Performance Evaluation of Attribute-Based Encryption on Constrained IoT Devices[☆]

Pericle Perazzo[*], Francesca Righetti[*], Michele La Manna[†*], Carlo Vallati[*]

Abstract

The Internet of Things (IoT) is enabling a new generation of innovative services based on the seamless integration of smart objects into information systems. This raises new security and privacy challenges that require novel cryptographic methods. Attribute-Based Encryption (ABE) is a type of public-key encryption that enforces a fine-grained access control on encrypted data based on flexible access policies. The feasibility of ABE adoption in fully-fledged computing systems, i.e., smartphones or embedded systems, has been demonstrated in recent works. In this paper, we consider IoT devices characterized by strong limitations in terms of computing, storage, and power. Specifically, we assess the performance of ABE in typical IoT constrained devices. We evaluate the performance of three representative ABE schemes configured considering the worst-case scenario on two popular IoT platforms, namely ESP32 and RE-Mote. Our results show that, if we assume to employ up to 10 attributes in ciphertexts and to leverage hardware cryptographic acceleration, then ABE can indeed be adopted on devices with very limited memory and computing power, while obtaining a satisfactory battery lifetime. In our experiments, as also performed in other works in the literature, we consider only the worst-case configuration, which, however, might not be completely representative of the real working conditions of sensors employing ABE. For this reason, we complete our evaluation by proposing a novel benchmark method that we used to complement the experiments by evaluating the average performance. We show that by always considering the worst case, the current literature significantly overestimates the processing time and the energy consumption.

Keywords: IoT, constrained devices, security, attribute-based encryption

1. Introduction

Recent advancements in wireless communication standards and embedded computing are fostering the creation of novel smart computing systems, which are rapidly getting real in heterogeneous contexts, from personal to industrial. A crucial enabling technology will be the Internet of Things (IoT), which refers to the multitude of heterogeneous smart objects seamlessly integrated into computing platforms. These IoT devices represent the bridge between the physical and the cyber worlds and enable novel functionalities, such as remote monitoring and big data collection for intelligent control and optimization [1, 2].

[☆]A preliminary version of this paper has been published as B. Girgenti, P. Perazzo, C. Vallati, F. Righetti, G. Dini and G. Anastasi, ”On the Feasibility of Attribute-Based Encryption on Constrained IoT Devices for Smart Systems,” 2019 IEEE International Conference on Smart Computing (SMARTCOMP), Washington, DC, USA, 2019, pp. 225-232.

The majority of IoT devices are resource constrained, i.e., characterized by scarce capabilities and features. IoT devices are typically implemented through low-cost embedded systems that have reduced computing and storage capabilities and are often battery powered. The scarcity of resources on those devices is currently driving the definition of specific network protocols that can accommodate the reduced features offered by them. An example is the Constrained Application Protocol (CoAP) [3], which is an application protocol tailored to allow applications to communicate with constrained devices.

In order to compensate the limited capabilities of IoT devices, more complex architectures are usually put in place, to allow the implementation of advanced services on top of the functionalities they offer. IoT systems are usually implemented in a multi-layered fashion, in which IoT devices are integrated into cloud-computing platforms. Intermediate devices such as gateways or brokers are usually installed in order to implement functionalities like protocol translation, data dispatching, or to support the execution of simple applications that require proximity with the IoT devices due to time constraints.

In this complex architecture, data generated by IoT devices can be processed by multiple heterogeneous entities, which can be either different applications interested in analyzing the data or the above mentioned intermediate entities. In this context, novel encryption mechanisms are required to enforce security and guarantee a fine-grained access control over data. The latter, in particular, is an important requirement to tune the amount of information that can be accessed by each entity handling the data. For instance, while applications should have complete access to the data generated by IoT devices, an intermediate entity, like a broker, should have access only to the minimum set of information required to implement its functionalities [4]. Current security methods adopted in IoT are based solely on encrypted channels between the broker and the IoT devices, plus optionally an access control mechanism enforced by the broker. Secure channels however do not avoid the broker to access to all data in the clear, and thus an attacker that compromises the broker is able to jeopardize the confidentiality of the whole communication system.

Attribute-Based Encryption (ABE) is a public-key encryption technique that encrypts data and at the same time enforces a fine-grained access control on it based on flexible access policies. Broadly speaking, with ABE a source can encrypt data by using a set of attributes, and only those destinations that fulfill an access policy defined over such attributes can decrypt data afterwards. Unlike classic access control techniques, ABE *mathematically* enforces the access policy, in such a way that only authorized entities are able to decrypt data. As a consequence, ABE allows the broker to manage only encrypted data, so that an attacker compromising the broker cannot break data confidentiality. ABE adoption is foreseen as a crucial technique to handle many security issues in different scenarios, ranging from healthcare systems [5, 6] to smart city [7, 8] and smart home [9] services, from financial industry [10] to on-line social networks [11].

The academic literature has partially assessed the feasibility of adopting ABE in different contexts, from fully-fledged embedded IoT systems [12] to smartphones [13]. Its adoption in *constrained* IoT devices, instead, has not been investigated so far. Understanding the feasibility limits of adopting ABE in constrained IoT devices allows us to understand its current applicability to a vast range of IoT applications. In this work we carry out an extensive evaluation of ABE performance in constrained IoT devices. Specifically, we assess the performance of different ABE schemes on different devices with different memory and computational capabilities. We implemented three representative ABE schemes and test their performance on two popular IoT platforms, the ESP32 and the RE-Mote platforms. We selected these two IoT platforms for the evaluation as they are representative of the devices currently available in the market. The ABE schemes have been configured considering a worst-case scenario, often adopted in literature, in order to check the feasibility of adopting ABE on constrained devices in the most challenging conditions. Our performance evaluation shows that ABE has a significant impact on the lifetime of battery-powered devices, especially when a high number of attributes (i.e., 20-50) is used in ciphertexts. However, if we assume to employ fewer attributes (up to 10) and to leverage hardware elliptic-curve cryptographic acceleration, which is present on some platforms (e.g., RE-Mote), then ABE can indeed be adopted on devices with very limited memory and computing power. We also obtain a significant, yet tolerable, battery lifetime reduction.

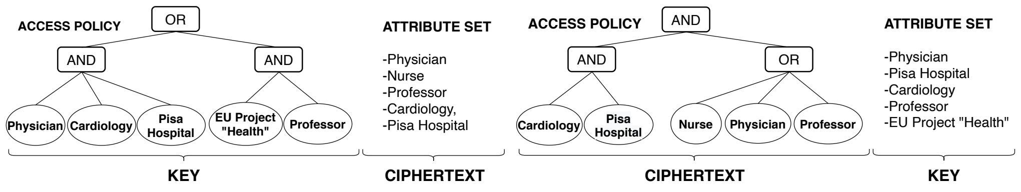
The worst-case configuration considered in our experiments is often adopted in literature, however, it might not be completely representative of the real working conditions of sensors employing ABE, as ABE configuration adopted in real use cases can often give better performance than the worst-case configuration. For this reason, we propose a novel benchmark method that allows us to estimate the average performance with a better accuracy with respect to the worst-case analysis. Our benchmark method is applicable to any ABE scheme, and it provides for a more realistic performance evaluation because it captures the average case. We exploited such a method to complete our evaluation. We show that the worst-case analysis significantly overestimates the processing time and energy. For example, with RE-Mote under some configurations, the energy consumption estimated from the average case is 67% less than that estimated from the worst case.

This paper is an extended version of a conference paper [14]. Compared with our conference work, the preliminary analysis has been extended as follows: (i) an additional IoT platform has been considered, the RE-Mote platform, to make the range of tested IoT platforms more representative by considering devices characterized by very limited memory capabilities and capable of hardware support for some of the cryptographic operations required to implement ABE schemes; (ii) a novel benchmark method to estimate the average-case performance has been introduced and used to complete our evaluation.

The rest of the paper is organized as follows. Section 2 presents some background on ABE. Section 3 overviews related work. Section 4 introduces a set of reference use cases and threat models. Section 5 introduces the hardware platforms and the methodology adopted in our experiments. Section 6 presents the experimental results for the ESP32 and the RE-Mote platforms. Section 7 presents the novel benchmark method for estimating the average-case time and energy consumption and the results obtained with it. Finally, Section 8 concludes the paper.

2. Attribute-Based Encryption

Attribute-Based Encryption (ABE) [15, 16, 17, 18] is a cryptography paradigm which allows one to encrypt a message in such a way that only a set of authorized parties can decrypt it afterwards. Such an authorization is expressed through an *access policy*, which is a Boolean function evaluated on some *attributes*. ABE can be viewed as a self-enforcing fine-grained access control mechanism based on cryptography. Moreover, it is a public-key encryption technique, in the sense that the key used for encrypting (*encryption key*) can be public, thus allowing everyone to encrypt. The keys used for decrypting (*decryption keys*) are instead private and unique for each party. An access policy can be expressed by means of a tree (*policy tree*), in which leaves represent attributes (Boolean arguments), while internal nodes represent "AND" and "OR" Boolean operators. The "NOT" operator is not permitted in the majority of ABE schemes, which thus allow only for *monotonic* access policies (see [19] for an exception). At decryption time, the access policy is evaluated with an *attribute set* as argument. The attribute set includes all the "true"



(a) KP-ABE (b) CP-ABE

Fig. 1: Example of realistic policies and attribute sets for KP-ABE (a) and for CP-ABE (b).

attributes. All the attributes excluded from the attribute set are implicitly considered "false". If the access policy evaluates to true, then data will be decryptable, otherwise data will not be decryptable.

Two main ABE paradigms have emerged in the literature, namely, *Key-Policy ABE* (*KP-ABE*) and *Ciphertext-Policy ABE* (*CP-ABE*). In the KP-ABE paradigm [16], the access policy is associated to the decryption key, and the attribute set is associated to the ciphertext. Thus, the attribute set semantically describes data. The KP-ABE paradigm can be viewed as a system that tags data with attributes, and then it gives users a "ticket" that tells which data (s)he can decrypt and which not. On the other hand, in the CP-ABE paradigm [17], the access policy is associated to the ciphertext, and the attribute set is associated to the decryption key. Thus, the attribute set semantically describes a decrypting user. The CP-ABE paradigm can be viewed as a system that tags each user with a set of attributes, and then it associates an access policy to each piece of data.

All the KP-ABE and CP-ABE schemes implement at least the following algorithms: **Setup**, **KeyGen**, **Encrypt**, and **Decrypt**. In both KP-ABE and CP-ABE paradigms the **Setup** algorithm generates a *master key* MK and an encryption key EK. The **KeyGen** algorithm generates a *decryption key* DK. In KP-ABE schemes, such an algorithm takes as input the master key and an access policy $\mathcal{T}$, which describes the owner of the generated decryption key. In CP-ABE schemes, it takes as input the master key and an attribute set γ, which describes the owner of the generated decryption key. The **Encrypt** algorithm generates a *ciphertext* C. In KP-ABE schemes, such an algorithm takes as input the encryption key, a message M, and an attribute set γ, which describes the data being encrypted. In CP-ABE schemes, it takes as input the encryption key, a message M, and an access policy $\mathcal{T}$, which describes the data being encrypted. In both KP-ABE and CP-ABE paradigms the **Decrypt** algorithm takes as input a decryption key and a ciphertext, returning the decrypted message if the attribute set satisfies the access policy.

In Fig. 1 we show a realistic use case in the e-health scenario, described with a KP-ABE paradigm (Fig. 1a) and a CP-ABE one (Fig. 1b). In both paradigms, the key describes a physician working at the Cardiology department of the Pisa Hospital, who is also involved in an European Project as a professor. The ciphertext represents some cardiology data retrieved in the Pisa Hospital, destined to be accessed by nurses, physicians, or professors.

3. Related Work

The application of ABE schemes to implement fine-grained access control and confidentiality has been already proposed in different contexts, e.g. healthcare [6], smart city [8], financial industry [10] and on-line social networks [11], but none of these studies focused on assessing the cost of introducing ABE in practice. Instead, an evaluation of the adoption of ABE has been carried in the following works.

In [20], the authors made the first full benchmark of a KP-ABE scheme and a CP-ABE scheme in terms of execution time, energy consumption, memory usage, data overhead. Their benchmark is carried out on a PC-class device (Intel Quad-Core i7 @ 1.60GHz) and a mobile device (Intel Atom Z2460 @ 1.60GHz smartphone). In [13], the authors evaluated the feasibility of adopting ABE on smartphone devices. Specifically, the authors developed an ABE library for the Android operating system, and then they evaluated its performance by means of real experiments. In [21], the authors carried out a comprehensive analysis of different ABE schemes, with respect to their application for decentralized secure data sharing. In particular, they performed a realistic estimation of the resource consumption and workload exploiting real-world system traces. Their evaluation considered heterogeneous devices, namely a laptop and a smartphone.

The results of [13, 20, 21] confirmed the possibility of using ABE on laptops and smartphones, showing that such devices have an acceptable amount of resources to implement ABE schemes and the resulting energy cost is acceptable. The following works focused instead on more constrained devices. In [4], the authors surveyed the various existing implementations of ABE schemes and, as a side contribution, they perform a benchmark on a single-board computer (Raspberry Pi 2 @ 900MHz) of various ABE schemes with the Charm library, used for fast prototyping of cryptographic schemes. Similarly, in [12] the authors assess the feasibility of using ABE in single-board computers, namely Raspberry Pi and Intel Edison. Experimental results demonstrate that exploiting ABE in such systems is feasible, although they also highlight that future works should focus on improve its efficiency. Notably, the authors of [4] and [12] do not focus on *actually* constrained IoT devices, but on more powerful platforms that have resources comparable with smartphone devices. Specifically, single-boards computers like Raspberry Pi and Intel Edison have enough resources to run a fully-fledged operating system. However, it is not clear from their results whether ABE schemes are feasible on far more constrained devices, which is the focus of our work.

In this paper we consider constrained devices with significantly less memory/computing capabilities, i.e. boards equipped with a microcontroller with less than 1MB of RAM. Those devices, very popular in IoT solutions, cannot support the execution of a fully-fledged OS and usually run an ad-hoc OS with limited features. Our analysis shows that, if we assume to employ simple access policies and to leverage hardware elliptic-curve cryptographic acceleration, then ABE can indeed be adopted on devices with very limited memory and computing power.

As a final note, all the previous papers [20, 4, 13, 12, 21] evaluate the decryption performance considering the worst-case scenario of policies using only "AND" operators. In this paper we show that when relying on worst-case decryption, the processing time and the energy consumption are significantly overestimated. We thus propose a novel benchmark method that allows us to evaluate the average decryption performance, instead of the worst-case one, complementing our experimental analysis.

4. Use Case

A popular application scenario for ABE that largely involves constrained devices is the medical field. Future medical systems will largely adopt Wireless Body Area Networks (WBANs) [6] to collect data. Patients that require continuous monitoring, for instance, will be equipped with wearable and/or implantable sensors, which collect biometric parameters for real-time monitoring, e.g. to ensure a rapid response in case of an emergency or automate the administration of treatments. These WBANs produce highly-sensible data, which is consumed by other con-

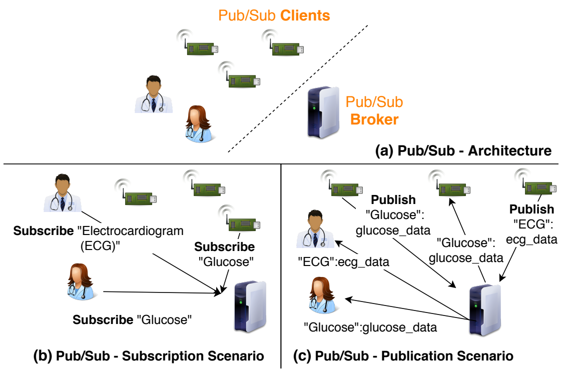

Pub/Sub Clients
Pub/Sub Broker
(a) Pub/Sub - Architecture
Subscribe "Electrocardiogram (ECG)"
Subscribe "Glucose"
Subscribe "Glucose"
(b) Pub/Sub - Subscription Scenario
Publish "Glucose": glucose_data
"Glucose": glucose_data
Publish "ECG": ecg_data
"ECG":ecg_data
"Glucose":glucose_data
(c) Pub/Sub - Publication Scenario

Fig. 2: Publish/subscribe architecture and mechanism

strained devices, e.g. an insulin pump that analyzes data from other biomedical sensors to select the proper dose, or by humans, e.g. a doctor that remotely checks the status of a patient. In this context, data should be protected from unauthorized accesses by means of encryption. However, since multiple recipients are involved, a fine-grained access control is mandatory to regulate which piece of information can be accessed by which user or device of the system. For instance, a glucose sensor can be programmed to encrypt its measurements in order to allow only the insulin pump and the patient's physician to access them.

In such applications, the information is often shared using a *publish/subscribe* system. Publish/subscribe is a common information-flow pattern adopted by different IoT application protocols, such as the Message Queue Telemetry Transport (MQTT) protocol [22] and the Constrained Application Protocol (CoAP) [23], to decouple the producer of information to the consumer. The overall architecture of a publish/subscribe system is depicted in Fig. 2(a). On one side, we have a set of constrained IoT devices, e.g. sensors or actuators, and users that behave as publish/subscribe clients and produce and consume messages, e.g. periodic updates on a physical measurement. On the other side, we have a broker, which is a full-resource device that is responsible for receiving, storing and dispatching messages. An IoT device or a user that is interested in receiving messages on a given *topic* contacts the broker to issue a subscription to that topic (Fig. 2(b)). Every time an IoT device generates new data for a given topic, it sends a message to the broker. The broker is responsible for dispatching messages to all the subscribers (Fig. 2(c)). This approach allows us to overcome the main limitations that characterize constrained IoT devices. First, their limited capabilities in terms of memory and computational power allow them to interact only with one application at a time. The adoption of a broker, instead, allows such devices to be used by multiple applications at the same time, thanks to the dispatching capabilities of the broker. Secondly, the publish/subscribe architecture facilitates the communication with battery-powered IoT devices. In order to minimize the energy consumption, such devices should often operate in power-saving mode in which they turn off their radio. In a publish/subscribe architecture, the broker can store the generated messages, thus allowing the IoT devices to go in sleep mode, without compromising information availability.

The core of this architecture is the broker, which can access all the messages. Such entity is often outsourced, i.e. it is deployed on an external infrastructure, e.g. a cloud computing platform, or is completely operated by external entities, e.g. cloud computing providers, which offer MQTT brokers as a service to customers. For those reasons, a broker is not only subject to external attackers, but it could be directly managed by an untrusted third party.

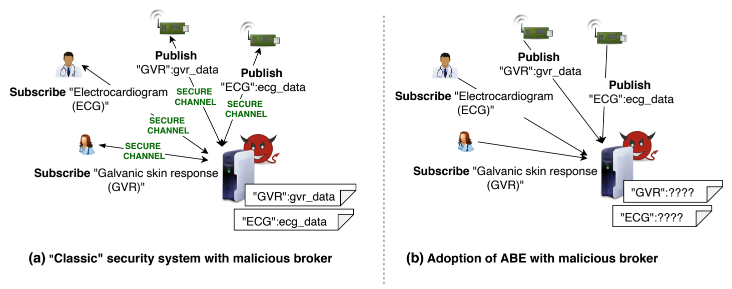

Publish
"GVR":gvr_data
Publish
"ECG":ecg_data
Subscribe "Electrocardiogram (ECG)"
SECURE CHANNEL
SECURE CHANNEL
SECURE CHANNEL
SECURE CHANNEL
Subscribe "Galvanic skin response (GVR)"
"GVR":gvr_data
"ECG":ecg_data
(a) "Classic" security system with malicious broker
Publish
"GVR":gvr_data
Publish
"ECG":ecg_data
Subscribe "Electrocardiogram (ECG)"
Subscribe "Galvanic skin response (GVR)"
"GVR":????
"ECG":????
(b) Adoption of ABE with malicious broker

Fig. 3: Malicious broker with traditional ABAC mechanism (a) and with ABE (b)

In this context, we consider an adversary capable to compromise the broker. However, it is worth to highlight that the same considerations apply if the broker's owner tries to access data dispatched by the broker itself. Thus, hereafter we refer to a broker that is either compromised or owned by an untrusted third party server, as a *malicious broker*. A traditional ABAC mechanism enforced by the broker would have exposed data to a confidentiality risk in case of a malicious broker. This is due to the fact that those "classic" security systems are based entirely on secure channels (e.g. TLS), in which the broker establishes a secure channel with each entity involved in the architecture. In such systems, the broker can access all the messages, thus it is a single point of trust, as shown in Fig. 3(a).

With ABE, instead, a malicious broker cannot do much, since it stores and dispatches messages in an encrypted fashion, so that the broker is not able of decrypting them (see Fig. 3(b)). Such a resistance to this adversary is given by ABE itself: ABE ensures that the broker has access only to metadata, e.g. the data type or the topic, but not to the data itself, which is stored and dispatched by the broker in an encrypted form.

Note that ABE does not preclude the possibility to use *also* secure channels, for example as a means of authenticating the messages. In addition to this, ABE enforces a fine-grained access control on encrypted data, thus preventing malicious or compromised applications to access unauthorized data.

5. Experimental Setup

In this section we present the experimental setup adopted for our performance evaluation. Specifically, in the followings, we first introduce our reference ABE schemes, the adopted hardware and software platforms, and then we present the methodology.

5.1. *Reference ABE Schemes*

In this paper we focus on three representative ABE schemes, namely: (i) the Goyal-Pandey-Sahai-Waters scheme [16][1] (throughout referred to as "GPSW", for brevity), which has been the first proposed KP-ABE scheme in the literature; (ii) the Bethencourt-Sahai-Waters scheme [17] ("BSW"), which has been the first proposed CP-ABE scheme in the literature; and (iii) the Yao-Chen-Tian scheme [18] ("YCT"), which is a quite recent KP-ABE scheme for IoT devices focused on encryption and decryption efficiency.

[1]In the cited paper, the authors actually present two schemes, offering respectively a small and a large attribute universe. We refer to the first one, which is the most lightweight of the two, thus suitable for very constrained devices.

In all the three considered schemes, the most expensive operation performed by the encryption algorithm is the *point-scalar multiplication*, which is an elliptic-curve operation (see [24] for details). In particular, the GPSW and the YCT schemes perform one point-scalar multiplication for each attribute in the attribute set. The BSW scheme performs two point-scalar multiplications for each leaf in the policy tree, and for each internal node of the policy tree, the BSW scheme creates a random polynomial of zero degree if the node is an OR operator, or degree equal to the number of children minus one if the node is an AND operator. Furthermore, the BSW scheme also performs a hashing of each attribute name on an *elliptic-curve group* as a first stage of the encryption operation. Since this hashing operation has a non-negligible impact on encryption performance, and since it can be easily precomputed given the set of attribute names that the encrypting device uses, we chose *not* to include the hash operations in our performance evaluation. In both GPSW and BSW schemes, the most expensive operation performed by the decryption algorithm is the *bilinear pairing*, which is a quite expensive cryptographic operation (see again [24] for details). Remind that the decryption algorithm does not have to visit all the nodes of the tree, but only a subset of them necessary to reach the root. The GPSW scheme performs one bilinear pairing for each visited leaf in the policy tree. On the other hand, the YCT scheme does not use bilinear pairing, so it is more efficient than GPSW. The YCT scheme performs one point-scalar multiplication for each visited leaf in the policy tree. The BSW scheme performs two bilinear pairings for each visited leaf in the policy tree.

5.2. Hardware and Software Platforms

As an example of constrained IoT devices, in our experiments we exploited the ESP32 and the RE-Mote boards. We have chosen those two IoT platforms as they are representative of two different categories of IoT devices currently available in the market, i.e., very constrained devices (like the RE-Mote board) designed to operate on batteries characterized by scarce memory/computing capabilities and equipped with a ultra low-power wireless transceiver (e.g. IEEE 802.15.4) and constrained devices with slightly more memory and computing capabilities equipped with a WiFi transceiver (like the ESP board).

The ESP32 [25] is an IoT platform produced by Espressif Systems that is growing in popularity due to its low cost, high availability and rich set of features. A dual-core Xtensa LX6 microprocessor at 240 MHz designed to have ultra low-power consumption is the core of the system. The board is equipped with 520 KB of SRAM and 448 KB of programmable ROM. It includes both WiFi and Bluetooth connectivity to accommodate a wide range of IoT use cases. The chip includes some cryptographic hardware acceleration support, namely for AES, SHA2, RSA algorithms. Unfortunately, it does not provide hardware acceleration for elliptic-curve cryptography (ECC) algorithms, which are the most burdensome ones in ABE. The board is natively supported by FreeRTOS [2]. FreeRTOS is a popular Operating System (OS) for embedded devices which supports a wide range of microcontrollers. It is written in the C language and provides support for multi-threaded programming. Compared with fully-fledged OSs, FreeRTOS lacks support for many advanced features and includes only a basic support for memory management and networking operations. A basic support for cryptographic operations is included, such as the popular wolfSSL library.

The RE-Mote [26] is a platform jointly designed by universities and industrial partners and produced by Zolertia, which targets industrial-grade design and ultra-low power consumption.

[2]FreeRTOS, https://www.freertos.org, accessed: 2019-12-06

Table 1: ESP32 and RE-Mote specifics

	ESP32	RE-Mote
CPU	Tensilica Xtensa dual-core LX6 microprocessor, operating at 240 MHz	ARM Cortex-M3, operating at 32 MHz
Radio	Wi-Fi: 802.11 b/g/n - Bluetooth: v4.2 BR/EDR and BLE	IEEE 802.15.4 (ISM 2.4-GHz and 863-950-MHz) & Zigbee
RAM	448 KB flash and 520 KB RAM	512 KB flash and 32 KB RAM

The board is equipped with the Texas Instruments CC2538 ARM Cortex-M3 System on Chip (SoC) working at 32 MHz and it can provide ECC hardware acceleration, i.e., it can accelerate point-scalar multiplication and point addition. Note that many other SoCs provide ECC hardware acceleration for standard elliptic curves only, which cannot be used for ABE, and thus they are useless for our aims. In contrast, the CC2538 SoC can accelerate the *generic* elliptic curve, including those suitable for ABE (*paring-friendly curves*), and thus it can be exploited to boost ABE operations. The RE-Mote board has native support for different OSs for IoT devices, including the Contiki-NG OS [3]. Compared to ESP32, RE-Mote offers a comparable programmable ROM size (512 KB), but it has a smaller SRAM size (32 KB), which makes storing of cryptography information on device challenging.

A summary of the specifications of the two adopted boards is reported in Table 1.

To carry out our performance evaluation, three existing libraries for Linux OS implementing the three considered ABE schemes have been ported to FreeRTOS and Contiki-NG, namely, the *libcelia* library [4] which implements the GPSW scheme, the *libbswabe* library [5] which implements the BSW scheme, and the *kpabe-yct14* library [6] which implements the YCT scheme. We configured all the three libraries to use pairing-friendly elliptic curves with embedding degree 2 and with effective security strength of 80 bits, which is equivalent to a 1024-bit RSA encryption. The libraries have been modified in order to suit the features offered by FreeRTOS and Contiki-NG. Specifically, the major modifications consisted into: (i) removing any usage of GLib, which is unavailable in both OS, and (ii) adapting the code to use the wolfSSL library on FreeRTOS and mbedTLS library on Contiki-NG[7] instead of the more popular OpenSSL, which is not supported on FreeRTOS and Contiki-NG.

5.3. Methodology

In order to assess the performance of the three considered ABE schemes on ESP32 and RE-Mote, three simple main programs, one for each library, have been developed to perform a sequence of operations. After the initial **Setup** algorithm which generates a master key and an encryption key, the program generates a decryption key with the **KeyGen** algorithm. Then, it creates a random 4-byte string that emulates the message to be transmitted. After that, the program encrypts the message and subsequently decrypts it. For each operation the program prints a message over the serial connection. This allows us to measure the time required to perform every operation. In order to measure the energy consumption, a high precision USB power meter is adopted. Specifically, we used the AVHzY USB Power Meter Tester[8], which supports automatic data collection from an attached PC and allows measurements with a resolution of 10^{-15} mWh.

[3]Contiki-NG, https://contiki-ng.org/, accessed: 2019-12-06

[4]Libcelia library: https://bit.ly/33lESZN, accessed: 2019-12-06

[5]Libbswabe library: https://bit.ly/2QRtEJV, accessed: 2019-12-06

[6]Kpabe-yct14 library: https://bit.ly/2DeJCWy, accessed: 2019-12-06

[7]Two different TLS/SSL libraries for the two ESP32 and RE-Mote implementations have been considered considering their availability on the FreeRTOS and Contiki-NG OSs, respectively.

[8]Power Meter Tester product page: https://goo.gl/vQDyac

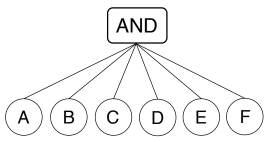


Fig. 4: Example of flat policy

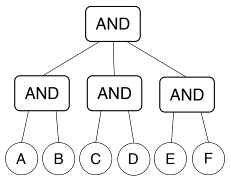


Fig. 5: Example of 3-level policy

The comparison between the log from the board and the readings from the power meter allowed us to measure the energy consumed for each specific operation.

To evaluate the three ABE schemes we considered the following metrics:

- *Encrypton/decryption time* (s), defined as the time required to execute an **Encrypt**/**Decrypt** algorithm.
- *Encrypton/decryption energy consumption* (mWh), defined as the overall energy consumed by the board to execute an **Encrypt**/**Decrypt** algorithm.

An increasing number of attributes, from 5 to 50, has been considered for encryption. Such a number represents the number of leaves in the policy tree for the CP-ABE scheme (BSW), or the size of the attribute set for the KP-ABE schemes (GPSW and YCT). These two quantities have not the same meaning, because the leaves of a policy tree represent the formal arguments of the policy, whereas the attribute set represents the actual arguments used to evaluate a policy. However, they both give a measure of the complexity of the access control rules involved in an application. For each experimental scenario, 10 independent replicas of the experiment have been executed. Our results report the average value of the measurements and the 95% confidence interval. Note that the number of replicas and the number of configurations in terms of attribute number considered in our experiments are limited. This is due to the fact that some steps for the execution of the experiments cannot be automated, thus greatly increasing the time required for their execution. Regarding the number of the replicas, 10 replicas should however be enough to get statistically sound averages, due to the small variability of the results. This is suggested also by the very small confidence interval, which are almost unnoticeable in the plots.

As mentioned in Section 2, the number of operations performed by the **Decrypt** algorithm grows up linearly with the number of visited leaves and the number of visited internal nodes (including the root). This means that policies with the same number of leaves but a different "shape" can perform differently. We shaped the access policies as *flat policies*, with a single internal node (the root) associated to an AND operator, and many child nodes, one for each attribute. Fig. 4 shows an example of flat policy. Flat policies represent the worst case for decryption algorithms. Indeed, with a flat policy the **Decrypt** algorithm is forced to visit all the leaves of the policy tree, and the leaf visit is generally the most expensive operation in decryption. In some experiments, we also used access policies shaped as *3-level policies*. A 3-level policy is semantically equivalent to a flat policy, but it has an additional intermediate level between the leaves and the root. The nodes in this intermediate level are AND operators, and they have no more than two leaves as children. Hence, the root has only $\lceil n/2 \rceil$ child nodes, where n is the number of leaves, opposed to n child nodes of a flat policy. Fig. 5 shows an example of 3-level policy, which is equivalent to the flat policy shown in Fig. 4. The 3-level policies are useful to test algorithms whose efficiency depends on the average number of children of internal nodes. Indeed, the internal nodes in a 3-level policy have in general fewer children than those in the equivalent flat policy.

5.3.1. Drawbacks of Flat and 3-Level Policies

Intuitively, results measured by using flat and 3-level policies give an *overestimation* of ABE resource consumption, since both types of policy are by definition the decryption's worst case scenario. However, in the real world, policies should be much more diverse, as they reflect the complexity of a human's access rights (KP-ABE), or they describe the vast range of entities that can access a single piece of information (CP-ABE). In practice, this variety and flexibility is rendered by building access policies using also OR gates. Such gates drastically diminish the number of nodes and leaves that need to be evaluated to satisfy a policy, therefore decreasing time, resources, and energy depleted by the sensors. Now, we anticipate the basic idea behind the *average-case scenario* (analyzed in Section 7), which allows us to estimate decryption performance more realistically. The average-case performance is measured by randomly generating many access policy/attribute set couples –ensuring for each couple that the generated attribute set satisfies the generated policy–, and then performing a decryption operation upon each couple. As in the worst case, we analyze the decryption time and the energy consumption needed to perform the decryption operation, so that we can compare the results with the ones obtained considering the worst-case scenario. To do this, we perform decryption over several thousands of different access policy/attribute set couples. The average-case scenario and the results achieved accordingly to it are thoroughly described in Section 7.

6. Experimental Results

In this section we present the results of our experiments. We first discuss the results obtained with the ESP32 boards, and then we analyze the results obtained with the RE-Mote boards.

6.1. Results with ESP32 Boards

6.1.1. Encryption Time and Energy Consumption

Figs. 6 and 7 show the encryption time and the encryption energy consumption of the three considered schemes, as a function of the number of involved attributes. Note that, as we said before, such "involved attributes" assume a different meaning in CP-ABE schemes (i.e., BSW) and in KP-ABE schemes (i.e., GPSW and YCT). In this figure and the following ones, we put them in the same X axis for the mere reason of saving space, but this should not be interpreted as a comparison between CP-ABE and KP-ABE schemes. As expected, the GPSW and the YCT schemes exhibit similar behavior, since they both perform one point-scalar multiplication for each attribute in the attribute set. The BSW scheme is quite expensive in encryption, in terms of both time and energy consumption. This is because it performs two point-scalar multiplications for each leaf in the policy tree, and it generates a random polynomial for each internal node.

Despite many previous papers on ABE [20, 13, 12] report an encryption time linear on the number of leaves for the BSW scheme, we actually experienced an over-linear time. This is mainly due to the random polynomial generation that the BSW scheme performs for each internal node of the policy tree. The complexity of generating such a random polynomial grows in an over-linear fashion with respect to the number of children of the internal node. To confirm this, we re-shaped the flat policy into an equivalent 3-level policy, in which each internal node has fewer children, and run an additional set of experiments with all the considered ABE schemes. In Figs. 6 and 7 we report the results of the BSW scheme with a 3-level policy. The results obtained with GPSW and YCT are omitted for the sake of brevity since both execution time and energy consumption for encryption do not deviate significantly from the results obtained with

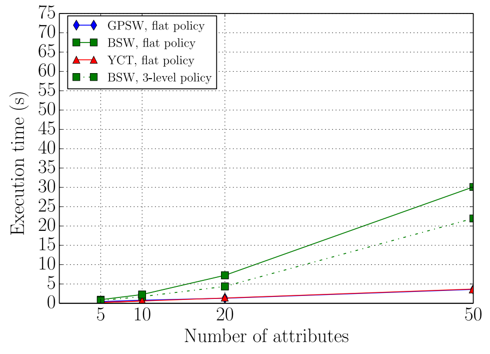


Fig. 6: Encryption time. ESP32

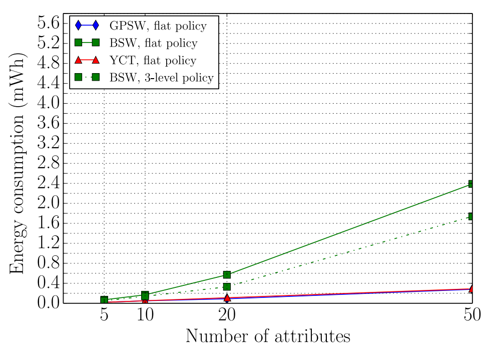


Fig. 7: Encryption energy consumption. ESP32

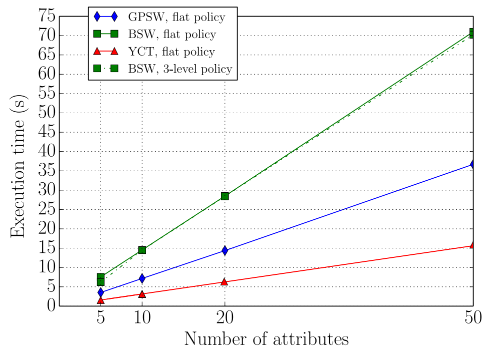


Fig. 8: Decryption time. ESP32

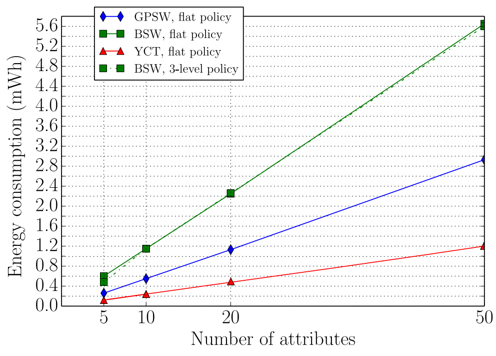


Fig. 9: Decryption energy consumption. ESP32

the flat policy. This is coherent with the fact that they do not generate random polynomials. As it can be seen, the encryption time and the encryption energy consumption of the BSW scheme with a 3-level policy decrease sensibly with respect to the flat policy. This result seems counter-intuitive since with a 3-level policy there are more polynomials to generate. However, it confirms that the over-linear cost of BSW encryption is due to the random polynomial generations, which grow in an over-linear fashion with respect to the number of children of each single internal node. This also suggests that shaping the policies in many levels is a good practice to improve the performance of BSW encryption. The reason why the over-linear behavior do not appear in other previous performance evaluation of ABE available in the literature [20, 13, 12] is that these papers include also the hashing of the attribute names within the encryption operation. In our performance evaluation, we do to include the hash operations because the hashes of the attributes' names can be precomputed, leading to a noticeable time saving.

6.1.2. Decryption Time and Energy Consumption

Regarding the decryption, Figs. 8 and 9 show the decryption time and energy consumption of the three considered schemes, as a function of the number of involved attributes, with flat policies. Interestingly, the YCT scheme is sensibly more efficient than the GPSW scheme in decryption. This is mainly because it is not based on bilinear pairings, but rather on point-scalar multiplications, which are less expensive. This suggests us that the GPSW and the YCT schemes are equivalent in those applications in which IoT devices only produce and encrypt data, for example in wireless sensor networks. Conversely, where also actuators are involved, and thus

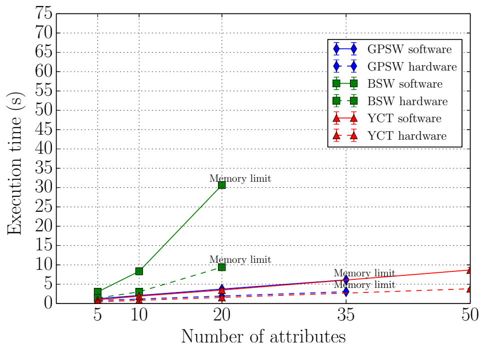


Fig. 10: Encryption time. RE-Mote

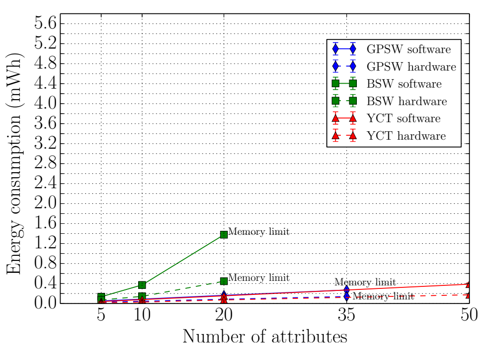


Fig. 11: Encryption energy consumption. RE-Mote

IoT devices are required to both encrypt and decrypt, the YCT scheme is significantly more convenient. As expected, the BSW is a quite expensive scheme also for decryption, because it performs two bilinear pairings for each visited leaf in the policy tree. No over-linear trend has been observed in BSW decryption, coherently with the fact that no random polynomials are generated.

Figs. 8 and 9 show also the decryption time and the decryption energy consumption of the BSW scheme both with flat and 3-level policies. As it can be seen, the decryption time and energy consumption seem independent of the shape of the policy tree. This confirms that shaping the policies in many levels is a good practice in the BSW scheme because it improves the performance of encryption, without decreasing the performance in decryption.

6.2. *Results with RE-Mote*

6.2.1. *Encryption Time and Energy Consumption*

Figs. 10 and 11 show, respectively, the encryption time and energy consumption of the three considered ABE schemes, vs. the number of involved attributes, when using flat policies. Differently from ESP32, RE-Mote provides ECC hardware acceleration, which can improve ABE performance. Therefore, we carried out experiments with and without ECC hardware acceleration. In the former case, the hardware support has been exploited to implement the ECC operations, in the latter one such operations have been implemented via software, as for the experiments carried out with ESP32. The results obtained with hardware acceleration are represented with dashed lines. As it can be seen, in some of the experiments the encryption operation failed, as a certain number of attributes is reached. Specifically the BSW scheme failed when more than 20 attributes were adopted, the GPSW scheme failed with more than 35 attributes. This is due to the limited amount of SRAM available on the RE-Mote platform, which was not sufficient to accommodate all the data structures required for the encryption operations.

As expected, the performance of the three ABE schemes exhibits the same trend as that obtained with ESP32 boards, both in terms of energy consumption end encryption time. By enabling hardware acceleration, both the encryption time and the corresponding energy consumption reduce significantly. For instance, if we consider the points of maximum reduction, the execution time is reduced by approximately 70% for BSW obtained with 20 attributes, by approximately 50% for GPSW obtained with 35 attributes and by approximately 55% for YCT obtained with 50 attributes. If we compare the results obtained with RE-Mote without ECC hardware acceleration and the results obtained with ESP32, we can notice that the encryption time and the encryption energy consumption are higher with RE-Mote. This is expected as the

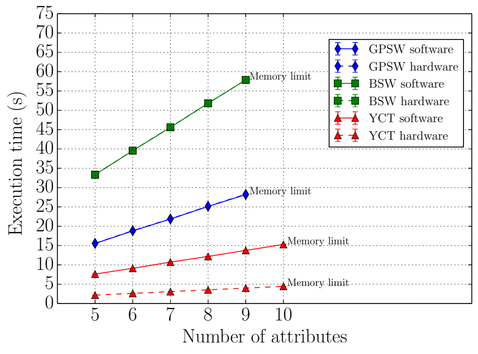


Fig. 12: Decryption time. RE-Mote

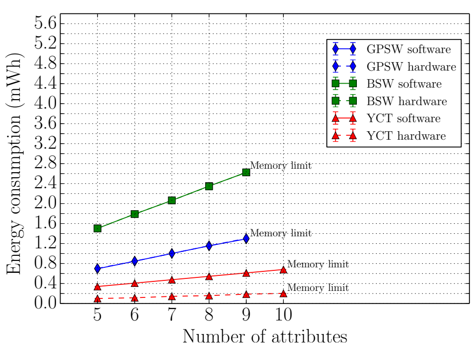


Fig. 13: Decryption energy consumption. RE-Mote

microcontroller installed in RE-Mote has a lower frequency than the one installed in ESP32, thus resulting in higher times (and consequently higher energy consumption) for encryption.

6.2.2. Decryption Time and Energy Consumption

Figs. 12 and 13 show the decryption time and the decryption energy consumption, respectively, for RE-Mote when using flat policies. The GPSW scheme does not benefit from hardware acceleration. As it can be seen, it does not obtain any performance gain via ECC hardware acceleration with respect to the software implementation of such operations. Instead, the YCT scheme improves its performance, i.e. it reduces the decryption time by 70% with hardware acceleration. The BSW scheme does not benefit from hardware acceleration either. This is because the BSW and the GPSW schemes use burdensome bilinear pairing operations in decryption, which are not accelerated in RE-Mote. Instead, the YCT scheme does not use bilinear pairings, but rather point-scalar multiplications, which are accelerated. Though some prototypes of bilinear pairing hardware accelerators have been developed in literature (see [27] for an example), none of them is commercially available to the best of authors' knowledge. Therefore, we can expect that decryption operations in pairing-based schemes (BSW and GPSW) do not benefit from hardware acceleration in any platform currently available in the market.

Also in decryption, the maximum number of attributes is limited by the constrained memory of RE-Mote. In particular, the number of attributes for which the decryption can be performed does not exceed 9 and 10, depending on the scheme. When compared to encryption, the maximum number of attributes is significantly reduced in decryption. This can be explained by taking into account the larger data structures required by decryption operations with all the considered ABE schemes.

When devices with low memory are considered, the maximum number of attributes that can be employed is bounded to the amount of memory available. This is further exacerbated when data decryption has to be implemented on an IoT device, e.g. in an actuator, as the maximum number of attributes is significantly lower in this case for all the schemes. With this respect, YCT is more efficient than GPSW, as the former allows to perform encryption up to 50 attributes. Furthermore, CP-ABE scheme can also be used by IoT constrained devices.

As regarding the hardware acceleration, in decryption it is only beneficial for the YCT scheme that is the only scheme that can actually exploit it, since it uses point-scalar multiplications.

6.3. *Battery Lifetime Analysis*

In order to better analyze the energy consumption that comes from to the adoption of ABE schemes and to obtain a key performance indicator that directly evaluates the feasibility of adopting ABE in a real IoT scenario, we also estimated the lifetime of a battery-powered sensor node, i.e. the time of operation for the sensor node to exhaust its battery. The battery lifetime resulting from Figs. 14 and 15 is a simplified measure that does not consider the energy consumption for all the operations performed by a sensor node. Nevertheless it can be useful as high-level comparison between different platforms. The evaluation is performed analytically by exploiting both the measurements from real experiments and the energy consumption data from the datasheet of ESP32 [25] and RE-Mote [26].

We consider a scenario in which a battery-powered sensor periodically collects a sample of a physical measurement, encrypts it using ABE, and then transmits the encrypted message over a wireless connection. For the sake of brevity, we assume that the device only produces and encrypts data (i.e., it is a sensor).

In our analysis, both the ESP32 and the RE-Mote boards are assumed to be powered by two AA 1.5 V batteries, which can provide 2.85 Ah each. The evaluation of the energy consumption of the sensors included: (i) the overall energy consumed by the sensors for data encryption; (ii) the energy consumed for the transmission of the encrypted message over WiFi (for ESP32) or IEEE 802.15.4 (for RE-Mote); (iii) the energy consumed in between two subsequent transmissions, assuming that the sensors remain idle. For the first component we used the measurements obtained from the real experiments, while for the latter two we exploited the power consumption values from the datasheet. To this aim, for ESP32 we assumed that the radio transceiver transmits data at 1 Mbps using DSS (Discrete Spread Spectrum) modulation, which results in a power consumption of 240 mW. RE-Mote, instead, transmits data at 250 Kbps, using again DSS modulation, and results in a power consumption of 72 mW. Moreover, we assumed that both the boards, when idle, enters a light-sleep mode, which results in a power consumption of 2.64 mW for ESP32 and of 1.8 mW for RE-Mote (according to the corresponding datasheets). Finally, a value of 60 seconds is considered for the sampling period.

Figs. 14 and 15 show the estimated battery lifetime, for the ESP32 and the RE-Mote boards respectively, expressed in days with respect to the number of involved attributes. As we said in the previous section, such "involved attributes" assume a different meaning in CP-ABE schemes (BSW) and in KP-ABE schemes (GPSW and YCT). In this figure and the following ones, we put them in the same X axis for the mere reason of saving space, but this should not be interpreted as a comparison between CP-ABE and KP-ABE schemes.

6.3.1. *Analysis with the ESP32 board*

In Fig. 14, the horizontal red line corresponding to 134 days represents the battery lifetime of an ESP32 sending data in the clear, i.e. without the cost of any encryption. With fairly small attribute sets, i.e. 10 attributes, a battery lifetime up to 62 days (54% decrease with respect to no-encryption scenario) can be obtained with the GPSW and YCT schemes. On the other hand, with 10-attribute access policies, the BSW scheme experiences a lifetime of around 50 days. As expected, as the number of involved attributes increases, the battery lifetime reduces proportionally to the energy consumption of all the three ABE scheme. With an attribute set of 50 attributes, the resulting battery lifetime of GPSW and YCT is 25 days. On the other hand, with 50-attribute policies BSW depletes the battery after few days of use, but we can see that the use of 3-level policies can improve -although slightly- the performance. The difference between the flat policy and the 3-level policy in BSW is due to that the over-linear cost of BSW encryption

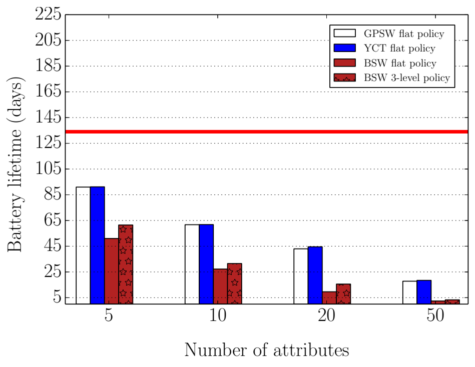


Fig. 14: Battery lifetime. ESP32

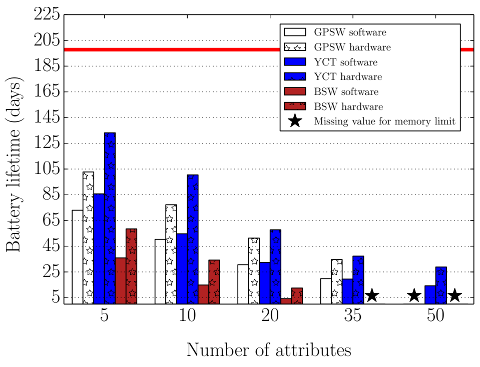


Fig. 15: Battery lifetime. RE-Mote

is due to the random polynomial generations, which grow in an over-linear fashion with respect to the number of children of each single internal node. It also suggests that shaping policies in many levels, while maintaining the same logical meaning, is a good practice to improve the performance.

6.3.2. Analysis with the RE-Mote board

Fig. 15 shows the battery duration of RE-Mote with flat policies. Again, the horizontal red line (about 198 days) represents the battery lifetime of a RE-Mote sensor sending data in the clear. It can be seen that, with 10 involved attributes, the battery lifetime is up to 101 days (49% decrease) with hardware-accelerated YCT, up to 78 days (61% decrease) with hardware-accelerated GPSW, and up to 35 days (82% decrease) with hardware-accelerated BSW. As for ESP32, the battery lifetime reduces proportionally with the number of attributes. If we compare the results with and without hardware acceleration, we can notice that hardware acceleration helps in improving the battery lifetime in all the three schemes. This is because the hardware support reduces the energy consumption for encryption. If we compare those results with the ones obtained with ESP32, we can note that the adoption of YCT and GPSW leads to the same battery lifetime on ESP32, but this is not true on RE-Mote. This can be explained by considering the different payload size of the encrypted message, which is larger with GPSW, and the different bitrates of the two wireless technologies. Due to the low bitrate of the IEEE 802.15.4, on RE-Mote a larger payload results in a longer time-on-air for each transmission, which impacts noticeably on the battery lifetime. On ESP32, instead, the increased payload has a negligible impact on the time-on-air due to the higher transmission bitrate of the WiFi, thus resulting in a comparable battery duration.

6.4. Final Remarks

In our performance evaluation we used two different IoT boards, namely ESP32 and RE-Mote. The results show that the CP-ABE scheme (BSW) is the most expensive for both of them in terms of execution time and energy consumption, and this applies either to encryption and decryption operations. If we compare the results of the ESP32 and the RE-Mote (Figs. 6-13) without the use of hardware acceleration (only available on RE-Mote), we can conclude that the power consumption of BSW is much higher with the RE-Mote board. This is due to the fact that their energy consumption is twice the one of the ESP32 board and they also have a less powerful microcontroller, thus encryption operations take more time to complete. On the other hand, the RE-Mote board can take advantage of the hardware acceleration for encryption operations, but in the case of the CP-ABE scheme, even though the gain obtained is significant (i.e. 70%), the power consumption is still in the same order of the ESP32 that cannot exploit hardware acceleration.

As regarding the KP-ABE schemes (GPSW and YCT), they have the same performance in encryption whereas in decryption the best performance is obtained with YCT. The hardware acceleration allows the RE-Mote to have encryption execution time and energy consumption in the same order of the ESP32, and, when possible, it should be exploited to fill the performance gap given by low frequency microcontrollers. For decryption operations with the RE-Mote board, instead, the hardware acceleration is beneficial only for YCT, since it uses point-scalar multiplications. It is worth to highlight that due to memory limit in the RE-Mote board, the decryption cannot be performed if the number of attributes is greater than 10. Thus the RE-Mote board should not be used if the attributed involved in the ABE policy are greater than 10.

To sum up, although the adoption of ABE has a noticeable cost in terms of energy consumption, the lifetime reduction can still be considered acceptable if we use small attribute sets in KP-ABE schemes, i.e. up to 20 attributes, and small policies in CP-ABE schemes, i.e. up to 5-attribute policies. When instead a higher number of attributes is considered, the resulting battery lifetime is shorten significantly down to a value that could be unacceptable in scenarios in which a frequent battery replacement is not feasible or desirable. In any case, the use of hardware ECC acceleration, which is available on some platforms like RE-Mote, helps in prolonging the battery lifetime too. We finally remark that some optimization techniques that are not considered in our analysis could be adopted to further improve the sensor battery lifetime. For example, it is often the case that all the pieces of data periodically transmitted by the sensor must be encrypted with the same attributes. In this case, the sensor can encrypt a single symmetric key (e.g., a 128-bit AES key) with ABE, store it on the untrusted broker, and then encrypt all the actual pieces of data with symmetric encryption, which introduces far less processing and bandwidth overhead with respect to ABE. This technique is used for instance by [28].

7. Average Decryption Performance Evaluation

In this section we explain the needs and motivations that led us to develop this evaluation framework as well as the main idea behind it, how it works, and what we want to achieve. This section is organized in the following way: in Section 7.1 we present the limitations of the main ABE benchmarks proposed in the literature; in Section 7.2 we show in detail the construction and use of our framework; in Section 7.3 we argue the plausibility of the synthetic policies; and finally in Section 7.4 we show the results obtained evaluating the average case with the proposed method.

7.1. Motivation: Worst Case vs. Average Case

A recurring problem while benchmarking ABE schemes is how to correctly evaluate the decryption performance. Indeed, while the encryption performance, i.e. time and energy consumption, is quite predictable given the number of attributes in the ciphertext, the decryption performance highly depends on unpredictable factors, namely the structure of the access policy and the attribute set. To evaluate the decryption performance, it is a common practice in the literature [20, 13, 12] to consider the worst case, which corresponds to the flat policy we used in this paper. However, this worst case is quite infrequent in practice: while it can be useful to validate the feasibility of adopting ABE in constrained devices by assessing its performance in the worst-case scenario, this configuration might be infrequent in a real scenario. Indeed, in a real scenario, access policies are supposed to have OR operators, and they are supposed to be structured in trees of many levels. However, it is extremely difficult to find large datasets related to *real* access policies and attribute sets used in companies and organizations, upon which evaluate the average case. This is more than understandable, since those are security-critical information belonging to the company. For this reasons, we propose a method that allows us to create *synthetic* access policies in order to measure the decryption performance of an ABE scheme in the average case. We believe this approach to be more realistic than testing them in the worst-case scenario, as done in many previous papers [20, 13, 12] and can be used to complement the experiments and provide a more comprehensive evaluation.

7.2. Framework Construction

Broadly speaking, the basic idea is to generate many random, but realistic, access policies and, for each generated policy, generate a random attribute set that fulfills such a policy. Then, for a KP-ABE scheme, we generate a decryption key associated with each of the access policies and a ciphertext associated with each of the attribute sets. For a CP-ABE scheme we do *vice versa*. Finally, we benchmark the decryption of the ABE scheme with the generated decryption keys and ciphertexts and we average the results, thus obtaining the decryption performance in the average case.

More in detail, we generate a random policy having n attributes according to the following steps.

1. We assign a conventional name to each of the n attributes, say A, B, etc., we build a leaf for each attribute, and we fill a *parent-less node set* $\mathcal{N}$ with all these leaves. Thorough the algorithm, the parent-less node set will contain the nodes of the policy we are creating that do not have a parent node yet. Since at this stage the random policy is still a collection of unstructured leaf nodes, the parent-less node set contains all of them. At the end of the algorithm, the random policy will be a tree and the parent-less node set will contain only the root.

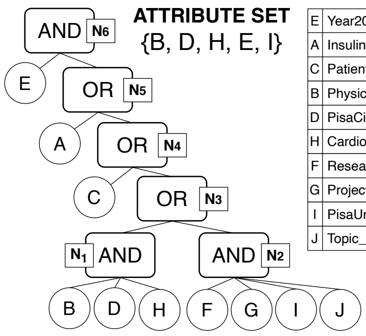
2. We randomly select each node in the parent-less node set with a given probability p_c, and we define the set of selected nodes as $\mathcal{N}_c$. If $\mathcal{N}_c$ contains less than two nodes we repeat the node selection procedure, until at least two nodes have been selected.

3. We build a new node having the nodes in $\mathcal{N}_c$ as children, and we assign a conventional name to it, say N_1. The nodes created in the successive iterations of the algorithm will be N_2, N_3, etc. We choose the Boolean operation associated to the new node to be AND with a given p_{AND} probability, or OR with $1 - p_{AND}$ probability.

4. We modify the parent-less node set by removing the nodes in $\mathcal{N}_c$ and adding the new node.

5. If the parent-less node set now contains only one node ($|\mathcal{N}| = 1$), we terminate the algorithm. The random policy is fully built. Otherwise, we repeat from Step 2.

The parameter $p_c \in (0, 1]$ influences how many children each node has on average, and thus the average height of the generated policy tree. The higher p_c is, the more nodes will be selected at each algorithm iteration, so the final policy will be more "flat". By choosing $p_c = p_{AND} = 1$, the method always generates flat policies, which represent the worst case for decryption.



E	Year2020
A	InsulinPumpID
C	PatientID
B	Physician
D	PisaCityHospital
H	CardiologyDept.
F	Researcher
G	Project_EuropeanSmartHealth
I	PisaUniversity
J	Topic_Diabetes

Fig. 16: Example of random policy and random fulfilling attribute set, and a possible attribute interpretation considering a medical scenario

After the random policy is generated, in order to perform a decryption operation we need to generate a random attribute set that fulfills the policy. We do it in the following way.

1. We select randomly each attribute in the n attributes used in the previous algorithm with probability p_a.
2. If the selected attributes fulfill the synthetic policy, we terminate the algorithm. The attribute set is fully built. Otherwise, we repeat from Step 1.

The parameter $p_a \in (0, 1]$ influences how many attributes are included in the attribute set. The higher p_a is, the larger the attribute set will be on average. Fig. 16 shows an example of random policy with 10 attributes and $p_c = p_{AND} = 0.5$, and a random attribute set that fulfills it, both created with the aforementioned method. Note that the created policy has many levels and different Boolean operators, which recalls real-world access policies written by system administrators. Note also that the created attribute set is suboptimal to decrypt, since for example the attribute set $\{A, E\}$ can fulfill the same policy by visiting fewer leaves and fewer internal nodes. Suboptimal attribute sets are extremely likely to appear in a real-world scenario.

7.3. Realism of the Synthetic Policies

The best way to corroborate the plausibility of synthetic access policies would be to analyze a dataset of real access policies. In this case we can extract and compare various parameters like the depth of the policy tree, the frequency of the AND/OR gate, the distributions of certain attributes, and many others. Unfortunately, as we said, companies and organizations are reluctant to disclose such precious information. However, we provide an example to support the idea that any synthetic policy can be mapped to a real world-policy. We do so by showing that it is possible to give a realistic meaning to the randomly generated policy of Fig. 16, considering the medical use case introduced in Section 4. Suppose that Pisa's city hospital provides a patient with a smart sensor which continuously monitor the glucose level in the blood. The same patient has a smart insulin pump that can be activated by the data received from such a smart sensor. Data produced by the smart sensor can be read also by the patient and by any physician that works at the Pisa's city hospital inside the cardiology department. Moreover, the patient gives also his consent to the processing of personal data to researchers belonging to Pisa University that participate in the "EuropeanSmartHealth" project in the "Diabetes" workpackage. Finally, for security reasons, keys are renewed every year, so each key has an attribute associated to the

	Pairing	Mod. exp.	Mod. mul.	Point-scalar mul.	Point add.
Time (SW)	3093 ms	44 ms	1 ms	167 ms	5 ms
Time (HW)	3093 ms	44 ms	1 ms	74 ms	2 ms
Energy (SW)	140 μWh	2 μWh	0.05 μWh	10 μWh	0.2 μWh
Energy (HW)	140 μWh	2 μWh	0.05 μWh	3 μWh	0.1 μWh
GPSW decryption	n_L	$n_L + n_N - 1$	$n_L - 1$	-	-
BSW decryption	$2n_L + 1$	$n_L + n_N - 1$	$n_L + 1$	-	-
YCT decryption	-	-	-	$2n_L + n_N - 1$	$n_L - 1$

Table 2: Processing time of basic crypto operations on RE-Mote, and number of basic crypto operations needed in decryption by the different schemes.

current year. Considering the CP-ABE paradigm, the policy that the smart sensor must enforce on its produced data has the shape of the random policy in Fig. 16 (refer to the table on the right of the figure for the attribute meanings). Furthermore, the randomly generated attribute set describes a physician working at the Pisa city hospital inside the cardiology department, who is also affiliated with the University of Pisa. This simple example demonstrate that the generated policies and attribute sets, though random, can have a realistic interpretation.

7.4. Simplified Method for Memory-Constrained Devices

The aforementioned method allows us to evaluate the decryption performance of generic ABE scheme in the average case. However, it requires to load the device under test with a high number of policies and associated attribute sets. This is hardly feasible with memory-constrained devices like ESP32 and RE-Mote. Fortunately, from our experience we noticed that the decryption efficiency of a scheme is analytically predictable given the number of basic cryptographic operations performed (e.g., pairings, modular exponentiations, etc.), and their individual processing time. We thus used a simplified method that involves first a benchmark of the basic cryptographic operations, secondly the random generation of many couples of access policies and attribute sets with the previously explained method, and finally the analytical computation of the decryption performance with every such random couple.

To better understand how the decryption performance can be analytically computed, remind that the decryption algorithm visits the policy tree in a bottom-up order, from the leaves to the root. The algorithm visits only those nodes that are strictly necessary to visit the root. For each visited node, the algorithm executes some basic elliptic-curve operations. In all the three examined schemes, the number of the various basic operations are expressible in terms of the number of visited leaves (n_L) and the number of visited internal nodes (n_N). Table 2 shows the processing times and the energy consumption of the basic cryptographic operations involved in decryption by the three schemes, measured on a RE-Mote with or without hardware acceleration. Each value has been obtained by averaging on 100 independent replicas of the experiment. The table shows also the number of different basic operations needed in decryption by the different schemes. Since we are interested in the difference between a performance evaluation based on the worst case and one based on the average case, we restrict our analysis on only one of the examined boards: namely the RE-Mote board. Performing a similar analysis on the ESP32 board is straightforward. We omit it for the sake of brevity. Basing on the numbers of Table 2, we applied the simplified method to benchmark the average decryption of the three schemes on RE-Mote.

Figs. 17 and 18 show respectively the resulting average decryption time and average energy consumption, compared to the worst-case ones, as they have been measured with the experiments

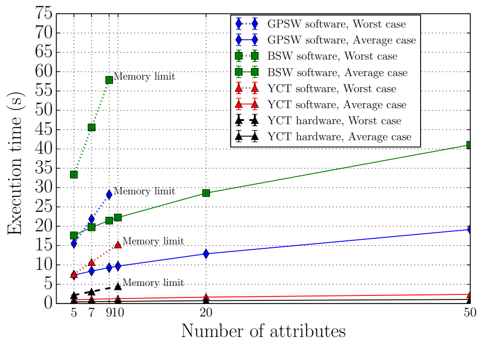


Fig. 17: Average- and worst-case decryption time on RE-Mote

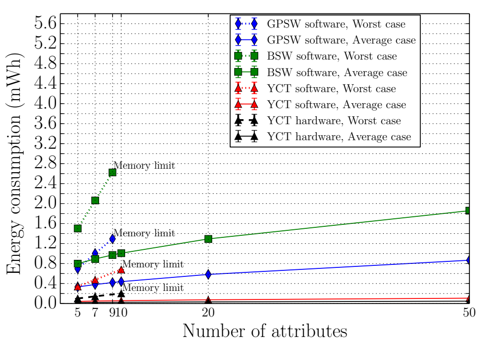


Fig. 18: Average- and worst-case decryption energy consumption on RE-Mote

of the previous sections (see Fig. 12). Each point of the plot relative to the average decryption time has been computed by averaging on 100,000 generated couples of random policies and attribute sets. Note that only considering flat policies, like practically all the literature about ABE scheme performance evaluation does [4, 12, 13, 20, 21], greatly overestimates the processing time of the average decryption operation. The average decryption time shows also a slight sublinear trend with respect to the number of attributes in the policy.

This is due to the presence of OR operators within the generated policies, which are instead absent in a flat policy. Indeed, in order to visit an OR node the decryption algorithm must first visit at least one of its children. The decryption algorithm will much probably visit the *most convenient* child, which is the one having less descendant leaves, to save time. As the attributes of the policy grow, also the number of OR operators grows, thus allowing the decryption algorithm to further save time by visiting the most convenient child each time.

We can conclude that always considering the worst-case decryption, the current literature significantly overestimates the processing time and energy.

However, such estimations are too harsh when considering the applicability of ABE in an IoT context. In fact, realistic policies can be shaped around the needs and the capability of the devices at disposal, as we showed before. We think that our framework is better in correctly estimating the feasibility of ABE in an IoT context, since it takes into account also the flexibility and the expressiveness of such a technique. Indeed, flexibility and expressiveness are two major features that are hard to quantify, but they have a great impact on the overall system. For example, with the GPSW scheme and 9-attribute policies, the energy consumption on RE-Mote of the average-case decryption is 67% less than that of the worst-case decryption. This shows that, if policies are well-thought and well-managed, ABE is far more performing than it is believed to be so far.

8. Conclusions

In this paper we carried out a performance evaluation of ABE in constrained IoT devices. Specifically, we implemented three representative ABE schemes and tested their performance on ESP32 and RE-Mote, two popular IoT rapid prototyping platforms. Our performance evaluation showed that ABE has a significant impact on the lifetime of battery-powered devices, especially when a high number of attributes (i.e., 20-50) is used in ciphertexts. However, if we assume to employ fewer attributes (up to 10) and to leverage hardware elliptic-curve cryptographic acceleration, which is present on some platforms (e.g. RE-Mote), then ABE can indeed be adopted on devices with very limited memory and computing power. We also obtained a significant, yet tolerable, battery lifetime reduction. To conclude, we presented a novel benchmark method that allows us to evaluate the average decryption performance, i.e. time and energy consumption, instead of the worst-case performance which is typically used by the literature. We exploited such a method to complete our evaluation by estimating the average decryption time and energy on RE-Mote. We showed that by always considering the worst-case decryption, the current literature significantly overestimates the processing time and energy.

Acknowledgments

This work was partially supported by the Italian Ministry of Education and Research (MIUR) in the framework of the CrossLab project (Departments of Excellence), and by the project EPI funded by European Unions Horizon 2020 research and innovation programmer under grant agreement No 826647. The authors would like to thank Benedetto Girgenti for his invaluable help in the implementation of the ABE policies and Iacopo Pacini for his help in carrying out the experiments.

References

[1] Y. Jiang, Combination of wearable sensors and internet of things and its application in sports rehabilitation, Computer Communications 150. `doi:https://doi.org/10.1016/j.comcom.2019.11.021`.

[2] X. Huang, Intelligent remote monitoring and manufacturing system of production line based on industrial internet of things, Computer Communications 150. `doi:https://doi.org/10.1016/j.comcom.2019.12.011`.

[3] Z. Shelby, K. Hartke, C. Bormann, The constrained application protocol (CoAP), Tech. rep. (2014).

[4] S. Zickau, D. Thatmann, A. Butyrtschik, I. Denisow, A. Küpper, Applied attribute-based encryption schemes, in: International ICIN ConferenceInnovations in Clouds, 2016.

[5] J. Qi, P. Yang, G. Min, O. Amft, F. Dong, L. Xu, Advanced internet of things for personalised healthcare systems: A survey, Pervasive and Mobile Computing, 2017.

[6] P. Picazo-Sanchez, J. E. Tapiador, P. Peris-Lopez, G. Suarez-Tangil, Secure publish-subscribe protocols for heterogeneous medical wireless body area networks, Sensors, 2014.

[7] M. Rasori, P. Perazzo, G. Dini, ABE-Cities: An attribute-based encryption system for smart cities, in: 2018 IEEE International Conference on Smart Computing.

[8] M. Rasori, P. Perazzo, G. Dini, A lightweight and scalable attribute-based encryption system for smart cities, Computer Communications 149.

[9] S. Sicari, A. Rizzardi, G. Dini, P. Perazzo, M. La Manna, A. Coen-Porisini, Attribute-based encryption and sticky policies for data access control in a smart home scenario: a comparison on networked smart object middleware, International Journal of Information Security.

[10] M. Qiu, K. Gai, B. Thuraisingham, L. Tao, H. Zhao, Proactive user-centric secure data scheme using attribute-based semantic access controls for mobile clouds in financial industry, Future Gener. Comput. Syst., 2018.

[11] R. Baden, A. Bender, N. Spring, B. Bhattacharjee, D. Starin, Persona: an online social network with user-defined privacy, SIGCOMM Comput. Commun. Rev, 2009.

[12] M. Ambrosin, A. Anzanpour, M. Conti, T. Dargahi, S. R. Moosavi, A. M. Rahmani, P. Liljeberg, On the feasibility of attribute-based encryption on internet of things devices, IEEE Micro, 2016.

[13] M. Ambrosin, M. Conti, T. Dargahi, On the feasibility of attribute-based encryption on smartphone devices, in: Workshop on IoT Challenges in Mobile and Industrial Systems, 2015.

[14] B. Girgenti, P. Perazzo, C. Vallati, F. Righetti, G. Dini, G. Anastasi, On the feasibility of attribute-based encryption on constrained iot devices for smart systems, in: 2019 IEEE International Conference on Smart Computing.

[15] A. Sahai, B. Waters, Fuzzy identity-based encryption, in: Advances in Cryptology – EUROCRYPT 2005, Springer Berlin Heidelberg.

[16] V. Goyal, O. Pandey, A. Sahai, B. Waters, Attribute-based encryption for fine-grained access control of encrypted data, in: 13th ACM Conference on Computer and Communications Security, 2006.

[17] J. Bethencourt, A. Sahai, B. Waters, Ciphertext-policy attribute-based encryption, in: 2007 IEEE Symposium on Security and Privacy (SP '07), 2007.

[18] X. Yao, Z. Chen, Y. Tian, A lightweight attribute-based encryption scheme for the internet of things, Future Gener. Comput. Syst., 2015.

[19] R. Ostrovsky, A. Sahai, B. Waters, Attribute-based encryption with non-monotonic access structures, in: 14th ACM Conference on Computer and Communications Security, 2007.

[20] X. Wang, J. Zhang, E. M. Schooler, M. Ion, Performance evaluation of attribute-based encryption: Toward data privacy in the IoT, in: 2014 IEEE International Conference on Communications (ICC).

[21] H. Kuehner, H. Hartenstein, Decentralized secure data sharing with attribute-based encryption: A resource consumption analysis, in: 4th ACM International Workshop on Security in Cloud Computing, 2016.

[22] U. Hunkeler, H. L. Truong, A. Stanford-Clark, MQTT-S a publish/subscribe protocol for wireless sensor networks, in: 3rd International Conference on Communication Systems Software and Middleware, 2008.

[23] M. Koster, A. Kernen, J. Jimenez, Publish-subscribe broker for the constrained application protocol (CoAP), Internet-Draft draft-ietf-core-coap-pubsub-06, Internet Engineering Task Force, work in Progress (Jan. 2019).

[24] D. Hankerson, A. J. Menezes, S. Vanstone, Guide to elliptic curve cryptography, Springer Science & Business Media, 2006.

[25] Espressif ESP32 datasheet, `https://bit.ly/2qW8yj1`, accessed: 2020-01-15.

[26] Zolertia RE-Mote datasheet, `https://bit.ly/2OkilYY`, accessed: 2020-01-15.

[27] A. Salman, W. Diehl, J. Kaps, A light-weight hardware/software co-design for pairing-based cryptography with low power and energy consumption, in: 2017 International Conference on Field Programmable Technology (ICFPT), 2017.

[28] M. La Manna, P. Perazzo, M. Rasori, G. Dini, fABElous: An attribute-based scheme for industrial internet of things, in: 2019 IEEE International Conference on Smart Computing.