Analysis and recommendations for a European certification and labelling framework for cybersecurity in Europe [12183/17 ADD 9] ", Luxembourg: European Commission Joint Research Centre, 2017.

[11] M. A. Razzaque et al., "Middleware for Internet of Things: A Survey", IEEE Internet of Things Journal, vol. 3, no. 1, pp. 70 - 95, 2015.

[12] A. H. Ngu et al., "IoT Middleware: A Survey on Issues and Enabling Technologies", IEEE Internet of Things Journal, vol. 4, no. 1, 2017.

[13] R. T. Tiburski, C. R. Moratelli, S. F. Johann, et al., "Lightweight Security Architecture Based on Embedded Virtualization and Trust Mechanisms for IoT Edge Devices," IEEE Communications Magazine, vol. 57, no. 2, pp. 67 - 73, 2019.

[14] J. D. de Hoz-Diego, J. Saldana, J. Fernandez-Navajas and J. Ruiz-Mas, "IoTsafe, Decoupling Security From Applications for a Safer IoT", IEEE Access, vol. 7, pp. 29942 – 29962, 2019.

[15] J. D. de Hoz-Diego, J. Saldana, J. Fernández-Navajas, J. Ruiz-Mas, et al. "SSH as an Alternative to TLS in IoT Environments using HTTP", in Global Internet of Things Summit, Bilbao, 2018.

[16] G. Arfaoui, S. Gharout and J. Traoré, "Trusted Execution Environments: A Look under the Hood", in IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, Oxford, UK, 2014.

[17] M. Sabt, M. Achemlal and A. Bouabdallah, "Trusted Execution Environment: What It is, and What It is Not", in IEEE Trustcom/BigDataSE/ISPA, Helsinki, Finland, 2015.

[18] GlobalPlatform Inc., "GlobalPlatform Device Technology: TEE Sockets API Specification", January 2017. [Online]. Available: https://globalplatform.org/specs-library/tee-sockets-api-specification-v1-0-1/ [Accessed 16 08 2019].

[19] B. McGillion, T. Dettenborn, T. Nyman and N. Asokan, "Open-TEE -- An Open Virtual Trusted Execution Environment", in IEEE Trustcom/BigDataSE/ISPA, Helsinki, Finland, 2015.

[20] L. Sánchez et al., "Federation of Internet of Things Testbeds for the Realization of a Semantically-Enabled Multi-Domain Data Marketplace", Sensors, vol. 18, no. 10, p. 3375, 2018.

[21] P. P. Ray, "A survey of IoT cloud platforms", Future Computing and Informatics Journal, vol. 1, no. 1-2, pp. 35-46, 2016.

[22] R. Minerva, A. Biru and D. Rotondi, "Towards a definition of the Internet of Things (IoT)", IEEE Internet Initiative, no. 1, 2015.

[23] RERUM consortium members, "Final System Architecture, RERUM FP7-ICT-609094", 4 September 2015.

[24] G. Moldovan et al, "An IoT Middleware for Enhanced Security and Privacy: The RERUM Approach", IFIP International Conference on New Technologies, Mobility and Security, Larnaca, Cyprus, 2016.

[25] M. Ammar, G. Russello and B. Crispo, "Internet of Things: A survey on the security of IoT frameworks", Journal of Information Security and Applications, vol. 38, pp. 8-27, February 2018.

[26] J. D. de Hoz-Diego, "Secure Communication Method And System Using Network Socket Proxying". PCT Application WO/2019/059754

[27] Eclipse IoT Working Group, IEEE IoT, AGILE IoT, "IoT Developer Survey" [Online]. Available: http://iot.ieee.org/images/files/pdf/iot-developer-survey-2016-report-final.pdf [Accessed 2019 07 18].

[28] Eclipse IoT Working Group, IEEE IoT, AGILE IoT, "IoT Developer Survey", 04 2017. [Online]. Available: https://ianskerrett.wordpress.com/2017/04/19/iot-developer-trends-2017-edition/ [Accessed 18 07 2019].

[29] Eclipse Foundation, "IoT Developer Survey 2019 Results", 04 2019. [Online]. Available: https://iot.eclipse.org/resources/iot-developer-survey/iot-developer-survey-2019.pdf. [Accessed 18 07 2019].

[30] C. Bormann, M. Ersue, A. Keranen and C. Gomez, "Terminology for Constrained-Node Networks: draft-bormann-lwig-7228bis-04", 11 March 2019. [Online]. Available: https://tools.ietf.org/html/draft-bormann-lwig-7228bis-04 [Accessed 18 07 2019].

[31] C. Bormann and Z. Shelby, "Block-Wise Transfers in the Constrained Application Protocol (CoAP)", August 2016. [Online]. Available: https://tools.ietf.org/html/rfc7959 [Accessed 18 07 2019].

[32] T. Ylonen and C. Lonvick, "The Secure Shell (SSH) Transport Layer Protocol", Internet Engineering Task Force, January 2006. [Online]. Available: https://tools.ietf.org/html/rfc4253 [Accessed 18 07 2019].

[33] NSA: Information Assurance Directorate, "Wireless Local Area Network Capability Package V.2.2", June 2018 [Online]. Available: https://www.nsa.gov/Portals/70/documents/resources/everyone/csfc/capability-packages/WLANCPv2.2_20180626.pdf [Accessed 18 07 2019].

[34] S. Kanti Datta and C. Bonnet, "Describing Things in the Internet of Things. From CoRE Link Format to Semantic Based Descriptions", in International Conference on Consumer Electronics, Taiwan, 2016.

[35] D. Chanet, et al. "Automated reduction of the memory footprint of the Linux kernel", ACM Transactions on Embedded Computing Systems (TECS) - Special Section LCTES'05, vol. 6, no. 4, 2007.

[36] J.-H. Hauer and V. Handziski, "Experimental Study of the Impact of WLAN Interference on IEEE 802.15.4 Body Area Networks", in EWSN '09 Proceedings of the 6th European Conference on Wireless Sensor Networks, Cork, Ireland, 2009.

[37] Y. Dong et al., "Wireless coexistence between IEEE 802.11 and IEEE 802.15.4 based networks: A survey", International Journal of Distributed Sensor Networks, pp. 1550-1329, July 2011.

[38] Chipcon, "CC24202.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver", 09 06 2004.

[39] A. Hithnawi, H. Shafagh and S. Duquennoy, "802.15.4, Understanding the Impact of Cross TechnologyInterference on IEEE", in 9th ACM international workshop on Wireless network testbeds, experimental evaluation and characterization , Maui, Hawaii, USA, 2014.

[40] R. G. Lomax and D. L. Hahs-Vaughn, Statistical Concepts: A Second Course., Routledge, 2000.

[41] U. A. Chude-Okonkwo, R. Ngah and T. Abd Rahman, "Time-scale domain characterization of non-WSSUS wideband channels", EURASIP Journal on Advances in Signal Processingvolume, 2011.

[42] U. A. Chude Okonkwo et al., "Time-scale domain characterization of nonstationary wideband vehicle-to-vehicle propagation channel", in IEEE Asia-Pacific Conference on Applied Electromagnetics (APACE), Port Dickson, Malaysia, 2010.

[43] M. Suárez-Albela et al., "A Practical Evaluation of a High-Security Energy-Efficient Gateway for IoT Fog Computing Applications", Sensors, no. 9, p. 1978, 2017.