On the Feasibility of Attribute-Based Encryption on Constrained IoT Devices for Smart Systems

Benedetto Girgenti, Pericle Perazzo, Carlo Vallati, Francesca Righetti, Gianluca Dini, Giuseppe Anastasi
Department of Information Engineering
University of Pisa
Pisa, Italy
name.surname@ing.unipi.it

***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. Such IoT devices generate an uninterrupted flow of information that can be transmitted through an untrusted network and stored on an untrusted infrastructure. The latter raises new security and privacy challenges that require novel cryptographic methods. Attribute-Based Encryption (ABE) is a new 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 assess the feasibility of the adoption of ABE in typical IoT constrained devices, characterized by limited capabilities in terms of computing, storage and power. Specifically, an implementation of three ABE schemes for ESP32, a low-cost popular platform to deploy IoT devices, is developed and evaluated in terms of encryption/decryption time and energy consumption. The performance evaluation shows that the adoption of ABE on constrained devices is feasible, although it has a cost that increases with the number of attributes. The analysis in particular highlights how ABE has a significant impact in the lifetime of battery-powered devices, which is impaired significantly when a high number of attributes is adopted.**

***Index Terms*—Internet of Things, constrained devices, security, attribute-based encryption, performance evaluation**

I. 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 allow us to enable novel functionalities, such as remote monitoring and big data collection for intelligent control and optimization.

The majority of IoT devices are 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 such devices. An example is the Constrained Application Protocol (CoAP) [1], 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 complex services on top of the functionalities offered by them. IoT systems are usually implemented as a multi-layered system 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 intermediate entities that implement functionalities like protocol translation, e.g. a gateway, or data dispatching, a broker. 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 their functionalities [2].

Attribute-Based Encryption (ABE) is a new type of public-key encryption that enforces a fine-grained access control on encrypted data based on flexible access policies. With ABE the source can encrypt data by using a set of attributes, which regulate the level of access to each single information.

In this paper we assess the feasibility of adopting ABE schemes in constrained IoT devices. Although, the feasibility of adopting ABE schemes in different contexts, such as fully-fledged embedded IoT systems [3] or smartphones [4] has been already assessed, at the best of authors knowledge, this is the first work analyzing the feasibility of implementing ABE in devices with limited capabilities in terms of memory and computational capabilities. Specifically, we implemented three ABE schemes for ESP32, a popular IoT rapid prototyping platform, and we evaluated the costs, in terms of time and energy consumption, of data encryption/decryption. Our

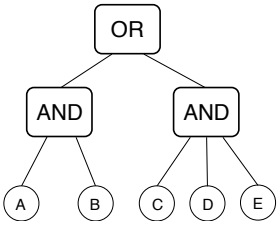

Fig. 1: Example of policy tree

performance evaluation shows that ABE can be adopted in constrained devices to ensure a fine-grained control on data access with a cost that increases with the number of attributes. Our analysis, in particular, highlights how ABE can have a significant impact on the lifetime of battery-powered devices, which can be reduced significantly when a number of attributes above 10 is employed.

The rest of the paper is organized as follows. Section II presents some background on ABE, Section III overviews related work, Section IV introduces a set of reference use cases and our threat model, Section V introduces our ABE implementations for the ESP32 platform and the results of the performance evaluation, while Section VI concludes the paper.

II. Attribute-Based Encryption

Attribute-Based Encryption (ABE) [5]–[8] is an emergent cryptography paradigm which allows 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 cryptography 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 decrypting party. An access policy is usually expressed by means of a tree (*policy tree*), in which leaves represent attributes, which are Boolean arguments, and internal nodes represent "AND" and "OR" Boolean operators. The "NOT" operator is not permitted in the majority of the ABE schemes, which thus allow only for *monotonic* access policies (see [9] for an exception). At decryption time, the access policy is evaluated with an *attribute set* as argument. The attribute set includes all the "true" attributes. An attribute not included in the attribute set is implicitly considered "false". If the access policy evaluates to true, then data will be decryptable, otherwise data will not be decryptable. Fig. 1 shows an example of policy tree which authorizes decryption only if the attributes A *and* B are true (i.e., they are included in the attribute set), *or* if the attributes C *and* D *and* E are true.

Two main ABE paradigms have emerged in the literature: *Key-Policy ABE* (*KP-ABE*) and *Ciphertext-Policy ABE* (*CP-ABE*). In the KP-ABE paradigm [6], the access policy is associated to the decryption key, and the attribute set is associated to the ciphertext. Thus, the attributes semantically describe data. KP-ABE paradigm can be viewed as a system that "tags" data with a set of attributes, and then it gives users a "ticket" that tells which data he can decrypt and which not. All the KP-ABE schemes implement at least the following four algorithms.

- $(MK, EK) = \mathbf{Setup}()$. This algorithm initializes the scheme and randomly creates and returns a pair of keys: a *master key* MK and an encryption key EK. The master key is kept secret by the authority in charge of generating decryption keys (*key authority*). The encryption key is made public and it is used to encrypt data.
- $(C) = \mathbf{Encrypt}(M, \gamma, EK)$. This algorithm encrypts a message M (*plaintext*) described by the attribute set γ, by means of the encryption key EK. It returns the encrypted message C (*ciphertext*), which embeds the attribute set.
- $(DK) = \mathbf{KeyGen}(\mathcal{T}, MK)$. This algorithm creates a new decryption key associated to the access policy $\mathcal{T}$, by means of the master key MK. It returns the decryption key DK, which embeds the given access policy.
- $(M$ or $\perp) = \mathbf{Decrypt}(C, DK)$. This algorithm decrypts a ciphertext M with the decryption key DK. It returns the original message M only if the access policy evaluates to true, otherwise it returns a null message $\perp$.

The number of operations performed by the **Encrypt** algorithm grows linearly with the size of the attribute set. The efficiency of the **Decrypt** algorithm is more complex to analyze, and it depends on the particular access policy and attribute set. In particular, the algorithm "visits" the policy tree in a bottom-up fashion, starting from a set of leaves associated to true attributes, and passing only by true internal nodes, up to the root. The algorithm does not have to visit all the nodes of the tree, but only a subset of them necessary to reach the root. If more than one subset of nodes permits the algorithm to reach the root, it chooses the minimal one. For example, supposing a decryption key with the policy tree of Fig. 1, and a ciphertext with an attribute set $\gamma = \{A, B, C, D, E\}$, the **Decrypt** algorithm can visit only the A and B leaf nodes, the AND node on the right, and the OR node at the root. The number of operations performed by the **Decrypt** algorithm grows linearly with the number of visited leaves and the number of visited internal nodes including the root.

In the Ciphertext-Policy ABE (CP-ABE) paradigm [7], the access policy is associated to the ciphertext, and the attribute set is associated to the decryption key. Thus, the attributes semantically describe a decrypting party. CP-ABE paradigm can be viewed as a system that tags each user with a set of attributes, and then associates an access policy to each data. CP-ABE schemes are generally considered more usable than KP-ABE ones [7]. This is because they realize a more intuitive access control approach, the same used for example by operating systems, where the policies on which user can access which resource are associated to the resources themselves. All the CP-ABE schemes implement at least the following four algorithms.

- $(MK, EK) = \mathbf{Setup}()$. This algorithm acts similarly to the one in the KP-ABE schemes.
- $(C) = \mathbf{Encrypt}(M, \mathcal{T}, EK)$. This algorithm encrypts a message M associated with the access policy $\mathcal{T}$, by

means of the encryption key EK. It returns the encrypted message C, which embeds the given access policy.

- $(DK) = \mathbf{KeyGen}(\gamma, MK)$. This algorithm creates a new decryption key associated with the attribute set γ, by means of the master key MK. It returns the decryption key DK, which embeds the given attribute set.
- $(M \text{ or } \perp) = \mathbf{Decrypt}(C, DK)$. This algorithm acts similarly to the one in the KP-ABE schemes.

The number of operations performed by the **Encrypt** algorithm grows linearly with the total number of leaves and the total number of internal nodes (including the root) in the policy tree. Similarly to KP-ABE schemes, the number of operations performed by the **Decrypt** algorithm grows linearly with the number of visited leaves and the number of visited internal nodes including the root.

III. Related Work

The application of ABE schemes to implement fine-grained access control and confidentiality has been already proposed in many different works, [10] [11] [12]. The feasibility of adopting different ABE schemes in practice, however, has been evaluated only in a few works.

In [4], the authors evaluated the feasibility of adopting ABE on smartphone devices. Specifically, an ABE library for the Android operating system is developed, then its performance is evaluated by means of real experiments. The results from the evaluation confirmed the possibility of using ABE on smartphone devices, showing that such devices have an acceptable amount of resources to implement ABE schemes and the resulting energy cost is acceptable.

In [3], instead, the authors assess the feasibility of using ABE in IoT devices. Conversely to ours, their work does not focus on constrained IoT devices but on more powerful platforms that have resources comparable with smartphone devices. Specifically the authors consider platforms like Raspberry Pi or Intel Edison Boards, which have enough resources to run a fully-fledged operating system. Experimental results demonstrate that exploiting ABE in such systems is feasible, although they also highlight that future works should focus on improve its efficiency.

In [13], 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. Their analysis highlighted how the resulting computation time can be acceptable in many use cases.

Compared with other works, at the best of authors' knowledge, this is the first one that evaluates the feasibility of exploiting various ABE schemes on very constrained devices. In particular, it is the first work considering low-cost embedded devices with less than 1 MB of RAM and a microcontroller.

IV. Use Case and Threat Model

The application of ABE schemes to implement fine-grained access control and confidentiality has been already considered

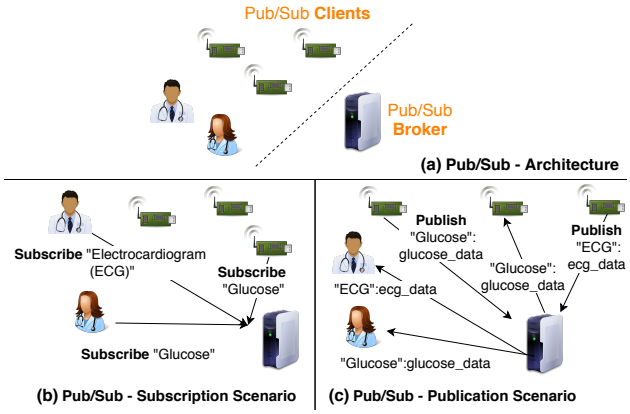

Fig. 2: Publish/subscribe architecture and mechanism

in literature in different and heterogeneous application scenarios, e.g. e-health [10], financial industry [11] and online social networks [12]. In all those systems, a fine-grained access control, like the one provided by ABE, is mandatory in order to differentiate access to information in a distributed manner. 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) [10] 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 constrained 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 must 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 information can be accessed by the users or devices of the system. For instance, the glucose sensor can be programmed to encrypt its data in order to allow only the insulin pump and the patient's physician to access the data.

In such systems, 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 [14] and the Constrained Application Protocol (CoAP) [15], 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/consume messages, e.g. periodic updates on a physical measurement. On the other side, we have a broker, which is a powerful 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 exploited 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 be sleeping, without compromising information availability.

In such architecture, ABE can ensure a fine-grained access control, thus allowing to manage multiple recipients. In addition to this, ABE can help in securing the system further. The core of the architecture is the broker, which is responsible for handling and dispatching the messages, such device, however, could be managed by untrusted third parties. The adoption of ABE could help to ensure that the broker has access only to metadata, e.g. the type of data or the topic, and not to the data itself, thus allowing the use of an untrusted broker.

A. Threat Model

We consider three types of adversary: (i) a simple eavesdropper; (ii) an adversary able to compromise the broker; (iii) an adversary able to compromise one or more IoT devices.

A simple eavesdropper cannot of course access data, since messages travel in an encrypted fashion and he does not own any decryption key. Notably, an adversary capable of compromising the broker cannot do any better, since the broker dispatches messages in an encrypted fashion, and it is not able of decrypting them. This is in contrast to "classic" security systems based on symmetric key encryption, in which the broker shares a key with each publisher and subscriber. In such systems, the broker is a single point of trust. If an adversary compromises it, the confidentiality of all the messages produced by the IoT devices is jeopardized. Conversely, ABE allows IoT devices to communicate in an end-to-end encrypted fashion, and at the same time it allows them to enforce access control policies over messages. Of course, a compromised broker can mount other attacks, for example it can simply drop some or all messages, thus causing a partial or complete denial of service. This type of attack may be counteracted by IoT intrusion-detection techniques (see for example [16]), and it falls outside the scope of the present paper.

An adversary capable of compromising one or more IoT devices can obtain different effects, depending on the type of compromised devices. If these devices only produce and encrypt messages, as it happens for example with sensors, their compromise does not endanger the confidentiality of any message. This is because such devices are unlikely to own a decryption key. This adversary can of course impersonate the sensor, and produce malicious encrypted messages. Again, this type of attack may be counteracted by IoT intrusion-detection approaches, and they fall outside the scope of the present paper. On the other hand, if the compromised devices also decrypt messages, as it happens for example with actuators, they must own a decryption key. Their compromise leads to the compromise of their decryption key, and this in turn endangers the confidentiality of *some* messages. Notably, the adversary can access only those messages that the compromised devices could access, since ABE enforces access control over data. In order to recover from a decryption key compromise, the publish/subscribe system must employ some rekeying mechanism, many of which exist in the literature [17]–[21].

V. Performance Evaluation

In this section we present the results of our experiments. Our goal is to assess the feasibility of adopting ABE schemes in constrained IoT devices, i.e. to analyze if ABE schemes can be implemented in devices characterized by scarcity of resources. In the followings, we first introduce our reference ABE schemes, our experimental settings, and then we analyze our obtained results.

A. Reference ABE Schemes

In this paper we focus on three notable ABE schemes:

1) the Goyal-Pandey-Sahai-Waters's scheme [6] (we will call it "GPSW scheme"), which has been the first proposed KP-ABE scheme in the literature;
2) the Bethencourt-Sahai-Waters scheme [7] ("BSW scheme"), which has been the first proposed CP-ABE scheme in the literature; and
3) the Yao-Chen-Tian scheme [8] ("YCT scheme"), which is a recent KP-ABE scheme for IoT devices focused on encryption and decryption efficiency.

In all the three considered schemes, the most expensive operation performed by the $\mathbf{Encrypt}$ algorithm is the *point-scalar multiplication*, which is an elliptic-cure cryptography operation (see [22] for details). In particular, the GPSW and the YCT schemes perform one point-scalar multiplication for each attribute in the attribute set, whereas the BSW scheme performs two point-scalar multiplications for each leaf in the policy tree. For each internal node of the policy tree, the BSW also 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. In both GPSW and BSW schemes, the most expensive operation performed by the $\mathbf{Decrypt}$ algorithm is the *bilinear pairing*, which is a quite expensive cryptographic operation (see [22] for details). Remind that the $\mathbf{Decrypt}$ 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, whereas BSW performs two bilinear pairings 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. The YCT scheme

performs one point-scalar multiplication for each visited leaf in the policy tree.

B. Experimental Settings

As an example of constrained IoT devices, in our experiments we exploited the ESP32 boards. ESP32[1] 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. At the core of the system we have a dual-core Xtensa LX6 microprocessor at 240 MHz designed to have ultra low-power consumption. The board is equipped with 520 KB of SRAM and 448 KB of programmable ROM. The board includes both WiFi and Bluetooth connectivity to accommodate a wide range of IoT use cases. The chip includes cryptographic hardware acceleration support, namely AES, SHA2, RSA, elliptic curve cryptography (ECC), random number generator (RNG).

The board is natively supported by the FreeRTOS operating system (OS)[2]. FreeRTOS is a popular OS for embedded devices which supports a wide range of microcontrollers. It is written in the C language and it 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[3].

To carry out our performance evaluation, three existing libraries for Linux OS implementing the three reference ABE schemes have been ported to FreeRTOS: 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. All the three libraries use elliptic curves 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. Specifically, the major modifications consisted into: (i) removing any usage of GLib, which is unavailable in FreeRTOS, and (ii) adapting the code to use the wolfSSL library instead of the more popular openSSL one, which is not supported by FreeRTOS. Our FreeRTOS porting of these three libraries is publicly available[7].

In order to assess the performance of the three ABE schemes on the ESP32, 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 which emulates the message to be transmitted. After that, the program encrypts the message and subsequentially decrypts it. For each operation the program prints a message over the serial connection, thus the time required to perform every operation can be measured. In order to measure the energy consumption, a high precision USB power meter is adopted. Specifically, we adopted the AVHzY USB Power Meter Tester[8], which supports automatic data collection from an attached PC. The comparison between the log from the ESP32 board and the power meter allowed us to measure the energy consumed for each specific operation.

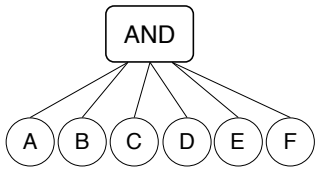

Fig. 3: Example of flat policy

The following metrics have been considered.

- *Encrypton/decryption time* (s), which measures the time required to execute an **Encrypt**/**Decrypt** algorithm.
- *Encrypton/decryption energy consumption* (mWh), which measures the overall energy consumed by the board to 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 size of the attribute set for the KP-ABE schemes (GPSW and YCT), or the number of leaves in the access policy for the CP-ABE scheme (BSW). Strictly speaking, these two quantities have not the same meaning, but they are usually compared in ABE performance evaluation papers [3], [4], [23] since they both give a measure of the complexity of the access policies involved in an application. For each experimental scenario, 5 independent replicas of the experiment have been executed. Our results report the average value of the measurements. The 95%-confidence intervals were also evaluated, however, they showed a negligible level of variability, therefore they are omitted in the results presented in this section.

As said in Section II, the number of operations performed by the **Decrypt** algorithm grows linearly with the num-bexxxxxxxxxxxr 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) which is an AND operator and many child nodes, one for each attribute. Fig. 3 shows an example of flat policy. With a flat policy, the **Decrypt** algorithm is always forced to visit all the leaves of the policy tree.

C. Results

Figs. 4 and 5 show the encryption time and the encryption energy consumption of the three schemes with respect to the number of involved attributes. Unfortunately, an internal bug of the kpabe-yct14 library prevented us to test the **Encrypt** algorithm of the YCT scheme with more than 10 attributes. Despite this, we believe the experimental results for 5 and 10

[1]https://www.espressif.com/en/products/hardware/esp32/overview
[2]https://www.freertos.org/
[3]https://www.wolfssl.com/
[4]https://github.com/gustybear/libcelia
[5]http://hms.isi.jhu.edu/acsc/cpabe/
[6]https://github.com/ikalchev/kpabe-yct14-cpp
[7]https://github.com/wellsaid

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

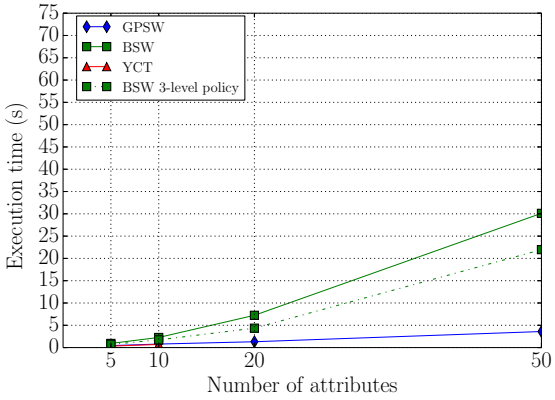

Fig. 4: Encryption time

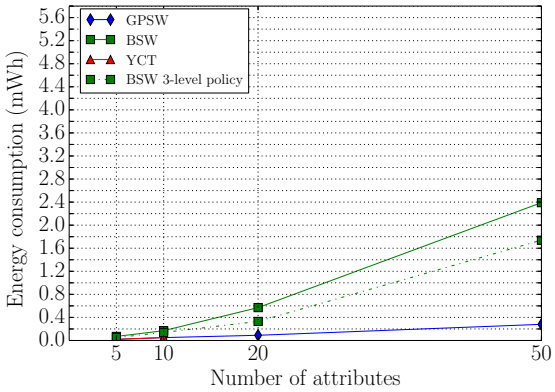

Fig. 5: Encryption energy consumption

attributes are interesting enough to be included in the paper, especially for the good decryption performances (see after).

As expected, the GPSW and the YCT schemes show similar performances, since they both perform one point-scalar multiplication for each attribute in the attribute set. On the other hand, the BSW scheme is the most expensive one 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. This suggests us that KP-ABE schemes are in general more efficient than CP-ABE ones. However, CP-ABE schemes are generally considered more usable than KP-ABE ones [7], because they associate the access policy to data, instead of keys. This corresponds to a more intuitive access control approach. This advantage in terms of usability is paid in terms of efficiency.

Despite many papers on ABE performance evaluation [3], [4], [23] report an encryption time linear on the number of leaves for the BSW scheme, we actually experienced an over-linear time. This is mainly ascribable to the random polynomial generation that the BSW scheme performs for each internal node of the policy tree. The complexity of generating such 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

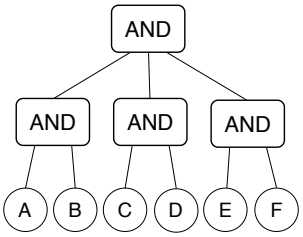

Fig. 6: Example of 3-level policy

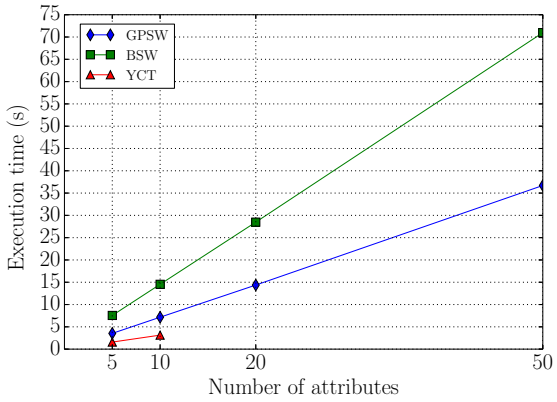

Fig. 7: Decryption time with flat policies

3-level policy, in which each internal node has fewer children. In a 3-level policy, there is 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 the flat policy. Fig. 6 shows an example of 3-layer policy, which is equivalent to the flat policy of Fig. 3. As it can be seen in Figs. 4 and 5, the encryption time and the encryption energy consumption of the BSW scheme with a 3-level policy decreases sensibly with respect to the flat policy. This 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 suggest us that shaping the policies in many levels is a good practice to improve the performances of the BSW encryption, and in general of the CP-ABE schemes.

Regarding decryption, Figs. 7 and 8 show the decryption time and the decryption energy consumption of the three schemes with respect to the number of involved attributes, with flat policies. As expected, the BSW is again the most 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 decryption. 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,

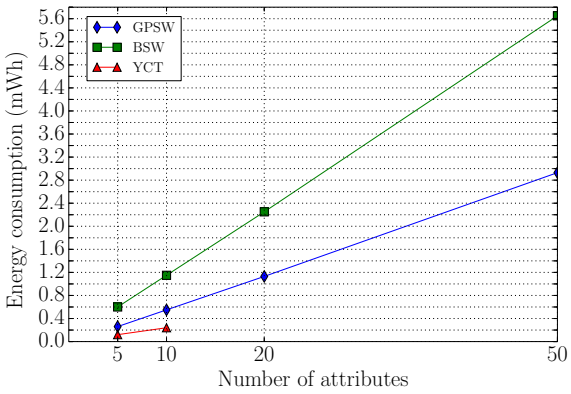

Fig. 8: Decryption energy consumption with flat policies

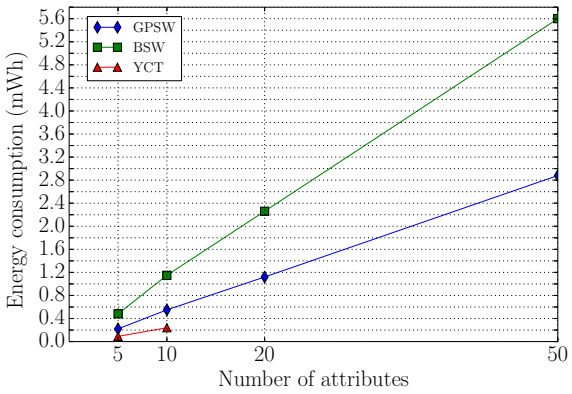

Fig. 10: Decryption energy consumption with 3-level policies

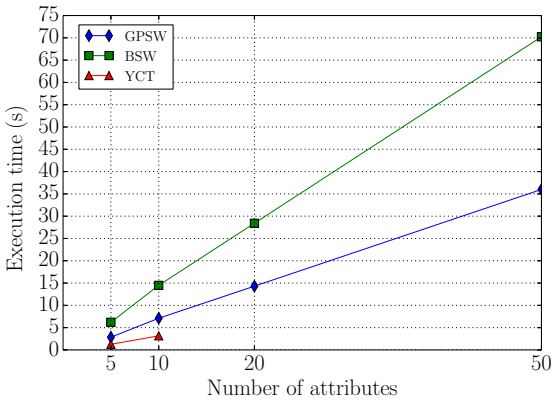

Fig. 9: Decryption time with 3-level policies

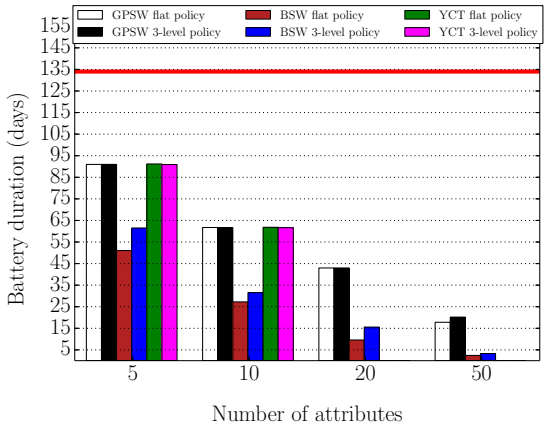

Fig. 11: Battery lifetime

where also actuators are involved, and thus IoT devices are required to both encrypt and decrypt data, the YCT scheme is sensibly more convenient.

Finally, Figs. 9 and 10 show the decryption time and the decryption energy consumption of the three schemes with respect to the number of involved attributes, with 3-level policies. In all the three schemes, decryption efficiency seems independent from the shape of the policy tree. This confirms us that shaping the policies in many levels is a good practice in CP-ABE schemes because it improves the performance of encryption, without decreasing the performance in decryption. On the other hand, it seems not to influence the performances for KP-ABE schemes. Indeed, for both the GPSW and the YCT schemes, the efficiency is independent from the shape of the policy trees, both in encryption and decryption.

D. Battery Lifetime Analysis

In order to better analyze the consumed energy 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 evaluation is performed analytically by exploiting both the measurements from real experiments and the energy consumption data from the datasheet of the ESP32 [9]. A scenario in which a battery-powered sensor collects samples of a physical measurement with a given period, encrypts the measurement using ABE, and then transmits the encrypted message over a WiFi connection is considered. For the sake of brevity, we only consider the scenario in which the device only produces and encrypts data, i.e., it is a sensor.

In our analysis, the sensor is assumed to be powered by two AA 1.5V batteries, which can provide 2.85 Ah each. The overall evaluation of the energy consumption of the sensor included: (i) the overall energy consumed by the sensor for data encryption; (ii) the energy consumed for the transmission of the encrypted message over WiFi; (iii) the energy consumed in between two subsequent encryption/transmission in which the sensor remains 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, we assumed that the radio transceiver transmits data at 1 Mbps using DSSS as modulation, which results in a power consumption of 792 mW. Moreover, we assumed that the board, when idle, enters into light-sleep mode, which results in a power consumption of 2.64 mW. Finally, a value of 60 seconds is considered for the

[9]https://goo.gl/BHv51B

measurement period.

In Fig. 11 the resulting battery duration is reported in days for every combination of attributes, ABE schemes and number of attributes. The horizontal red line in correspondence to 134 days represents the battery lifetime of a sensor sending data in clear, i.e. without the cost of any encryption. In the scenario with the lowest number of attributes, i.e. 5 attributes, a battery lifetime up to 90 days (32% decrease with respect to no-encryption scenario) can be obtained with the schemes GPSW and YCT, the ones resulting in the lowest energy consumption. With the BSW scheme, instead, a lower lifetime is obtained, approximately around 50 days. As expected, as the number of attributes increases the battery duration reduces proportionally to the energy consumption of each ABE scheme. As the number of attributes increases up to 50 the resulting battery lifetime goes below the month. Specifically, it results in 25 days with GPSW, while it drops significantly to some days with BSW.

Although the adoption of ABE encryption has a noticeable cost in terms of energy consumption, the lifetime reduction can still be considered acceptable when a low number of attributes and an energy-efficient scheme are adopted, i.e. 5/10 attributes and the GPSW scheme. When, instead a higher number of attributes is adopted, 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.

VI. Conclusions

In this paper, the feasibility of adopting ABE schemes in constrained IoT devices was analyzed by means of real experiments. Existing ABE implementations were ported to the ESP32 IoT platform and the performance of each scheme was analyzed to assess the feasibility of its adoption in a real implementation. The results of our experiments highlighted how ABE encryption/decryption operations have a significant execution time and energy cost on constrained IoT devices characterized by limited memory and processing. Such costs, however, are still bearable when a limited number of attributes is adopted. An analytical analysis of the battery lifetime was also carried out to offer an estimate of the impact of ABE operations on battery powered devices. The analysis highlighted that ABE operations result in a noticeable reduction of the battery lifetime, which, however, can be considered acceptable when the number of attributes is low. As future work, we plan to assess the performance of ABE also in other IoT platforms, considering boards that are equipped with hardware support for the execution of some of the cryptographic operations required by ABE schemes.

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 PRA_2018_81 “Wearable sensor systems: personalized analysis and data security in healthcare” funded by the University of Pisa.

References

[1] Z. Shelby, K. Hartke, and C. Bormann, “The constrained application protocol (CoAP),” Tech. Rep., 2014.

[2] S. Zickau, D. Thatmann, A. Butyrtschik, I. Denisow, and A. Küpper, “Applied attribute-based encryption schemes.”

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

[4] M. Ambrosin, M. Conti, and T. Dargahi, “On the feasibility of attribute-based encryption on smartphone devices,” in *Proceedings of the 2015 Workshop on IoT Challenges in Mobile and Industrial Systems*, ser. IoT-Sys ’15. New York, NY, USA: ACM, 2015.

[5] A. Sahai and B. Waters, “Fuzzy identity-based encryption,” in *Advances in Cryptology – EUROCRYPT 2005*, R. Cramer, Ed. Springer Berlin Heidelberg, 2005.

[6] V. Goyal, O. Pandey, A. Sahai, and B. Waters, “Attribute-based encryption for fine-grained access control of encrypted data,” in *Proceedings of the 13th ACM Conference on Computer and Communications Security*, ser. CCS ’06. New York, NY, USA: ACM, 2006.

[7] J. Bethencourt, A. Sahai, and B. Waters, “Ciphertext-policy attribute-based encryption,” in *2007 IEEE Symposium on Security and Privacy (SP ’07)*, May 2007.

[8] X. Yao, Z. Chen, and Y. Tian, “A lightweight attribute-based encryption scheme for the internet of things,” *Future Gener. Comput. Syst.*, vol. 49, no. C, Aug. 2015.

[9] R. Ostrovsky, A. Sahai, and B. Waters, “Attribute-based encryption with non-monotonic access structures,” in *Proceedings of the 14th ACM Conference on Computer and Communications Security*, ser. CCS ’07, New York, NY, USA, 2007.

[10] P. Picazo-Sanchez, J. E. Tapiador, P. Peris-Lopez, and G. Suarez-Tangil, “Secure publish-subscribe protocols for heterogeneous medical wireless body area networks,” *Sensors*, vol. 14, no. 12, 2014.

[11] M. Qiu, K. Gai, B. Thuraisingham, L. Tao, and H. Zhao, “Proactive user-centric secure data scheme using attribute-based semantic access controls for mobile clouds in financial industry,” *Future Gener. Comput. Syst.*, vol. 80, no. C, Mar. 2018.

[12] R. Baden, A. Bender, N. Spring, B. Bhattacharjee, and D. Starin, “Persona: an online social network with user-defined privacy,” *SIGCOMM Comput. Commun. Rev.*, vol. 39, August 2009.

[13] H. Kuehner and H. Hartenstein, “Decentralized secure data sharing with attribute-based encryption: A resource consumption analysis,” in *Proceedings of the 4th ACM International Workshop on Security in Cloud Computing*, ser. SCC ’16, New York, NY, USA, 2016.

[14] U. Hunkeler, H. L. Truong, and A. Stanford-Clark, “MQTT-S a publish/subscribe protocol for wireless sensor networks,” in *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE ’08)*, Jan 2008.

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

[16] S. Raza, L. Wallgren, and T. Voigt, “SVELTE: Real-time intrusion detection in the Internet of Things,” *Ad Hoc Networks*, 2013.

[17] S. Yu, C. Wang, K. Ren, and W. Lou, “Achieving secure, scalable, and fine-grained data access control in cloud computing,” in *2010 Proceedings IEEE INFOCOM*, March 2010.

[18] S. Yu, K. Ren, and W. Lou, “Fdac: Toward fine-grained distributed data access control in wireless sensor networks,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 4, April 2011.

[19] J. Li, X. Huang, J. Li, X. Chen, and Y. Xiang, “Securely outsourcing attribute-based encryption with checkability,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 8, pp. 2201–2210, Aug 2014.

[20] M. Rasori, P. Perazzo, and G. Dini, “ABE-Cities: An attribute-based encryption system for smart cities,” in *2018 IEEE International Conference on Smart Computing (SMARTCOMP)*, June 2018.

[21] M. La Manna, P. Perazzo, M. Rasori, and G. Dini, “fABElous: An attribute-based scheme for industrial internet of things,” in *Smart Computing (SMARTCOMP), 2019 International Conference on (to appear)*. IEEE, 2019.

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

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