A Survey on Attribute-Based Encryption Schemes Suitable for the Internet of Things

Marco Rasori, Michele La Manna, Pericle Perazzo, and Gianluca Dini

***Abstract*—The Internet of Things (IoT) is an information service paradigm based on the integration of smart objects, mobile devices, and computers via the Internet. IoT technologies are key enablers for a multitude of applications in diverse fields, such as digital health, smart city, industrial automation, and supply chain. This raises new security and privacy challenges that can be addressed by advanced cryptographic methods. One of the most prominent is attribute-based encryption (ABE), which allows one to encrypt data while enforcing fine-grained access control on it. ABE is advantageous in many IoT applications since it allows data to be safely stored on untrusted storage, such as third-party cloud servers, hackable publish-subscribe brokers, physically accessible sensors, etc. This article surveys the ABE literature proposing schemes and solutions that are best suited for IoT applications. To do so, it first identifies three performance indicators that are key in IoT, namely, the data producer CPU efficiency, the data producer bandwidth efficiency, and the key authority bandwidth efficiency. Then, it analyzes only those schemes that are promising from the point of view of one or more indicators and, therefore, more applicable in typical IoT applications. As a further contribution, this article selects a subset of representative schemes and assesses their efficiency by thorough simulations. Such simulations show that no scheme excels in all three performance indicators at once, but some simultaneously perform well in two indicators.**

***Index Terms*—Access control, attribute-based encryption (ABE), data-centric security, Internet of Things (IoT).**

I. Introduction

IN THE last decade, Internet of Things (IoT) technologies have been growing and evolving very rapidly, and a myriad of services spanning across plenty of sectors is nowadays based on them. In a typical use case within the digital healthcare field, medical data can be acquired by body sensors monitoring patient vital signs. Such data may be then transmitted to some storage server and made available to different entities/parties, such as doctors, nurses, researchers, etc., with different access privileges, for different purposes. Other examples of IoT applications are within the fields of wireless sensor networks [1], industrial automation [2], [3], smart grid [4], smart city [5], smart health [6], home automation [7], smart vehicles and transportation [8], [9], supply chain [10], and so on [11]. IoT applications generally produce enormous amounts of data, which are often stored permanently or temporarily in untrusted storage, for example, cloud or edge servers managed by third parties and exposed to the Internet. Depending on the type of service, such data may include valuable, privacy-sensitive, or business-critical information. In these cases, protecting the confidentiality of data against unauthorized access is important. Unfortunately, the classic approach of protecting the communication links between the IoT devices and the cloud/edge servers, for example with the TLS protocol, is not enough. Indeed, it does not protect against data breaches on the cloud/edge servers, which could happen due to several reasons ranging from internal threats to software or hardware vulnerabilities [12], [13]. There are plenty of reasons for storing data on cloud/edge servers in the encrypted rather than cleartext form.

A promising encryption paradigm in the branch of functional encryption is attribute-based encryption (ABE) [14]. ABE, by its very nature, enforces fine-grained access control on encrypted data. The purpose of this survey is to review the current literature and filter only ABE schemes suitable for the IoT. This could help researchers and industries to find the right ABE scheme for their IoT scenario of interest. To achieve this objective, we survey ABE schemes under the point of view of specific features that are desirable in an IoT context. In particular, we identify three key performance indicators (KPIs), namely, the *data producer CPU efficiency*, the *data producer bandwidth efficiency*, and the *key authority bandwidth efficiency*. We believe these three indicators should be considered first when choosing an ABE scheme suitable for a particular IoT application since they give immediate advantages in the typical IoT use case. We also examine three accessory performance indicators (APIs) that might still be important for some applications but have less priority in IoT: *data producer storage efficiency*, *data consumer CPU efficiency*, and *data consumer bandwidth efficiency*. Note that our survey is intended to be *selective*, in the sense that we focus only on those schemes that are promising from the point of view of

Manuscript received July 30, 2021; revised December 3, 2021; accepted February 13, 2022. Date of publication February 25, 2022; date of current version May 23, 2022. This work was supported in part by the Italian Ministry of Education, University and Research (MIUR) in the Framework of the CrossLab Project (Departments of Excellence); in part by the European Union within the Horizon 2020 Research and Innovation Programme "European Processor Initiative—Specific Grant Agreement 1" under Grant 826647 and "European Processor Initiative—Specific Grant Agreement 2" under Grant 101036168; and in part by the University of Pisa within the Research Project PRA_2020_92 "Quantum Computing, Technologies and Applications." *(Corresponding author: Marco Rasori.)*

Marco Rasori is with the Institute of Informatics and Telematics, National Research Council, 56124 Pisa, Italy (e-mail: marco.rasori@iit.cnr.it).

Michele La Manna is with the Department of Information Engineering, University of Pisa, 56126 Pisa, Italy, and also with the Department of Information Engineering, University of Florence, 50121 Florence, Italy (e-mail: michele.lamanna@unifi.it).

Pericle Perazzo and Gianluca Dini are with the Department of Information Engineering, University of Pisa, 56126 Pisa, Italy (e-mail: pericle.perazzo@unipi.it; gianluca.dini@unipi.it).

Digital Object Identifier 10.1109/JIOT.2022.3154039

one or more KPIs or APIs and, thus, are more likely to be employed in typical IoT applications. Moreover, we select only schemes provided with formal security proof. By doing this, we intentionally leave out many schemes whose potential in IoT is not sufficient or whose security is not sufficiently proved. Our selective survey enables researchers and practitioners to get up to speed quickly on the characteristics of the different ABE schemes present in the literature and understand their suitability for the IoT application of their interest, obviating the need to read many cryptographic papers. As a further contribution, we employ thorough simulations to assess the efficiency of a subset of representative schemes. Basing on such simulations, we observe that no scheme excels in all three KPIs at once, but some simultaneously perform well in two indicators.

In the past, other surveys on ABE [15]–[23] have been published, but they differ from ours for various reasons. Some of the existing surveys [18], [21], [22] are outdated; As ABE is a trending topic, a lot of new and more sophisticated schemes are proposed every year. Some surveys do not carry out exhaustive research on ABE, which means that they limit their contribution analyzing schemes with specific characteristics. For example, Al-Dahhan *et al.* [16] and Liu *et al.* [19] surveyed only revocable ABE schemes, while Moffat *et al.* [20] do not consider revocable schemes. Recent surveys [17], [20], [24] do not explore novel strategies, e.g., ABE schemes with asymmetric pairings, and they do not quantitatively estimate the KPIs of the suitable schemes through experiments. In the present work, we do not survey lattice-based ABE schemes, for example [25], [26]: although they are interesting from a post-quantum perspective, they seem to be too burdensome for the class of constrained IoT devices that we consider in this article.

The remainder of this article is organized as follows. Section II presents the methods used to select the articles surveyed in this work. Section III introduces the typical IoT architecture on which it is possible and fruitful to apply ABE techniques, introduces the KPIs, and advocates why they deserve priority. Section IV introduces the basic concepts of pairing-based cryptography and ABE. Sections V–VII survey the different strategies employed in the literature to improve the KPIs. In particular, Section V focuses on data producer CPU efficiency, Section VI focuses on key authority bandwidth efficiency, and Section VII focuses on data producer bandwidth efficiency. In Section VIII, we simulate some promising ABE schemes that excel in at least one KPI, and we discuss the results. Section IX surveys the different strategies employed in the literature to improve the APIs. Finally, Section X concludes this article and discusses future works directions.

II. Methodology

The core idea of this survey is to selectively review the present literature on ABE schemes and solutions best suited for IoT applications. Thus, from our previous experience and knowledge of the topic, we first identified three KPIs, namely, the data producer CPU efficiency, the data producer bandwidth efficiency, and the key authority bandwidth efficiency, which, in our perspective, are the most advantageous properties in the typical IoT use case employing ABE. Moreover, we determined effective *strategies* that influence each KPI. For instance, the producer bandwidth efficiency is affected by factors such as the ciphertext size, whereas the key authority bandwidth efficiency depends upon revocation mechanisms. Second, we also identified and examined APIs, namely, the data producer storage efficiency, the data consumer CPU efficiency, and the data consumer bandwidth efficiency. APIs can be desirable properties in some IoT applications employing ABE, but they usually impact less on the overall performance and are less critical compared to the KPIs. A more detailed description of KPIs/APIs relevance will be given in each corresponding section. KPIs and APIs selection was essential for narrowing down the literature research and define the keywords to be used.

The search of articles herein presented was conducted on several websites and scientific databases, namely, Google Scholar, IEEE Xplore, ACM digital library, Springer Link, Elsevier Scopus, and DBLP using “ABE” OR “attribute-based encryption” as main keyword, along with others, such as “IoT” OR “Internet of Things,” and keywords related to the various strategies to improve the performance indicators, such as “ciphertext size,” “revocation,” “encryption outsourcing,” “decryption outsourcing,” etc.

An initial selection of potentially relevant papers was made by excluding those articles whose titles and/or abstracts showed contents unrelatable to the topic of this survey. For example, we excluded articles that do not present new ABE schemes, but only applications of existing schemes. Then, a full-text screening of the remaining articles followed to evaluate their relevance. Furthermore, the reference list of each selected article was screened to identify additional articles. The criteria of exclusion applied were: 1) articles presented obsolete schemes compared to others and 2) articles did not include a security proof.

Our selection methodology did not exclude *a priori* ABE schemes with peculiar additional features, such as *puncturable*, *traceable/accountable*, *policy hiding*, and *post-quantum secure* schemes. However, most of these schemes turned out to be unsuitable for IoT scenarios as these features often add excessive complexity and performance reduction. Few exceptions exist [27] that we analyze in the following.

III. ABE in IoT

The typical IoT architecture (Fig. 1) includes many *data producers* (or simply *producers*), which produce information, many *data consumers* (or simply *consumers*), which consume such information, and some *data storage*, on which information is either temporarily or permanently stored.

Data producers are typically sensing devices that measure physical quantities or detect events from the environment. They are often constrained and battery powered and have limited computing and connectivity capacities. In some cases, data producers are more resourceful devices, for example, single-board computers (e.g., Raspberry Pi) or mobile devices. Data consumers are typically devices that display information

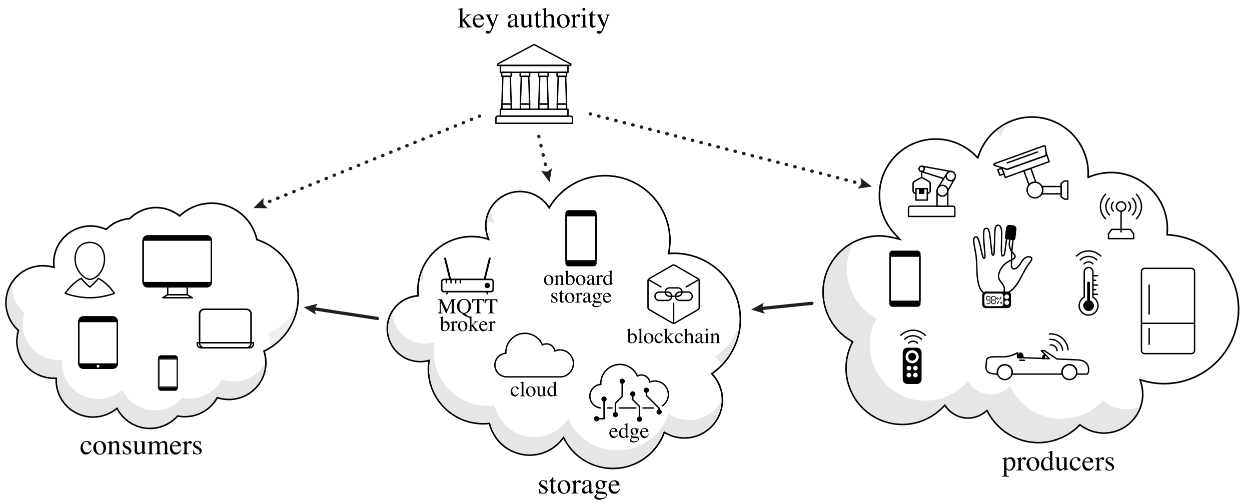


Fig. 1. ABE architecture.

to users, or they are actuators that automatically undertake actions based on such information. They can be smartphones, smartwatches, tablets, or even full-fledged computers with higher computing and connectivity capabilities than the average data producer. In some cases, data consumers are more constrained devices, for example, single-board computers or even battery-powered actuators. Data storage can be implemented in various ways, including at one extreme high-end mainframes providing cloud services to subscribers, and at the other extreme the data producers themselves in case they temporarily store sensed data locally instead of transmitting it immediately (*onboard storage*). Between these two extremes, data storage can be edge nodes, MQTT broker devices, etc. In addition, many recent IoT systems store data in blockchain data structures, such as Ethereum [28], [29]. The typical ABE scheme in the literature considers all data storage as *untrusted* for several reasons. Indeed, cloud servers are often managed by third-party companies based in foreign countries. Furthermore, edge servers and cloud servers are Internet-connected and constantly exposed to cyberattacks, both software and hardware [12], [13]. Onboard storage is considered untrusted because producers are often easy to hack or physically accessible and unattended, e.g., in wireless sensor networks. Finally, data are inherently public in the case of on-blockchain storage, so it must be encrypted to preserve its secrecy [30].

For all the reasons set out above, it is essential to encrypt the data when at rest in the data storage. ABE technology is very effective for all those systems required to preserve data confidentiality and enforce an access control mechanism. Some ABE schemes in the literature consider data storage as *semitrusted*, which may assume different meanings but generally implies that data storage can perform *some* malicious actions (e.g., attempting to decrypt data), but not others (e.g., actively sending malicious messages). The *honest-but-curious* trust model used in some papers [5], [31] falls into this category. Note that we do not consider data storage those nodes that are transparent from the point of view of data connections, for example, network gateways or Internet routers. Of course, these devices are untrusted, but they can be prevented from accessing data by simply establishing end-to-end secure channels, for example, with the DTLS protocol [32].

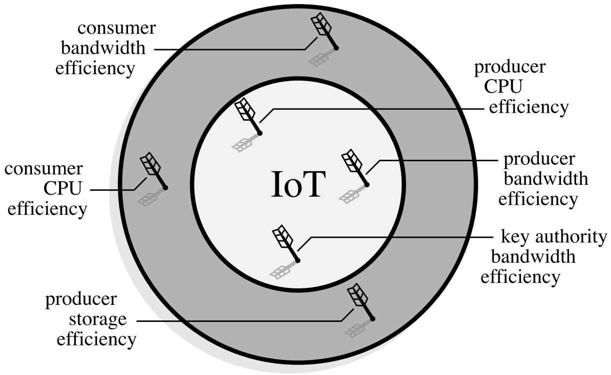

consumer bandwidth efficiency
producer CPU efficiency
consumer CPU efficiency
IoT
producer bandwidth efficiency
key authority bandwidth efficiency
producer storage efficiency

Fig. 2. Performance indicators.

If ABE is employed, a fourth entity is necessary in the architecture: the *key authority*, which is trusted by all the other entities. The key authority has the responsibility of creating, distributing, updating, and revoking ABE keys. We refer to all these activities with the general term *key management*. The key authority is typically a PC-class device that is not constantly connected to the Internet, but rather it goes online only when key management procedures must be fulfilled.

A. Performance Indicators

We identify three KPIs that an ABE scheme must offer to be best suited for IoT applications (see Fig. 2). These three primary properties are: 1) producer CPU efficiency; 2) producer bandwidth efficiency; and 3) key authority bandwidth efficiency. We further identify three APIs, which are desirable only if they do not jeopardize one or more KPIs. APIs are: 1) producer storage efficiency; 2) consumer CPU efficiency; and 3) consumer bandwidth efficiency. Of course, every choice about a set of key and APIs is debatable, and one should analyze the specific IoT application to identify key indicators case by case. Nevertheless, we feel that our choices fit the *typical*

IoT application, and they are also reasonable for many specific, less typical ones. In the following, we will explain our rationale behind such a claim.

The producer CPU efficiency is a key indicator for the typical IoT application because data are usually produced by constrained devices, which have limited processing capabilities. Most hardware platforms used in IoT applications are indeed CPU constrained. For example, the Espressif ESP32 platform [33] uses an Extensa dual-core LX6 microprocessor, operating at 240 MHz. The Zolertia RE-Mote platform [34], which we use in our experiments, is even more constrained as it mounts an ARM Cortex M3 microcontroller, operating at 32 MHz. Moreover, such devices are often battery powered, so that they must employ low-power communication protocols, which have low bit rates. For example, NB-IoT supports 250 kb/s (uplink) [35], while Bluetooth Low Energy 230 kb/s [36], IEEE 802.15.4 163 kb/s [37], and LoRa only 50 kb/s [38]. This makes the producer bandwidth efficiency a key indicator.

On the other hand, the CPU and bandwidth efficiency of the consumers are not as important as those of the producers because data are usually consumed by users equipped with resourceful devices. Of course, in those specific applications in which consumers are small battery-powered actuators, their efficiency may rise in importance. However, considering the typical case, we decided to categorize the consumer CPU and bandwidth efficiency as APIs. Notably, we consider the producer storage efficiency only as an accessory indicator because many ABE schemes take up very little storage on the producer, i.e., few hundreds of bytes. For those schemes in which the producer storage load is more pronounced, there are techniques to sensibly alleviate it (see Section IX-A). Note that the producer storage capacity may become the real bottleneck of the overall system efficiency if onboard storage is the only storage media used since possibly a large amount of encrypted data are stored locally. In this specific case, the producer storage efficiency may become a KPI.

Regarding the key authority, we believe that its CPU efficiency is not a key indicator since the key authority is typically a PC-class device with good computational capabilities. The only scenario in which the key authority CPU load might be relevant is that of a scheme using the naive revocation technique, described in Section IV-F. Rather, we consider the key authority bandwidth efficiency a key indicator mainly for scalability issues. Indeed, in many ABE schemes the key authority traffic grows with the number of consumers in the system, which tends to be quite large in IoT applications, i.e., hundreds to thousands of devices. Recent estimations by IoT Analytics Research [39] indicate about 12.3 billion of IoT connections worldwide in 2021, which are expected to grow to 27.1 billion in 2025.

Finally, in the specific case of blockchain data storage, it is worth noting that the storage efficiency may rise in importance as a performance indicator. Indeed, the typical blockchain, e.g., Ethereum, makes users pay each kilobyte of storage space in cryptocurrency. However, the blockchain storage efficiency can be enhanced by improving both the key authority bandwidth (i.e., smaller key update material) and the producer bandwidth (i.e., smaller ciphertexts). Therefore, we decided to neglect such a performance indicator because the strategies to improve it coincide with the strategies adopted to improve two other key indicators, explained, respectively, in Sections VI and VII.

Broadly speaking, given a specific IoT application, practitioners should first identify the most critical performance indicator for that application. Then, they should select a pool of ABE schemes that perform well on that indicator. After that, they should identify a second-most critical indicator and repeat the same process to restrict the scheme selection, and so on.

IV. Background

A. Pairing-Based Cryptography Basics

Pairing-based cryptography, inaugurated by Boneh and Franklin [40], refers to the usage of bilinear maps (also called *pairings*) to construct cryptographic schemes. Most ABE schemes currently in the literature use pairing-based cryptography. The following definitions are commonly used in pairing-based cryptography. Let $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ be three multiplicative cyclic groups of equal order whose group operations are efficiently computable. Let p be their prime order, g_1 be a generator of $\mathbb{G}_1$, and g_2 be a generator of $\mathbb{G}_2$. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a bilinear map that has the following properties.

1) *Bilinearity:* For all $a, b \in \mathbb{Z}_p$, $u \in \mathbb{G}_1$, and $v \in \mathbb{G}_2$, we have $e(u^a, v^b) = e(u, v)^{ab}$.
2) *Nondegeneracy:* $e(g_1, g_2) \neq 1$.
3) *Computability:* There exists an efficient algorithm to compute e.

In practical realizations of pairings, $\mathbb{G}_1$ and $\mathbb{G}_2$ are sets of points on an elliptic curve, while $\mathbb{G}_T$ is a finite field. The group operation of $\mathbb{G}_1$ and $\mathbb{G}_2$ is thus the point-scalar multiplication, while that of $\mathbb{G}_T$ is the modular exponentiation.

Literature on pairing-based cryptography traditionally categorizes pairings into three types [41].

1) *Type I:* These pairings have $\mathbb{G}_1 = \mathbb{G}_2$.
2) *Type II:* These pairings have $\mathbb{G}_1 \neq \mathbb{G}_2$, but there is an efficient homomorphism to map an element in $\mathbb{G}_2$ to an element in $\mathbb{G}_1$.
3) *Type III:* These pairings have $\mathbb{G}_1 \neq \mathbb{G}_2$, and there is no efficient homomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$.

Type I pairings are also called *symmetric pairings*. To describe schemes that use symmetric pairings, we will use a unique symbol $\mathbb{G}$ to represent both $\mathbb{G}_1$ and $\mathbb{G}_2$. Types II and III are also called *asymmetric pairings*. It is worth noting, however, that Type II pairings are rarely used to construct cryptographic schemes because they are inefficient compared to Type III ones, and they also miss some features, e.g., no efficient method to hash onto $\mathbb{G}_2$ [42].

B. Basic ABE Algorithms

The basic building block of ABE is the concept of *attribute*, which is a property associated with a piece of data or with a data consumer. An *access policy* describes the access authorization associated either with a piece of data or with a data consumer. It is typically represented through a Boolean formula that has attributes as arguments. An access policy is typically visualized as a tree (*policy tree*) in which leaf nodes are attributes, and intermediate nodes are Boolean operators. Different schemes allow for different degrees of *expressiveness* in the access policies, meaning that they impose constraints on the shape of the policy tree. For example, some ABE schemes allow only for *k*-of-*n* operators (*threshold gates*) or only for AND operators, or they limit the height of the policy tree. Some schemes use an linear secret sharing scheme (LSSS) representation, so that an access policy is expressed with a matrix–vector pair in place of a Boolean formula. However, there are algorithms in [43] and [44] to transform an LSSS representation into a policy tree; hence, in the following, we will always represent policies by means of policy trees without loss of generality.

ABE comes in two paradigms: 1) key-policy ABE (KP-ABE) and 2) ciphertext-policy ABE (CP-ABE). In both paradigms, to encrypt data, it is necessary to own a copy of the *public parameters*, which are public and unique for all encrypting parties. Moreover, to decrypt data it is necessary to own a *decryption key*, which is private and specific for each decrypting party. In KP-ABE, ciphertexts are associated with a set of attributes that describe them, and decryption keys are associated with an access policy. Access policies describe the “capability to access what,” referred to the owner of the decryption key. KP-ABE schemes empower the key authority since it decides the access authorizations when creating decryption keys. Conversely, in CP-ABE, ciphertexts are associated with an access policy, and decryption keys are associated with a set of attributes. Access policies describe the “capability to be accessed by whom,” referred to the encrypted data. CP-ABE schemes empower the data producers since they decide the access authorizations at the moment of encrypting data.

Any KP-ABE scheme implements at least the following four algorithms.

1) *(MK, EK) =* **Setup**(κ)*:* This algorithm initializes the scheme with a strength given by the security parameter κ and randomly creates and returns a *master key MK* and the public parameters *EK*. The master key is kept secret by the key authority, whereas the public parameters are made public and used to encrypt data.
2) *(C) =* **Encrypt**(M, γ, EK)*:* This algorithm encrypts a message M (*plaintext*) described by the attribute set γ, by means of the public parameters *EK*. It returns the encrypted message C (*ciphertext*), which embeds the given attribute set.
3) *(DK) =* **KeyGen**$(\mathcal{T}, MK)$*:* This algorithm creates a new decryption key associated with the access policy $\mathcal{T}$, by means of the master key *MK*. It returns the decryption key *DK*, which embeds the given access policy.
4) *$(M$ or $\perp)$ =* **Decrypt**(C, DK)*:* This algorithm decrypts a ciphertext C with the decryption key *DK*. It returns the original message M if and only if the access policy $\mathcal{T}$ evaluates to true on the attribute set γ embedded in the ciphertext C; otherwise, it returns the null value $\perp$.

Any CP-ABE scheme implements at least the following four algorithms.

1) *(MK, EK) =* **Setup**(κ)*:* This algorithm acts similarly to the one in the KP-ABE schemes.
2) *(C) =* **Encrypt**$(M, \mathcal{T}, EK)$*:* This algorithm encrypts a message M associated with the access policy $\mathcal{T}$, by means of the public parameters *EK*. It returns the encrypted message C, which embeds the given access policy.
3) *(DK) =* **KeyGen**(γ, 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.
4) *$(M$ or $\perp)$ =* **Decrypt**(C, DK)*:* This algorithm decrypts a ciphertext C with the decryption key *DK*. It returns the original message M if and only if the access policy $\mathcal{T}$ evaluates to true on the attribute set γ embedded in the decryption key *DK*; otherwise, it returns the null value $\perp$.

C. Security Guarantees

In this survey, we consider only ABE schemes for which formal security proof has been provided. Typically, ABE schemes are proved to be IND-CPA-secure under the standard model or the random oracle model, with the assumption of hardness of the Bilinear Diffie–Hellman (BDH) problem or some other related problem. An exception is the classic CP-ABE scheme by Bethencourt *et al.* [45], which provides a weaker security guarantee under the generic bilinear group model. Note that the IND-CPA security guarantee defends only against passive adversaries. However, cheap transformations (e.g., [46]) are usually applicable to an IND-CPA scheme to obtain an IND-CCA one, which also defends against active adversaries.

D. Universe Type

The *attribute universe* is the ensemble of all the attributes that can appear in access policies or attribute sets in a particular application. In literature, ABE schemes are traditionally divided into *small-universe schemes* and *large-universe schemes*. A scheme is small universe if the key authority must fix a finite attribute universe at setup time, i.e., when it executes the **Setup** algorithm. Small-universe schemes have public parameters that grow linearly with the size of the attribute universe. Typically, the key authority can create new attributes after the setup time, but then it must update the public parameters and deliver them to all the data producers. On the other hand, a scheme is large universe if the key authority does not need to fix a finite attribute universe at setup time. Large-universe schemes have public parameters whose size does not depend on the attribute universe. The attribute universe itself is virtually unlimited, in the sense that data producers can create new attributes at any time without communicating with the key authority.

E. Access Structure Language Expressiveness

Access structure languages define how the attributes in the universe can be combined to express access policies. The expressiveness of an access structure language is a qualitative measure we give to evaluate its capability to express

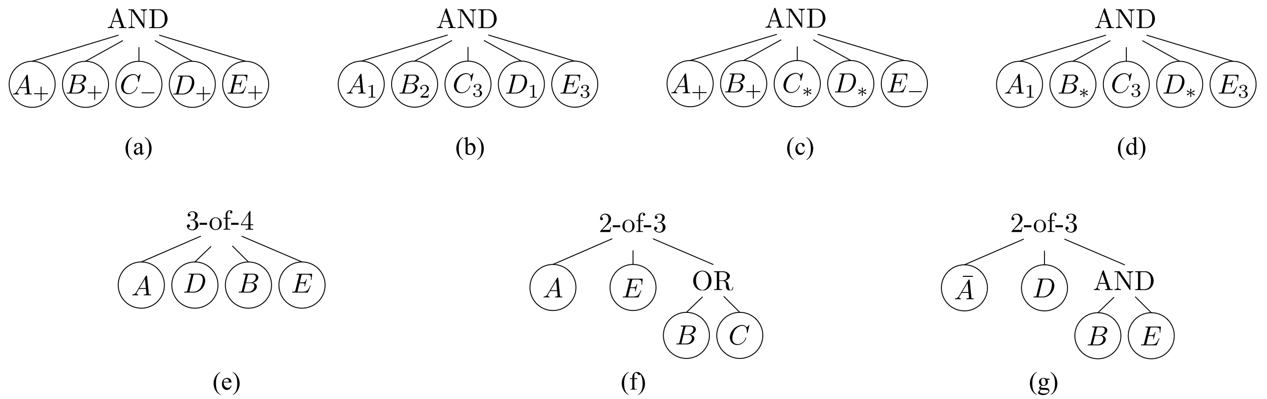

AND
A_+ B_+ C_- D_+ E_+
(a)
AND
A_1 B_2 C_3 D_1 E_3
(b)
AND
A_+ B_+ C_* D_* E_-
(c)
AND
A_1 B_* C_3 D_* E_3
(d)
3-of-4
A D B E
(e)
2-of-3
A E OR B C
(f)
2-of-3
$\bar{A}$ D AND B E
(g)

Fig. 3. Examples of access policies created with different access structure languages. The universe of attributes is $U = \{A, B, C, D, E\}$. In the multivalued access structure languages [Fig. 3(b) and (d)], each attribute can assume one among three distinct values, e.g., the attribute A can assume either the value A_1, A_2, or A_3. (a) AND$_\pm$. (b) AND$_m$. (c) AND$^*_\pm$. (d) AND$^*_m$. (e) k-of-n. (f) Full monotonic. (g) Full nonmonotonic.

different kinds of access policies. We briefly introduce few access structure languages sorted by increasing order of expressiveness.

1) AND gate on Boolean attributes, which can assume either a positive (e.g., A_+) or a negative (e.g., A_-) value, denoted by the symbol AND$_\pm$ [Fig. 3(a)]. All the attributes inside the universe *must* always be included in an access policy. Therefore, in each policy, the value of any attribute must always be specified (either positive or negative). This language is used and explained, for example, in [24].
2) AND gate with multivalued attributes, denoted by the symbol AND$_m$ [Fig. 3(b)]. It is as expressive as the AND$_\pm$, but it provides a better way to manage attributes. Each attribute in the universe is a group of mutually exclusive values (e.g., A_1, A_2, A_3,...): one and only one of these values must be in any given policy. This language is used and explained, for example, in [27].
3) AND gate on Boolean attributes with *wildcards* (e.g., A_*), and AND gate with multivalued attributes with wildcards, denoted by the symbols AND$^*_\pm$ and AND$^*_m$ [Figs. 3(c) and (d)], respectively. The basic idea of a wildcard is a "do not care," meaning that the value of the specific attribute is not relevant to satisfy the access policy. This leads to more effective policies. The AND$^*_\pm$ and the AND$^*_m$ languages are used and explained, for example, in [47] and [48], respectively.
4) Threshold monotonic language on the presence of attributes in the attribute set, denoted by the symbol k-of-n [Fig. 3(e)]. A policy is composed of n attributes, where n can be any value from 1 to the total number of attributes inside the universe. For the policy to be satisfied, at least a threshold of k attributes out of n (with $1 \leq k \leq n$) must be present in the attribute set. Note that, from this language on, attributes are not considered Boolean or multivalued variables but rather simple "tags" that can be either present within or absent from the attribute set. This language allows us to implement also AND gates (when $k = n$) and OR gates (when $k = 1$). Intuitively, this language is more expressive than the previous ones since more attribute sets can satisfy a single access policy. This language is used and explained, for example, in [14].
5) Full monotonic language on the presence of attributes in the attribute set, denoted by the symbol "full monotonic" [Fig. 3(f)]. This language considers the access structure as a tree, in which the internal nodes are k-of-n gates, and leaf nodes are attributes. Not every attribute inside the universe must appear inside the access policy. This language is used and explained, for example, in [45] and [49].
6) Full nonmonotonic language on the presence of attributes in the attribute set, denoted by the symbol "full nonmonotonic" [Fig. 3(g)]. This language includes all the benefits provided by the full monotonic language, plus the ability to express, inside the access structure, the required absence of an attribute from the attribute set (e.g., $\bar{A}$ is the absence of the attribute A). This language is used and explained, for example, in [50].

Note that the policies of the AND-based languages (AND$_\pm$, AND$_m$, AND$^*_\pm$, and AND$^*_m$) include all the attributes of the universe. With languages with higher expressiveness, one can create access structures that cannot be created using languages with lower expressiveness.

F. Key Management

To be used in practice, ABE schemes must provide for some additional functionalities of *key management*. In particular, they have to provide mechanisms to distribute decryption keys and to revoke them. While distributing a key to a joining consumer is usually an easy task, key revocation is more challenging, as shown by the rich literature dedicated to the problem [16], [19], [31], [51]–[54]. An ABE scheme providing for a native key revocation mechanism is called a *revocable scheme*.

It must be noted that any nonrevocable scheme can be extended to a revocable one as follows. When a key must be revoked, the key authority runs the **Setup** algorithm, thus creating a new master key and public parameters. Next, for each nonrevoked consumer, the key authority generates a new decryption key with the old access privileges. Then, the key authority has two options to confidentially distribute the new decryption keys. It can establish a secure channel with each consumer, e.g., with DTLS. Otherwise, it can encrypt the new decryption keys with the consumers' public keys, e.g., RSA keys, and store the resulting ciphertexts on the data storage. The latter choice is usually preferable for the key authority since it can rapidly conclude all its revocation operations and go back offline. Therefore, the key distribution task is delegated to the data storage and performed lazily upon data requests by consumers. Note that a consumer will be given more than one decryption key if more than one key revocation happened since its last data request. The key authority also stores the new public parameters on the data storage. Before encrypting a new piece of data, producers must download the latest version of public parameters from the data storage. We call this revocation mechanism, viable for all nonrevocable schemes, the *naive revocation mechanism*.

G. Cited Schemes and Naming Convention

From now on, we will refer to the concrete ABE schemes proposed by the literature with the initials of the authors' surnames, the publication year, and an optional final number discriminating different schemes within the same paper. For example, "SW05-2" refers to the second scheme presented by Sahai and Waters [14], "BSW07" refers to the (unique) scheme presented by Bethencourt, Sahai, and Waters [45], and so on. Table I contains the names of every ABE scheme cited in the present paper, with its precise reference and its basic characteristics.

V. Producer CPU Efficiency

In this section, we focus on the CPU efficiency of ABE schemes from the point of view of the data producers. It is worth noting that the encryption operations dominate the CPU load required by a scheme on the producers. Indeed, the producers are also involved in key management processes, but these typically do not require producers to perform any computation but rather to download new public parameters or lists of revoked consumers. In the typical ABE scheme, encryption requires point-scalar multiplications and, for large-universe schemes, hash functions whose output is in an elliptic-curve group. Hence, a preliminary observation regarding encryption efficiency is that large-universe schemes are typically less performant than small-universe ones. This is because hash functions over elliptic-curve groups are quite burdensome for resource-constrained devices (see Table III of Section VIII). A first straightforward method to save producer CPU is thus to use small-universe schemes. Of course, the additional cost of large-universe schemes can be alleviated by precomputing the hashes of the attributes used in encryption and avoiding encrypting with attributes other than the precomputed ones. However, this partially nullifies the advantage of being large universe since the producer can encrypt only with a predefined set of attributes, just like it happens for small-universe schemes. In many IoT applications, it is not possible to fix a static set of attributes for each producer to encrypt with, so other more flexible strategies to improve CPU efficiency are needed.

Another simple technique to lower the producer CPU load, featured in LPD21 is to use the *digital envelope*, that is, to let producers encrypt a symmetric key with ABE with a specific attribute set (KP-ABE) or a specific policy (CP-ABE). Then, all the plaintexts that must be protected by such an attribute set/policy are efficiently encrypted with such a symmetric key. This technique can be applied to every ABE scheme, and it is particularly advantageous in case producers must encrypt many times using the same attribute set/policy. Nonetheless, it makes the key management more complex because symmetric keys must be stored by producers and consumers, and possibly revoked by the key authority.

Besides these basic techniques, we identified three main strategies to improve the producer CPU efficiency: 1) encryption outsourcing [66], [67], [74]; 2) adopting alternative mathematics [71], [73], [77]; and 3) adopting Type III pairings [69], [75].

A. Encryption Outsourcing

The schemes adopting this strategy reduce the producer CPU load sensibly, but they need specific architectural or usage features, for example, the presence of full-resource neighbors or the presence of users that periodically load producers with precomputed quantities. Touati *et al.* [67] proposed a CP-ABE scheme[1] (TCB14) that collaboratively accomplishes the encryption of a message. In order to do that, the data producer needs to establish secure channels with at least two trusted full-resource neighbors to which delegate burdensome operations. The neighbors compute partial results and send them to the producer, which combines them, creating the final ciphertext. Touati and Challal [74] proposed a similar KP-ABE scheme (TC16). The offloading technique of TCB14 and TC16 greatly unburdens producers, but it needs multiple resourceful devices in the neighborhood, which could be missing. Second, the outsourcing system heavily impacts bandwidth so that producers could spend more time and energy in communicating than what they save in processing.

Another efficient solution is proposed by Hohenberger and Waters in [66]. The authors propose a KP-ABE scheme (HW14-1) and a CP-ABE one (HW14-2), based on the RW13-2 and RW13-1 schemes, respectively. Both the schemes split the encryption algorithm into two phases. In the first phase (offline phase), all the burdensome operations are preprocessed whereas, in the second phase (online phase), light operations are performed to generate the actual ciphertext. This solution is useful if data producers are mobile devices that experience battery charging cycles, e.g., smartphones. The offline phase is executed while the device is charging, and when the data are ready—and the device is possibly not charging—the online phase is executed. Note that HW14-1

[1] Despite [67] and [74] do not provide formal security proofs, we include them anyway in the present survey because similar security proofs of the base schemes BSW07 [45] and GPSW06-1 [49] should apply under the assumption of secure channels between the producer and the full-resource neighbors.

TABLE I
CITED ABE SCHEMES

Scheme Name	Reference	Paradigm	Universe	Access Structure Expressiveness
SW05-1	[14], Sec. 4.1	KP-ABE	small	k-of-n
SW05-2	[14], Sec. 6.1	KP-ABE	large	k-of-n
GPSW06-1	[49], Sec. 4.2	KP-ABE	small	full monotonic
GPSW06-2	[49], Sec. 5.1	KP-ABE	large	full monotonic
GPSW06-3	[49], App. A.1	KP-ABE	small	full monotonic
BSW07	[45], Sec. 4.2	CP-ABE	large	full monotonic
OSW07	[50], Sec. 3.1	KP-ABE	large	full nonmonotonic
BGK08-4	[51], Sec. 6	KP-ABE	small	full monotonic
AI09	[52], Sec. 4	KP-ABE	large	full monotonic
EMNOS09	[55], Sec. 4	CP-ABE	small	AND_m
HLR10	[56], Sec. 3	CP-ABE	small	k-of-n
LOSTW10-1	[57], Sec. 2.3	CP-ABE	small	full monotonic
YWRL10	[31], Sec. IV	KP-ABE	small	full monotonic
ZH10	[58], Sec. 3	CP-ABE	small	$\mathrm{AND}^*_\pm$
ALP11	[59], Sec. 5	KP-ABE	large	$\mathrm{AND}_\pm$
HN11	[60], Sec. 4	CP-ABE	large	full monotonic
W11-1	[61], Sec. 3	CP-ABE	small	full monotonic
W11-2	[61], Sec. 5	CP-ABE	small	full monotonic
W11-3	[61], Sec. 6	CP-ABE	small	full monotonic
W11-4	[61], App. A	CP-ABE	large	full monotonic
W11-5	[61], App. B	CP-ABE	large	full monotonic
GZCMZ12-1	[62], Sec. 3.1	CP-ABE	small	k-of-n
GZCMZ12-2	[62], Sec. 3.2	CP-ABE	small	k-of-n
LGRDY12	[27], Sec 3.2	CP-ABE	small	AND_m
SSW12-1	[63], Sec. 8	KP-ABE	small	full monotonic
SSW12-2	[63], Sec. 10	CP-ABE	small	full monotonic
RW13-1	[64], Sec. 4	CP-ABE	large	full monotonic
RW13-2	[64], App. C	KP-ABE	large	full monotonic
ZHW13	[48], Sec. 4	CP-ABE	small	$\mathrm{AND}^*_\mathrm{m}$
DJ14	[65], Sec. 4	CP-ABE	small	AND_m
HW14-1	[66], Sec. 3	KP-ABE	large	full monotonic
HW14-2	[66], Sec. 4	CP-ABE	large	full monotonic
PYS14	[47], Sec. 3	CP-ABE	small	$\mathrm{AND}^*_\pm$
TCB14	[67], Sec. IV	CP-ABE	large	full monotonic
ZCLLL14	[24], Sec. 5.3	CP-ABE	small	$\mathrm{AND}_\pm$
ZZCLL14	[68], Sec. 4	CP-ABE	small	$\mathrm{AND}^*_\mathrm{m}$
CGW15-1	[69], App. B.1	KP-ABE	small	full monotonic
CGW15-2	[69], App. B.2	CP-ABE	small	full monotonic
PYSC15-1	[70], Sec. 3	KP-ABE	small	$\mathrm{AND}^*_\pm$
PYSC15-2	[70], Sec. 4	CP-ABE	small	$\mathrm{AND}^*_\pm$
YCT15	[71], Sec. 4.2	KP-ABE	small	full monotonic
CDLQ16	[72], Sec. 4	CP-ABE	large	full monotonic
OD16	[73], Sec 3	CP-ABE	small	$\mathrm{AND}_\pm$
TC16	[74], Sec. IV	KP-ABE	small	full monotonic
AC17-1	[75], Sec. 3	CP-ABE	large	full monotonic
AC17-2	[75], App. B	KP-ABE	large	full monotonic
AC17-3	[75], App. D	CP-ABE	large	full monotonic
AC17-4	[75], App. E	CP-ABE	small	full monotonic
AC17-5	[75], App. F	KP-ABE	small	full monotonic
LYHZS17	[76], Sec. 7	CP-ABE	large	full monotonic
ODKCJ17	[77], Sec. IV	CP-ABE	small	$\mathrm{AND}_\pm$
QZZC17	[78], Sec. 3	CP-ABE	large	full monotonic
HWY18	[79], Sec. 5	CP-ABE	large	full monotonic
JSMG18-1	[80], Sec. 4	CP-ABE	small	$\mathrm{AND}_\pm$
JSMG18-2	[80], Sec. 5	CP-ABE	small	$\mathrm{AND}_\pm$
LYZL18	[81], Sec. 4	CP-ABE	small	full monotonic
XYM19	[82], Sec. 5	CP-ABE	large	full monotonic
CM21	[83], Sec. 4	CP-ABE	large	full monotonic
LPD21	[53], Sec. 5	CP-ABE	large	full monotonic
RPDY21	[54], Sec. 4	KP-ABE	small	full monotonic

and HW14-2 do not improve CPU efficiency strictly speaking, because they do not outsource encryption, but rather rationalize the CPU usage cycles. However, it is worth noting that such schemes can be seamlessly adapted to outsource encryption. Indeed, the offline phase can be outsourced to some trusted resourceful device, and the resulting preprocessed quantities can be loaded on the producers through some secure channel. We will explore this possibility in the experimental section (Section VIII). Note that HW14-1 and HW14-2 allow the producer to decide the message and the attributes used in encryption in the online phase when the full-resource device is offline. TCB14 and TC16 do not provide this feature.

To sum up, encryption outsourcing does not have general applicability in IoT. Outsourcing is possible only if there are trusted full-resource devices close to the producer. This happens, for example, in the case of a network of producers administered by a unique entity, in which one or more full-resource gateways are present. However, these full-resource nodes would become a single point of trust of the whole system, and their compromise might have a devastating effect on data confidentiality.

B. Alternative Mathematics

Some studies propose ABE schemes that do not employ pairing operations (*pairing-free schemes*). These schemes employ different cryptographic mathematics, usually ECC [71], [73] or RSA [77]. Pairing-free schemes allow for the fastest encryption in the literature. Indeed, RSA-based schemes employ extremely simple modular mathematics, which can also be hardware accelerated in modern IoT devices [34]. ECC-based schemes can employ small and standard elliptic curves, for example, the P-192 one for 96-bit security or the P-256 one for 128-bit security [84]. These curves do not support an efficiently computable pairing operation, but they are generally more efficient than pairing-friendly curves with the same level of security. This is because they can be represented on the shortest possible number of bits, for example, 160 bits for obtaining 80-bit security. Also, operations on the standard curves are hardware-accelerated in modern IoT devices [34]. Prominent pairing-free ABE schemes in the literature are YCT15 and OD16, which employ ECC mathematics, and ODKCJ17, which employs RSA mathematics.

Unfortunately, the security of such ABE schemes is debated in the cryptography community. Some of them, i.e., YCT15, OD16, and ODKCJ17, have been successfully cryptanalized by successive papers: respectively [85] and [86] for YCT15, and [87] for OD16 and ODKCJ17. A recent paper [86] cryptanalized several other pairing-free ABE schemes. Herranz [87] provided a simple argument motivating the reason why a secure RSA/ECC ABE scheme should not exist. Indeed, since ABE is a generalization of identity-based encryption, if one could design a secure RSA/ECC ABE scheme, this could be easily converted into a secure RSA/ECC IBE scheme. However, designing such an IBE scheme has shown to be an extremely hard problem. In practice, such a problem is unsolved since 1984, when the IBE problem was first stated by Shamir [88]. This argument raises doubts about the security of all the RSA/ECC ABE schemes published until now, despite they are usually accompanied by formal security proofs. We chose not to neglect RSA/ECC schemes in this survey, waiting for further research to clarify whether secure RSA/ECC ABE schemes are possible or not.

C. Type III Pairings

The majority of ABE schemes have been designed, proved for security, and benchmarked with Type I pairings. This is probably for historical reasons, since the first pairing-based cryptographic schemes were designed with this type of pairings [40], [89]. However, using Type III pairings allows us to speed up some cryptographic operations. This is because Type III pairings permit smaller representations of $\mathbb{G}_1$ elements with the same security level [75], thus leading to faster operations on them. Fortunately, many existing ABE schemes, including the classic SW05-1, GPSW06-1, and BSW07, are easily "portable" to Type III pairings. By doing so, their security proofs are invalidated, but there are formal methods to convert a security proof with Type I pairings to an equivalent one with Type III pairings [90]. Notably, there is no unique way to convert a scheme from Type I to Type III pairings. Broadly speaking, this is because each Type I pairing $e(A, B)$ (with $A, B \in \mathbb{G}$) employed in the scheme can be converted in two different ways: 1) assuming $A \in \mathbb{G}_1$ and $B \in \mathbb{G}_2$ and, thus, leaving the pairing as is, or *vice versa* 2) assuming $A \in \mathbb{G}_2$ and $B \in \mathbb{G}_1$ and, thus, inverting the pairing to be $e(B, A)$. These choices lead to different performance in different operations. Typically, the most convenient choice for the producer efficiency is the one that converts the highest number of $\mathbb{G}$ elements to $\mathbb{G}_1$ elements in the ciphertext. In this way, the encryption performs point-scalar multiplications in $\mathbb{G}_1$, which are the efficient ones. Type III pairings also enjoy much more efficient $\mathbb{G}_1$ hash operations than Type I ones; thus, they are convenient also to reduce the cost of large-universe schemes. On the negative side, using Type III pairings decreases the efficiency of pairing operations and point-scalar multiplications in $\mathbb{G}_2$, which are typically used in decryption and key generation, respectively. Thus, in those IoT applications in which the consumer CPU efficiency and/or the key authority CPU efficiency is more important than the producer's one, adopting Type III pairings could not be a convenient solution.

Some recent studies [69], [75] propose ABE schemes that have been designed explicitly for Type III pairings, to improve encryption efficiency. Chen *et al.* [69] proposed a framework for building ABE schemes, and they applied such a framework to propose two concrete schemes: 1) CGW15-1 (KP-ABE) and 2) CGW15-2 (CP-ABE). The authors used Type III pairings for their schemes in order to improve the encryption performance. Agrawal and Chase [75] proposed AC17-1 (CP-ABE, named "FAME" by the authors) and AC17-2 (KP-ABE), both employing Type III pairings. Such schemes are inspired by Chen *et al.*'s ones, but they provide for large universes. In the same paper, the authors also provide other schemes, i.e., AC17-3, AC17-4, and AC17-5, which are Type III conversions of BSW07, W11-1, and GPSW06-3, respectively. Not all the schemes presented by Agrawal and Chase are optimized for encryption. Among these schemes, the fastest ones in encryption are AC17-5 (for the KP-ABE paradigm) and AC17-1 (for the CP-ABE one).

VI. Key Authority Bandwidth Efficiency

Key authority bandwidth efficiency depends entirely on key management operations. A system deployed to run over the long haul must foresee that consumers' roles and privileges can change over time, consumers can join or leave the system, and consumers' keys can get compromised, either because stolen by an attacker or lost. In response to these events, the key authority should distribute new keys or revoke old ones. While key distribution for joining consumers is typically a trivial task, key revocation is more complex. In the literature, key revocation is classified into three categories: 1) *direct*; 2) *indirect*; and 3) *attributewise*. In direct revocation, consumers' decryption keys are associated with *identifiers*, and revoking a key means disabling the consumer identifier from decryption of new ciphertexts. The list of revoked identifiers, usually referred to as *revocation list*, must be available to all the producers, which use it during encryption in such a way to exclude revoked keys from being capable of decrypting the new ciphertexts. Differently, in indirect revocation, the revocation process involves the nonrevoked consumers. In particular, their decryption keys are updated in order to decrypt new ciphertexts, while revoked ones are not. Usually, producers do not take part in the revocation process, but some schemes, e.g., LPD21, require producers to update a small part of the public parameters and use them for new encryptions. The naive revocation technique presented in Section IV-F falls in this category, but its performances are poor since the key authority generates and distributes new decryption keys to all the nonrevoked consumers. Also, all the producers must obtain the new public parameters. Attributewise revocation can be seen as a type of indirect revocation at attribute level. To revoke a compromised key, the key authority issues a new version of the attributes present in that key. Every decryption key, except for the revoked one, is updated to the new version. Producers need to obtain the new version of the public parameters to generate new ciphertexts that the revoked key cannot decrypt. In the following, we describe existing revocation strategies and analyze some approaches proposed in the literature that attempt to limit the key authority effort to handle key revocations.

We identified three main strategies to reduce the key authority traffic: 1) adopting direct revocation [70], [81], which completely unburdens the key authority of the key revocation tasks; 2) adopting binary tree structures within indirect revocation [51], [52], [63], [72], [78], [82], which reduce the key authority traffic from linear to logarithmic in the number of consumers; and 3) adopting attributewise revocation [31], [60], [76], which makes the traffic generated by revocation tasks dependent on the number of revoked attributes instead of the number of consumers.

A. Direct Revocation

Direct revocation is the most effective strategy to reduce key authority traffic. When a decryption key is compromised, the key authority simply adds the identifier associated with that key to a revocation list. Producers use the revocation list as additional input during encryption to generate new ciphertexts that decryption keys associated with identifiers in the revocation list cannot decrypt. However, as already highlighted in Section III, an IoT ABE scheme should weigh as few as possible on the resource-constrained producers. Direct revocation often fails in this because producers' bandwidth and encryption efficiency are inevitably reduced. Indeed, producers need to obtain an updated copy of the revocation list prior to encryption (bandwidth overhead), use the revocation list during encryption (encryption overhead), and then upload a larger ciphertext on the data storage (bandwidth overhead). As a consequence of these clear disadvantages, some studies proposed ABE schemes with direct revocation that enlighten the burden on the producers. For example, Liu *et al.* [81] proposed a direct revocable CP-ABE scheme with a short revocation list (LYZL18). To achieve this, the revocation list is condensed into a single $\mathbb{G}$ element in the ciphertext. In this scheme, decryption keys have a planned expiration date, and revoked keys—whose expiration date is over—are excluded from the revocation list to relieve encryption efficiency.

Phuong *et al.* [70] proposed a direct revocable ABE scheme for both KP-ABE (PYSC15-1) and CP-ABE (PYSC15-2) paradigms. They combine ABE with broadcast encryption to impede decryption to revoked identifiers. The encryption algorithm takes as input the list of nonrevoked identifiers. This is a drawback because producers must update their list when a consumer is revoked and every time a new consumer joins the system. As in the LYZL18 scheme, the list used during encryption is condensed into a single $\mathbb{G}$ element. Overall, the scheme reaches good efficiency in terms of bandwidth and storage because the KP-ABE variant has short ciphertexts and constant-size decryption keys, and the CP-ABE variant has constant-size ciphertexts and short decryption keys.

In short, direct revocation is the strategy that weighs less on the key authority but inevitably hampers the other KPIs as it adds computational and communication overhead on the producers.

B. Binary Trees

Indirect revocation typically leverages an additional input to revoke consumers: time. Indirect revocation schemes are organized in time periods, and at the beginning of a new period, the key authority updates only nonrevoked consumers' decryption keys. In a naive approach [40], the key authority generates a new key for each consumer at each new period and individually sends them to consumers through secure channels. Obviously, this solution does not scale well with regard to the key authority since its computational and communication costs increase linearly with the number of consumers.

Boldyreva *et al.* [51] improved the efficiency of the keys update mechanism. In their scheme (BGK08-4), the costs for the key authority are reduced from linear to logarithmic in the number of consumers. To achieve this performance, they first proposed to use a binary tree for creating key-update material. Notably, this information, which they call *key update*, is not a secret. The key update can be published on the data storage to eliminate the need for interaction between consumers and the key authority. In this scheme, decryption keys are associated with identifiers. A consumer owns a long-term secret key linked to its identifier and a short-term decryption key that is valid for the current time period only.

At each time period, the consumer creates a new short-term decryption key by combining the key update with its long-term secret key. Only nonrevoked consumers are capable of performing this operation. Indeed, the key authority generates key updates that do not allow revoked consumers to create a new valid short-term decryption key. The size of a key update is $O(R\log(N/R))$ elements in $\mathbb{G}$, where N is the total number of consumers, and $R \leq N$ is the number of revoked consumers.

Inspired by Boldyreva *et al.*'s work [51], many indirect revocable ABE schemes using binary tree construction have been proposed [72], [78], [82], [91]. Sahai *et al.* [63] extended the concept of revocation to a broader sense and proposed an indirect revocable ABE scheme for both KP-ABE (SSW12-1) and CP-ABE (SSW12-2) paradigms. They dealt with the problem of revoking access also to previously encrypted data. In their construction, based on LOSTW10-1, the untrusted storage is enabled to re-encrypt old ciphertexts to a more restrictive policy using only public information. More precisely, a ciphertext encrypted at time t is transformed to an independent encryption of the same message under the same attribute set at time $t + 1$. Note that re-encryption is performed without accessing the message. Xu *et al.* [82] proposed a revocable ABE scheme (XYM19) that adds a feature to the one introduced by Sahai *et al.* [91]. In their construction, based on RW13-1, the authors deal with the *decryption key exposure* attack,[2] which was first introduced by Seo and Emura [92]. The authors show that the performance of their scheme is very similar to that of SSW12-1.

Cui *et al.* [72] (CDLQ16), Qin *et al.* [78] (QZZC17), and, recently, Cheng and Meng [83] (CM21) also extended the techniques of Boldyreva *et al.* [51] in the CP-ABE realm. In all these constructions, the untrusted storage (which does not hold any secret information) carries out the majority of decryption and revocation workload. As in [51], at the beginning of a new time slot t, the key authority loads the key update on the data storage. For a consumer with identifier *id*, the data storage computes a *transformation* key for the consumer *id* and the time slot t. On a data request by a consumer, the data storage uses the transformation key to manipulate the requested ciphertext. The consumer finalizes the decryption at a low and constant cost. The difference between CDLQ16 and both QZZC17 and CM21 is that the latter have decryption key exposure resistance, adding a moderate cost for the consumer. While CDLQ16 and QZZC17 prove their security only in the *one-user setting*, CM21 proves its security against an enhanced security model called *multiuser setting*.[3]

In short, the binary tree is an efficient construction to achieve indirect revocation as it limits the key authority bandwidth to be logarithmic in the number of consumers.

1) Hybrid Scheme: Attrapadung and Imai [52] proposed a hybrid revocable KP-ABE scheme (AI09) that allows for both direct and indirect revocation modes. The indirect revocation technique is pretty much the one proposed in Boldyreva *et al.*'s work [51] that we previously described. That is, the key authority publishes a key update of size $O(R\log(N/R))$ at each time period through which nonrevoked consumers can update their keys. In the direct revocation, producers use the elements in the key update as additional input during encryption. If direct revocation is used to create a ciphertext at time slot t, a nonrevoked consumer is not required to update its key for that time period. If indirect revocation is used to create a ciphertext at time slot t, a nonrevoked consumer is required to update its key for that time period. Depending on its resources, a producer can use and switch between the direct and indirect revocation modes. However, if producers mix direct and indirect modes within the same time period, the scheme takes the worst of both worlds because it asks an additional effort to handle revocation for both consumers and producers. Moreover, the key authority bandwidth always results as that of BGK08-4.

C. Attributewise Revocation

Attributewise revocation is more fine-grained than the previous strategies because it can revoke the privileges of a consumer at attribute level. This means that the key authority can invalidate just one attribute in a consumer's decryption key. With this strategy, the key authority bandwidth depends on the number of revoked attributes.

Yu *et al.* [31] proposed an attributewise revocable KP-ABE scheme (YWRL10) based on GPSW06-1. We recall that GPSW06-1 has a small universe, and the public parameters contain one *component* (a $\mathbb{G}$ element) for each attribute in the universe. In the YWRL10 construction, all the attributes of the universe (except for one, called *dummy attribute*) are subject to versioning. When the key authority wants to revoke a set of attributes λ embedded in a decryption key, it generates a new version of the public parameters components for the attributes in λ. Then, it computes a secret quantity, i.e., an element in $\mathbb{Z}_p$ called *re-encryption key*, for each attribute in λ. Finally, it loads the updated public parameters' components and the re-encryption keys on the semitrusted storage, thus transferring $|\lambda|\cdot(|\mathbb{G}|+|\mathbb{Z}_p|)$ bits, where the symbol $|\cdot|$ denotes the size of an element expressed in bits. Note that if the key authority wants to revoke a whole compromised key, it can revoke only the minimal set of attributes without which the embedded access policy can never be satisfied. For example, if the compromised key's access policy is A AND (B OR C), the minimal set is $\{A\}$. Depending on the shape of the access policy, this enhancement can save a lot of bandwidth overhead. The data storage is in charge of updating decryption keys of nonrevoked consumers which shared at least one attribute with the revoked attribute set λ. The YWRL10 assumes the data storage as honest-but-curious. Rasori *et al.* [54] proposed a scheme (RPDY21) based on the YWRL10 one, which is secure even with an untrusted data storage.

Hur and Noh [60] proposed an attributewise revocable CP-ABE scheme (HN11) based on BSW07. As in YWRL10, the key authority can revoke a set of attributes of a decryption key associated with a consumer identifier. Producers are not affected by revocation, and they generate CP-ABE ciphertexts

[2] A scheme has decryption key exposure resistance if the compromise of the short-term decryption key does not imply the compromise of the long-term secret key.

[3] For additional details on this security model, refer to [83].

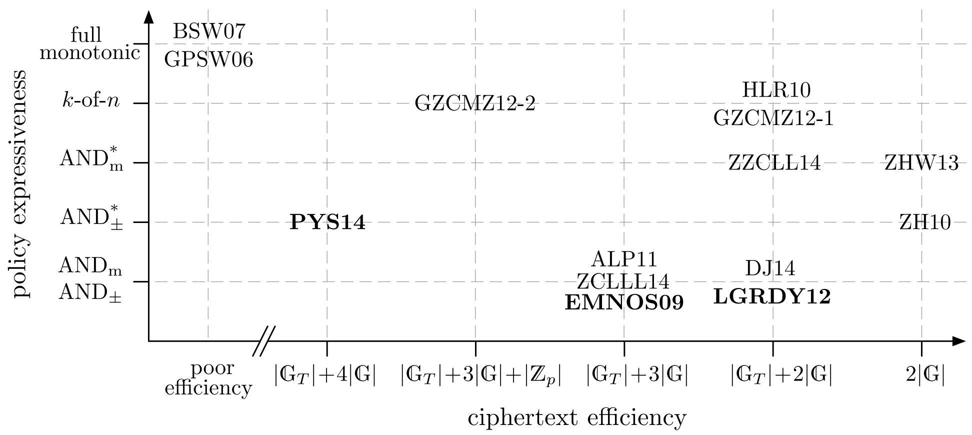


Fig. 4. Analyzed constant-size ciphertext schemes. The classic CP-ABE (BSW07) and KP-ABE (GPSW06-1) schemes are shown as a reference. Schemes in bold have been proved secure under standard assumptions.

by always using the same public parameters. The key authority only communicates to the semitrusted storage the consumer identifier and the change of access privileges it wants to actuate. For example, if the key authority wants to revoke the whole key of consumer *id*, it communicates all the attributes in that key and the associated consumer identifier. The data storage manipulates each ciphertext and re-encrypts it at attribute level so that only nonrevoked consumers (for that attribute) can use that attribute during decryption. On the contrary, if the consumer *id* is revoked for the attribute *i*, it cannot use that attribute anymore for decrypting. This scheme applies two layers of encryption. The first layer is CP-ABE encryption, which is performed by producers, and the second layer is symmetric encryption. The second layer is applied by the data storage and enforces revocation at attribute level. To decrypt a ciphertext, a consumer first proves that it has access privileges for an attribute by performing symmetric decryption. Then, it uses CP-ABE decryption to retrieve the message. This scheme, too, makes use of the binary tree to manage revocation through symmetric cryptography. The HN11 scheme suffers from a vulnerability for which a nonrevoked consumer can collude with a revoked consumer and restore its access privileges. Li *et al.* [76] proposed a scheme (LYHZS17) based on the HN11 one that fixes this vulnerability.

A benefit of attributewise revocation is that, differently from indirect revocation, the key authority can stay offline and perform no task as far as no revocation occurs. Moreover, being attributewise revocation not time-dependent, a key revocation can come into force with immediate effect.

VII. Producer Bandwidth Efficiency

In this section, we focus on the bandwidth efficiency of ABE schemes from the point of view of the data producers. The bandwidth overhead introduced by a scheme on the producers includes the *encryption bandwidth overhead*, i.e., the difference between the ciphertext size and the plaintext size, and the *key management bandwidth overhead*, i.e., the traffic related to key distribution and key revocation mechanisms. Indeed, the producers are also involved in key management processes to download new public parameters or even lists of revoked consumers.

A general and straightforward way to lower the encryption bandwidth overhead is to use the digital envelope technique described in Section V. This is because symmetric-key encryption introduces much less encryption bandwidth overhead compared to ABE. Of course, as already highlighted, it makes the key management overhead more complex because symmetric keys must be stored by producers and consumers, and possibly revoked by the key authority.

Besides this basic technique, we identified three main strategies to lower the bandwidth overhead on the producer: 1) the use of a constant-size ciphertext [24], [27], [47], [48], [55], [56], [58], [59], [62], [65], [68]; 2) the implementation of an efficient key management mechanism [60], [76], [81]; and 3) the use of small group elements for the public parameters and ciphertexts [75].

A. Constant-Size Ciphertext

An effective strategy to reduce the encryption bandwidth overhead is to have ciphertexts of small or constant size. Typically, in many ABE schemes, the ciphertext size depends on the number of attributes either in the access policy (CP-ABE), e.g., BSW07, HW14-2, and LYZL18, or in the attributes set (KP-ABE), e.g., YWRL10, GPSW06-1, and AI09. Clearly, this dependency is detrimental to the producer's traffic. On the other hand, schemes with constant-size ciphertexts reduce the traffic by disposing of such a dependency. However, we notice that a tradeoff emerges between policy expressiveness and encryption bandwidth overhead. Usually, schemes with constant-size ciphertexts use poorly expressive access structures languages, while schemes with nonconstant-size ciphertexts tend to use more expressive access structure languages, allowing the creation of access policies that cannot be built in the constant-size ciphertexts schemes. This is because, in poorly expressive access structures languages (essentially all the AND-based languages), the attributes in the ciphertext are condensed into a single $\mathbb{G}$ element.

Fig. 4 shows some prominent constant-size ciphertext schemes on a Cartesian plane. The *x*-axis denotes efficiency in terms of ciphertext size (the rightmost, the better); the y-axis denotes the expressiveness of the language (the higher, the better); schemes that are proved secure under standard assumptions are shown in bold. Moreover, the classic schemes BSW07 and GPSW06-1 are shown as a reference. We note that the scheme that features the largest ciphertext size is PYS14, which uses the $\text{AND}^*_{\pm}$ language and provides a ciphertext of $|\mathbb{G}_T| + 4|\mathbb{G}|$ bits.

A slightly smaller ciphertext is achieved by the scheme GZCMZ12-2, which features a ciphertext size of $|\mathbb{G}_T|+3|\mathbb{G}|+|\mathbb{Z}_p|$ and uses the threshold monotonic (k-of-n) language. Three schemes, namely, EMNOS09, ALP11, and ZCLLL14, which use AND_m, $\text{AND}_{\pm}$, and $\text{AND}_{\pm}$ languages, respectively, provide a ciphertext of size $|\mathbb{G}_T| + 3|\mathbb{G}|$. Among all the schemes considered for this strategy, ALP11 is the only one that follows the KP-ABE paradigm. It is interesting to point out that ZCLLL14 also provides a very efficient key revocation system concerning the producers. They must download only one $|\mathbb{G}|$ element to update the public parameters after a key revocation, instead of the whole set of public parameters needed in all the other schemes, typically $O(n)$, being n the number of attributes in the universe.

Then, a considerable number of schemes feature a ciphertext size of $|\mathbb{G}_T| + 2|\mathbb{G}|$. Among them, the schemes DJ14 and LGRDY12 provide the worst expressiveness (AND_m), whereas the scheme ZZCLL14 provides slightly better expressiveness, using AND^*_m. However, the schemes with the best expressiveness are HLR10, and GZCMZ12-1, because they use the threshold monotonic language.

Finally, the schemes that feature the smallest ciphertext size are ZH10 and ZHW13, with an overhead of only $2|\mathbb{G}|$. They are the only two schemes that do not need to embed a $\mathbb{G}_T$ element in the ciphertext. Indeed, in decryption, the data consumer combines the elements of its decryption key and the ciphertext to produce a $\mathbb{G}_T$ element, which is the symmetric key used to encrypt the actual message. However, ZH10 and ZHW13 have poor expressiveness, as they use the $\text{AND}^*_{\pm}$ and the AND^*_m languages, respectively.

A general technique for improving the expressiveness of the $\text{AND}_{\pm}$, $\text{AND}^*_{\pm}$, AND_m, and AND^*_m access structure languages is to provide *redundancy* of decryption keys in KP-ABE or redundancy of ciphertexts in CP-ABE. For example, in a KP-ABE scheme, a single consumer could hold three different decryption keys. From the access structure point of view, this is like binding the consumer to an AND / OR gate access structure: the root node is an OR, and the three branches are the single decryption keys. This technique can benefit KP-ABE schemes such as ALP11: consumers simply hold two or more different keys, which can better describe their access rights. In an equivalent example for the CP-ABE paradigm, a producer could create and transmit three ciphertexts for every piece of information to be encrypted. Even though this technique aims at improving the expressiveness, in CP-ABE this is detrimental to the producer bandwidth (and computation) performance, since for a single piece of data, the producer must transmit (and compute) two or more different ciphertexts. We investigate this technique in our simulations in Section VIII.

Be aware that some schemes, e.g., JSMG18-1 and JSMG18-2, achieve a small ciphertext size, but they are not very bandwidth efficient. The ciphertext itself is indeed small, but the producer must create and transmit additional cryptographic material along with each ciphertext to make the scheme work. Indeed, such schemes provide an unusual technique for updating the policy of an existing ciphertext, and this feature comes with a cost in terms of producer bandwidth. They can either add (JSMG18-1) or remove (JSMG18-2) the required positive value of an attribute from the policy embedded in the ciphertext, from a minimum of 1 to a maximum of m attributes inside a single policy. However, to do so, they must upload to the data storage $(m - |\mathcal{P}|)|\mathbb{G}|$ more bits along with the ciphertext, increasing further the encryption bandwidth overhead (being $|\mathcal{P}|$ the number of attributes used inside the policy).

As a side note, the schemes EMNOS09, LGRDY12, and PYS14 are the only schemes with constant-size ciphertext that have been formally proved to be secure under standard assumptions.

B. Efficient Key Management

As outlined in Section VI, there are three different key revocation mechanisms: direct, indirect, and attributewise. We aim to identify the mechanism that reliably impacts the least the producer bandwidth.

An example of an attributewise revocation that is not reliably convenient for the producer is YWRL10. Indeed, we recall that when the key authority wants to revoke a set of attributes λ embedded in a decryption key, it generates a new version of the public parameters for the attributes in λ. This means that a producer has to download a number of elements in $\mathbb{G}$ equal to λ after each revocation. Notably, the required bandwidth can be optimized if the producer maintains up-to-date only a subset of the public parameters (e.g., a sensor that encrypts sensed data always under the same attribute set). In this case, the producer has to download a number of elements in $\mathbb{G}$ from 0 to λ after each revocation, depending on how many attributes it uses are inside λ. This is surely the most unpredictable mechanism in terms of required bandwidth since its traffic depends on the number of attributes involved in the revocation. On the other hand, the schemes HN11 and LYHZS17 also feature an attributewise revocation mechanism, but the producers are not required to download anything after a revocation happens. In fact, a producer will always encrypt data using the same public parameters, and then it uploads the ciphertext on the data storage, which enforces the revocation.

In the direct revocation mechanism, usually, the producer must hold a list of identifiers of the revoked users. Typically, this is the only cost sustained by producers in terms of bandwidth for key management operations. The basic idea is to create a "trapdoor" in the ciphertext by using the identifiers of all the revoked consumers. Such a trapdoor "activates" when a revoked consumer tries to decrypt the ciphertext with its decryption key and, therefore, its identifier. The trapdoor hides a needed quantity to decrypt the ciphertext, so this step cannot be avoided or cheated in any way. The only cost in terms of producer bandwidth is, therefore, the revocation list. The more the system runs, the more consumers will be revoked: downloading an entire revocation list each time can be detrimental for the producer bandwidth and may also impact its limited storage capabilities. The direct revocable scheme LYZL18 (see Section VI-A) keeps the revocation list short by removing expired decryption keys to lessen key management bandwidth overhead on the producer.

Finally, in the indirect revocation mechanism, the producer potentially does not have to download anything. For example, in BGK08-4 and QZZC17, the producer contributes to revocation only by encrypting the ciphertext with an additional attribute referring to the time of encryption. This is a task that requires no interaction with the other parties and saves bandwidth. Therefore, this approach is very convenient for the producers since they neither have to download updated public parameters after each revocation nor have to hold an updated copy of the revocation list.

C. Small Group Elements

Using small group elements in the ciphertext can reduce the bandwidth overhead due to the ciphertext size. A viable strategy is to adapt ABE schemes to use Type III pairing by converting the highest number of $\mathbb{G}$ elements to $\mathbb{G}_1$ elements in the ciphertext, as discussed in Section V. Smaller $\mathbb{G}_1$ elements are convenient to compute more efficiently some operations in encryption, and they also save the producer a conspicuous amount of bandwidth. Using fewer bits for a single $\mathbb{G}_1$ group element dramatically reduces the bandwidth needed to upload a ciphertext to the data storage.

For example, we can compare a ciphertext of the GPSW06-3 KP-ABE scheme with a ciphertext of the AC17-5 scheme, which represents a possible Type III conversion of GPSW06-3. To do so, we use the standard Type I (curve `a.param`) and Type III (curve `d201.param`) curves of the PBC library[4] with 80-bit security and 20 attributes in the ciphertext. The resulting encryption bandwidth overhead of GPSW06-3 is 1408 bytes, while that of AC17-5 is 654 bytes, leading to a 53.6 % overhead reduction for *every* ciphertext uploaded to the data storage.

In other schemes, the bandwidth saving is less pronounced. For example, we compare the BSW07 CP-ABE scheme with the AC17-3 scheme, which is a possible Type III conversion of BSW07, with the same curves as before and 20 attributes in the ciphertext. The resulting encryption bandwidth overhead of BSW07 is 2752 bytes, while that of AC17-3 is 2237 bytes, leading to an 18.7 % overhead reduction.

Note that, in contrast with the strategy of adopting schemes with constant-size ciphertexts (Section VII-A), this strategy can improve bandwidth efficiency without jeopardizing the expressiveness of the policies. Note also that the two strategies can be applied together to gain even more efficiency.

VIII. Experimental Evaluation

In this section, we evaluate, through an event-based MATLAB simulator, the performance of a variety of ABE schemes that we described in the previous sections. Such a quantitative comparison clarifies several aspects regarding performance, and it allows us to provide a better insight about the goodness of the various ABE schemes.

TABLE II
KPIs of Simulated Schemes

Scheme Name	Producer CPU Efficiency	Key Authority Bandwidth Efficiency	Producer Bandwidth Efficiency
BGK08-4		✓	✓
AI09		✓	
YWRL10		✓	
ZH10			✓
HW14	✓		
AC17-1	✓		✓
AC17-5	✓		✓
QZZC17		✓	✓
LYZL18		✓	✓

For each strategy discussed—except for “alternative mathematics”—we selected a scheme to represent it. All the simulated schemes excel in one or two KPIs, but none of them in all the three KPIs at once. The simulated KP-ABE schemes are BGK08-4, AI09, YWRL10, HW14-1, and AC17-5. The simulated CP-ABE schemes are ZH10, HW14-2, AC17-1, QZZC17, and LYZL18. Table II shows in which KPI(s) each selected scheme excels. The schemes BKG08-4 and QZZC17 implement indirect key revocation and efficient key management: producers are not affected by revocations. ZH10 is the scheme with the smallest constant-size ciphertext. AC17-1 and AC17-5 use Type III pairings and have ciphertexts with small group elements. YWRL10 implements attributewise revocation. AI09 implements a hybrid revocation. HW14 exploits encryption offloading to reduce producers’ computational load. LYZL18 implements direct key revocation and efficient key management.

Each simulated scheme comprises a key authority, a data storage, many producers, and many consumers. The simulator runs for a simulated period of time within which it randomly generates four types of events: data production, data consumption, consumer join, and key revocation. The corresponding algorithm of each scheme, e.g., **Encrypt**, **Decrypt**, etc., is simulated within these events. The simulator neither performs actual math operations nor implements some protocol for exchanging messages between the entities. Rather, it records the number and type of math operations and eventually estimates the total computational load for each entity. Moreover, the simulator records the number and the size of the messages exchanged between the entities and estimates their experienced traffic overhead.

A. Simulator Description and Configuration

The simulator simulates a generic architecture as the one of Fig. 1, in which: 1) producers produce ciphertexts and upload them on the data storage; 2) consumers obtain ciphertexts from the data storage and decrypt them; 3) the key authority

[4] https://crypto.stanford.edu/pbc

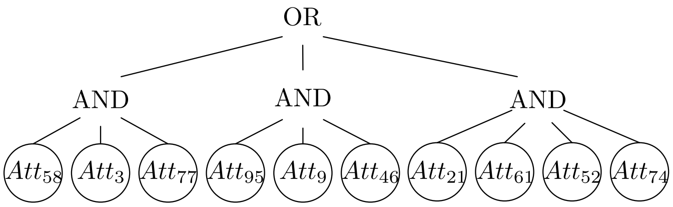


Fig. 5. Example of simulated access policy in DNF shape.

TABLE III
PAIRING-BASED CRYPTOGRAPHY BENCHMARKS ON ZOLERTIA RE-MOTE. FOR EACH OPERATION, 100 REPETITIONS ARE AVERAGED AND 95 %-CONFIDENCE INTERVALS ARE COMPUTED

Type I pairing	**time (ms)**	
	Mean	95 % CI
$\mathbb{G}$ point-scalar multiplication	265	4.05×10^{-3}
$\mathbb{G}_T$ modular exponentiation	74	1.48×10^{-3}
bilinear pairing	5673	0.80×10^{-3}
hash ($\mathcal{H} : \{0,1\}^* \to \mathbb{G}$)†	6088	2.17×10^{-3}

Type III pairing	**time (ms)**	
	Mean	95 % CI
$\mathbb{G}_1$ point-scalar multiplication	73	1.22×10^{-3}
$\mathbb{G}_2$ point-scalar multiplication	763	9.56×10^{-3}
$\mathbb{G}_T$ modular exponentiation	288	4.84×10^{-3}
bilinear pairing	9195	0.71×10^{-3}
hash ($\mathcal{H} : \{0,1\}^* \to \mathbb{G}_1$)	225	1.00×10^{-3}

† From our experiments, this value is largely independent of the input size.

generates decryption keys for joining consumers; and also 4) revokes decryption keys.

The simulator defines a universe of 100 attributes for each scheme. Scheme-specific attributes, such as the dummy attribute (used in YWRL10) or time attributes (used in BGK08-4, AI09, and QZZC17), are not considered in this number but are individually added to the schemes that need them. In the simulation of KP-ABE schemes, each ciphertext is labeled with a set of 30 distinct attributes chosen randomly among those in the universe, and each decryption key embeds an access policy consisting of 10 distinct attributes chosen randomly among those in the universe. In a dual way, in the CP-ABE schemes, a ciphertext embeds an access policy of 10 distinct random attributes, and a decryption key is labeled with 30 distinct random attributes. Within the same simulation, the access policy shape is fixed for all the ciphertexts (in CP-ABE) or decryption keys (in KP-ABE). We set the access policy shape to be in Disjunctive Normal Form (DNF), with an OR at the root and three AND children with three, three, and four attributes, respectively. Fig. 5 shows an example.

The simulator can be configured to start with an initial number of producers and consumers. In our simulations, we start with 10 000 producers and 10 000 consumers. The number of producers remains the same throughout the simulation, while the number of consumers varies every time a consumer join event or a key revocation event occurs. During a preliminary phase, the simulator creates an initial database of 10 000 ciphertexts and decryption keys for the consumers according to the methodology described above. As far as decryption key generation is concerned, we make sure that each decryption key can decrypt at least one of the initial ciphertexts.

After these preliminary operations, the simulator starts generating the events and recording the metrics. The events of data production, data consumption, consumer join, and key revocation are modeled as Poisson processes. More in detail, each producer generates a data production event every hour on average, each consumer generates a data consumption event every hour on average, the key authority generates a key revocation event every day on average, and a consumer join event is generated every day on average. At each data production event, we simulate, for each scheme, that the producer encrypts a new piece of data and uploads the ciphertext on the data storage. The new ciphertext is created according to the methodology described above. At each data consumption event, we simulate, for each scheme, that a random consumer downloads a random ciphertext from the data storage (among those its key is allowed to decrypt) and decrypts it. At each key revocation event, we simulate, for each scheme, that the key authority revokes a random consumer from the system. At each consumer join event, we simulate, for each scheme, that the key authority generates a new decryption key according to the methodology described above, encrypts it with the consumer's public key, and uploads it to the data storage; then, the consumer downloads and decrypts it. These events are executed until the simulated period of time, which we set to one month, is reached.

In the final phase of the simulation, the simulator averages the recorded metrics to obtain the results per single producer and consumer. The metrics, i.e., computational load and traffic overhead, are expressed in units of time and units of storage, respectively. We assume all producers to be IoT devices, i.e., Zolertia RE-Motes [34], which is a platform mounting an ARM Cortex M3 microcontroller, operating at 32 MHz. To determine the running time of the various basic math operations, we perform benchmarks that use the PBC library on such a device. We suppose that all the simulated schemes that employ symmetric pairing use a Type I pairing with 512-bit $\mathbb{G}$ group elements and 1024-bit $\mathbb{G}_T$ group elements, while those that employ asymmetric pairing use a Type III pairing with 201-bit $\mathbb{G}_1$ group elements, 603-bit $\mathbb{G}_2$ group elements, and 1206-bit $\mathbb{G}_T$ group elements.[5] These curves give an equivalent security level of 80 bits. Table III shows the results of our benchmarks.

The simulator performs 30 repetitions with the same configuration but with different random seeds to achieve statistically sound results. The final results are averaged, and confidence intervals are computed.

[5] These curves come with the PBC library and their parameters can be found in the files `a.param` and `d201.param`.

B. Simulated Schemes

In the following, we give details on scheme-specific configurations.

The AI09 scheme allows producers to arbitrarily choose whether to encrypt the ciphertext in "direct revocation mode" or in "indirect revocation mode." We modeled this opportunity with a random choice during the data production event, and we simulated two variants of the AI09 scheme. In the first variant, called AI09(H), where (H) stands for hybrid, we set the probability of producing a ciphertext in direct revocation mode to 0.5. In the second variant, called AI09(D), where (D) stands for direct, we force the scheme to act as a pure direct revocation scheme by setting this probability to 1. We neglect the pure indirect variant because it is very similar to the BGK08-4 scheme. For the schemes that use the binary tree, i.e., BGK08-4, QZZC17, and AI09, we create a complete binary tree of the minimum size that can accommodate the initial consumers and the joining consumers. Moreover, we set the duration of the time period to one day. At the end of each time period, which we model through a periodic event, the key authority creates a key update and stores it on the data storage.

The ZH10 scheme allows only access policies composed of an AND gate on Boolean attributes with wildcards ($\text{AND}^{*}_{\pm}$). For a fair comparison, we improve the ZH10 expressiveness using redundancy. We realize DNF-shaped policies by encrypting the same piece of data for a number of times equal to the number of AND gates in the original DNF. Therefore, the producer creates three ciphertexts: each ciphertext specifies three or four positive attributes, and the remaining attributes of the universe are wildcards. On the other hand, a decryption key contains 30 positive attributes and 70 negative attributes. We simulate that the producer uploads three ciphertexts on the data storage, and the consumer downloads only the one it can decrypt.

For the HW14-1 and HW14-2 schemes, we simulate that the offline phase is outsourced to a trusted resourceful device, and the resulting preprocessed quantities are transmitted to the producers through some secure channel. When the online phase –which comes at no cost for the producer– is completed, the producer uploads the ciphertext on the data storage.

As regards the schemes that do not come with a revocation mechanism, i.e., ZH10, HW14-1, HW14-2, AC17-1, and AC17-5, we implemented for them the naive revocation mechanism described in Section IV-F. For these schemes, during the consumer join event, the key authority generates as many decryption keys for the joining consumer as the number of key revocations occurred so far. All these keys have the same access policy (or attribute set), but each one has been generated with a different master key. In this way, the consumer can access ciphertexts generated before its joining time.

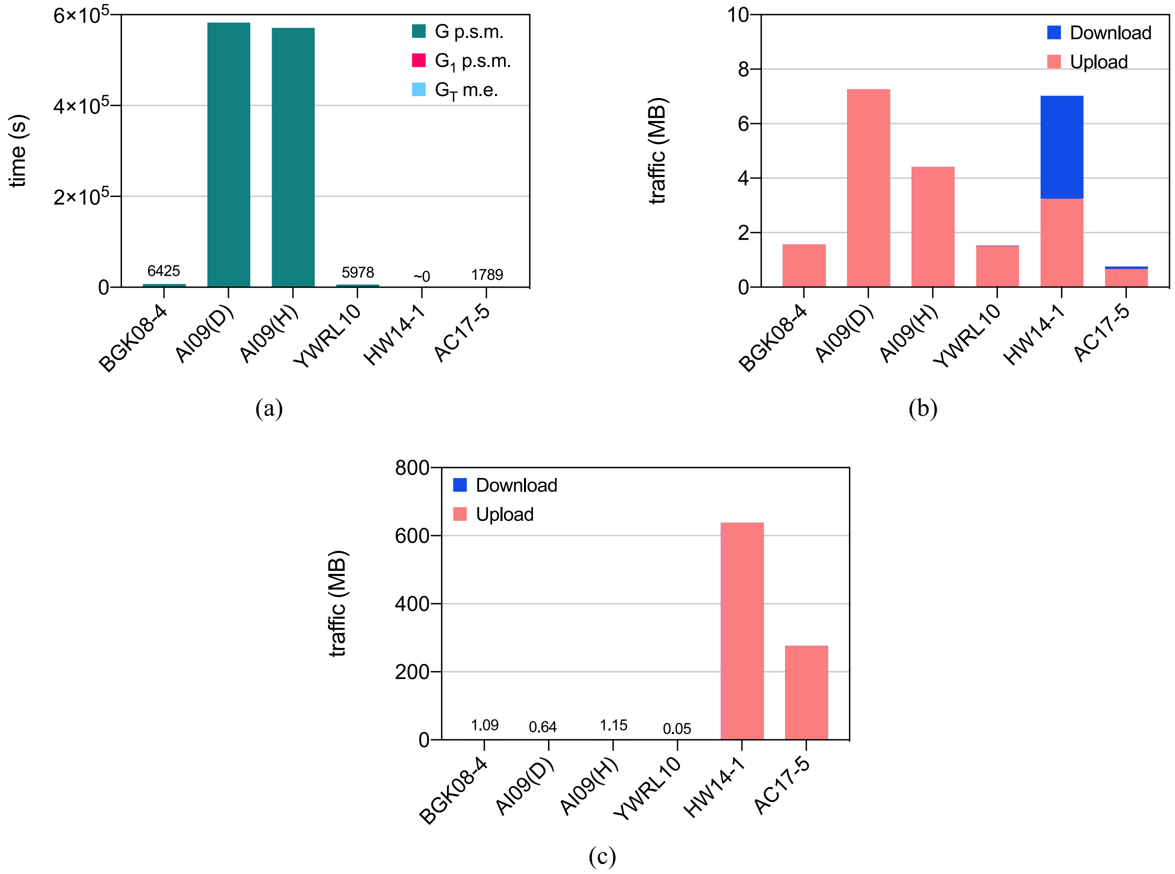
C. Discussion

In the following, we show the performance of the selected ABE schemes against the KPIs. We treat KP-ABE and CP-ABE schemes separately since such two paradigms are not meaningfully comparable. We first analyze the results obtained by simulating the KP-ABE schemes, and later we focus on CP-ABE.

Fig. 6 shows average values and 95 %-confidence intervals of the KPIs for the simulated KP-ABE schemes. We note that, concerning the producer's performance, the best scheme is AC17-5 which can keep both the CPU load and bandwidth overhead low by taking advantage of the asymmetric pairing. Indeed, as we notice from Table III, a point-scalar multiplication in $\mathbb{G}_1$ is roughly four times faster than a point-scalar multiplication in $\mathbb{G}$. Moreover, being a $\mathbb{G}_1$ group element smaller than a $\mathbb{G}$ element, the producer bandwidth for AC17-5 is the lowest among all the schemes tested, albeit the producer must download at each revocation event the public parameters because of the naive revocation. The schemes BGK08-4 and YWRL10 turn out to be slightly less efficient about the producer. However, if we look at the key authority bandwidth efficiency, we notice that these schemes handle the key revocation very efficiently, while in the AC17-5 scheme, at each revocation the key authority is burdened with the creation of new public parameters and new decryption keys for nonrevoked consumers. The AI09 scheme, which is revocable, performs well only concerning the key authority bandwidth efficiency. Indeed, the direct revocation mode used in both AI09(D) and AI09(H) reduces the producer's efficiency since it must perform more computations and generate larger ciphertexts than when indirect revocation mode is used.

In the HW14-1 scheme, the producer does not perform burdensome operations thanks to encryption outsourcing. It only executes few modular multiplications in $\mathbb{Z}_p$, which are enough time efficient to be negligible, therefore we do not simulate them. On the other hand, the producer must download the preprocessed quantities, which heavily impacts its bandwidth efficiency. Also, the key authority bandwidth overhead is high because of the naive revocation. Note how in revocable schemes, the key authority is efficient in terms of bandwidth, namely, two orders of magnitude more efficient than nonrevocable schemes.

Fig. 7 shows average values and 95 %-confidence intervals of the KPIs for the simulated CP-ABE schemes. We note that the producer CPU load of HW14-2 is negligible as it happens for its KP-ABE counterpart HW14-1. Unlike the previous simulation, the simulated asymmetric-pairing scheme (AC17-1) is not the one with the lowest computational load concerning the producer. In this scheme, the computation of the hashes heavily impacts the performance, and they are not precomputable. With a policy of ten attributes as in our simulation, at each encryption, the producer computes 60 hashes and spends about 13.5 s just for computing hash values; the advantages of asymmetric pairing are therefore nullified. Note that this contrasts with the results in [75], which reports hashes to be very fast and, thus, AC17-1 to be very efficient in encryption. Such results were obtained on a PC-class device and not on constrained devices. On the contrary, we experimentally noted that hashes are quite slow on the Zolertia RE-Mote platform. This suggests that AC17-1, though very efficient on PCs, loses much of its efficiency when adopted in IoT applications.



(a)

(b)

(c)

Fig. 6. Comparison of KP-ABE schemes performance with regard to the three KPIs. (a) Producer CPU load. (b) Producer bandwidth overhead. (c) Key authority bandwidth overhead.

In our simulations, the scheme that performs best concerning the producer efficiency is ZH10. Even though the producer generates three ciphertexts instead of one at each data production (for expressiveness fairness), ZH10 features the best producer CPU efficiency and a good producer bandwidth efficiency. In Fig. 7(b), its pronounced download overhead is due to the naive revocation. Compared to the other schemes with naive revocation, i.e., HW14-2 and AC17-1, the ZH10 scheme performs worse. This is because, unlike HW14-2 and AC17-1, ZH10 is a small-universe scheme, and thus public parameters are large. If we look at Fig. 7(c), we note that ZH10 has the worst key authority bandwidth efficiency. We recall that in the naive revocation, the key authority generates public parameters but also new decryption keys for nonrevoked consumers. Compared to HW14-2 and AC17-1, in ZH10, the key authority must generate larger keys.

The revocable schemes LYZL18 and QZZC17 are light in terms of key authority bandwidth overhead. About the producer, the LYZL18 scheme is more efficient, albeit it implements a direct revocation mechanism, and the producers must perform additional computations to enforce revocation. However, we recall that the revocation list in the ciphertext is condensed into a single $\mathbb{G}$ element, which also helps to keep the bandwidth overhead on the producer very low [Fig. 7(b)]. In the QZZC17 scheme, the producer experiences a slightly higher CPU load and bandwidth overhead. Nonetheless, this scheme is particularly suitable for applications in which a high efficiency on the consumer is needed (see Sections IX-B and IX-C).

IX. Accessory Performance Indicators

A. *Producer Storage Efficiency*

We identified in the literature three main strategies to improve the storage efficiency of the producer: 1) adopting large-universe schemes [45], [50], [52]; 2) storing partially the public parameters (e.g., the majority of small-universe schemes can adopt this strategy); and 3) using storage-efficient key revocation mechanisms [81].

In an ABE scheme, the minimum amount of information a producer must store is the public parameters, whose size depends on the universe type. Usually, large-universe schemes have small public parameters, while small-universe schemes have large public parameters. In particular, in small-universe schemes, the size of the public parameters grows linearly with the number of attributes in the universe. On the contrary, in large-universe schemes, the size of the public parameters is typically constant and composed of a few group elements. For example, in the GPSW06-1 (small-universe) scheme initialized with a universe of 100 attributes, the size of the public parameters is 6528 bytes, while in the BSW07 (large-universe) scheme, the size of the public parameters is only 320 bytes.[6]

[6] Considering a security level of 80 bits.

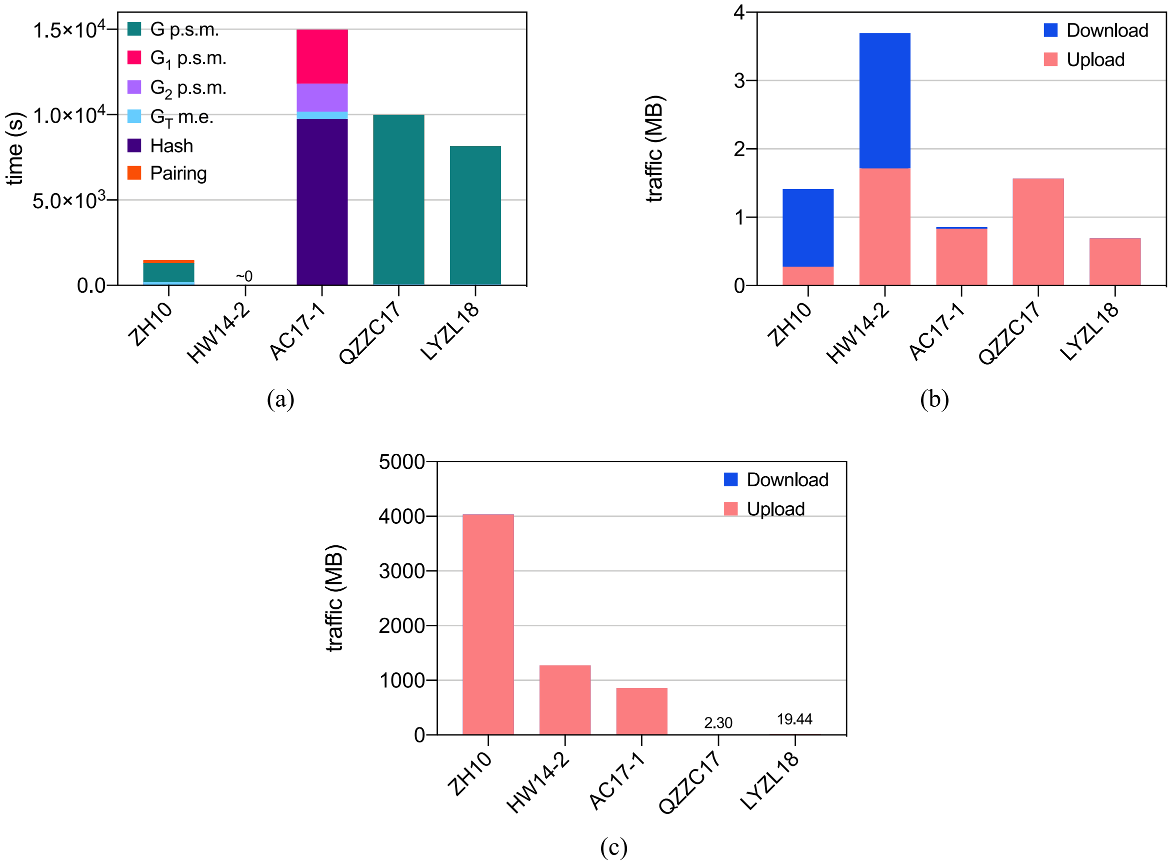


Fig. 7. Comparison of CP-ABE schemes performance with regard to the three KPIs. (a) Producer CPU load. (b) Producer bandwidth overhead. (c) Key authority bandwidth overhead.

Therefore, a viable strategy to keep the storage overhead low on the producer is to choose a large-universe scheme that features small and constant-size public parameters.

Regarding the small-universe schemes, we identify a general strategy that can be applied to plenty of such schemes to reduce the required storage on the producer: storing only a portion of the public parameters. In small-universe schemes, the public parameters typically include one $\mathbb{G}$ element (component) for each attribute in the universe. If the producer uses only a portion of them, e.g., it always encrypts with the same set of attributes, it can store only the components it uses for encryption. Referring to the previous example, if a producer uses only 30 attributes out of 100, the storage overhead can be reduced from 6528 to 2048 bytes. Unfortunately, this strategy is not viable for the vast majority of the schemes with constant-size ciphertext that we surveyed in Section VII. By their very nature, to create a ciphertext of constant size, these schemes require computations that involve each and every part of the public parameters.

The revocation mechanism also can impact the producer storage. Specifically, in direct revocation, each producer must store the revocation list, which contains the revoked identifiers. An identifier can be expressed as a group element or as a mere progressive number. In both cases, the producer storage might be severely affected when the system reaches a large number of revoked consumers. Therefore, using an indirect revocation mechanism typically leads to a better producer storage efficiency. In direct revocation, to relieve the storage overhead on the producers, some schemes, e.g., LYZL18, embed an expiration date in the decryption keys. In this way, the expired decryption keys can be excluded from the revocation list.

B. Consumer CPU Efficiency

The computation efficiency of an ABE scheme on the data consumers includes the operations the consumers perform to decrypt data and those they perform for key management procedures. As already outlined, the key management operations are typically much less frequent, so, in this section, we focus on decryption efficiency to represent the overall consumer CPU efficiency. We identified in the literature three main strategies to improve the efficiency of decryption on the consumer: 1) outsourcing burdensome decryption operations [72], [78], [79], [93]; 2) using constant-complexity decryption [27], [65]; and 3) adopting mathematics alternative to pairings, for example, ECC [71] or RSA [77] mathematics.

The first strategy leverages data storage with abundant computational capabilities, for example, a cloud server. This is necessary because, at each data request from the consumers, the data storage must perform some operations on the requested ciphertext before transmitting it. In CDLQ16, QZZC17, and CM21 schemes, the data storage uses public information related to the requesting consumer *id* to transform the ciphertext so that only the consumer *id* can decrypt it. The transformed ciphertext is sent to the consumer, which finalizes the decryption at a low and constant cost, i.e., one operation in $\mathbb{G}_T$ in CDLQ16 and two pairings in both QZZC17 and CM21. Note that since the data storage does not hold any secret information to transform the ciphertext, it can be untrusted. Additionally, anyone can verify the correctness of the transformation by performing the same algorithm executed by the data storage. In HWY18, too, the decryption is outsourced to the data storage that transforms the ciphertext and leaves just one modular exponentiation in $\mathbb{G}_T$ to the consumer for retrieving the plaintext. Recently, the work in [93] proposes a generalization method to include the possibility of decryption outsourcing in any ABE scheme. The authors claim—providing a game-based proof—that the outsourced version of an ABE scheme does not affect the security of the original scheme whatsoever. Moreover, they propose an innovative technique to allow parallel computation during the decryption operation, resulting in a much reduced latency in retrieving the transformed ciphertext. In all these schemes, the decryption performed by the consumer is independent of the attributes in its decryption key and is performed in constant time.

The second strategy tries to reduce the cost of the decryption algorithm. Very often, its complexity grows linearly with the number of attributes used to satisfy the access policy. However, for many schemes with limited expressiveness (see schemes surveyed in Section VII), the cost of decryption is low and constant, and it is usually dominated by a few pairings. For example, in the schemes DJ14, LGRDY12, EMNOS09, and ZZCLL14, the decryption cost is fixed to two pairings. In many cases, limited expressiveness results in a good CPU efficiency for consumers.

The third strategy is to adopt mathematics different from pairings, e.g., ECC (YCT15 and OD16) and RSA (ODKCJ17), which lighten the consumer CPU because they eliminate the burdensome pairing operation usually employed in decryption. However, at the time of writing, we do not suggest their employment for the security concerns explained in Section V.

C. Consumer Bandwidth Efficiency

We state in Section VII that the main tasks involving the producer bandwidth overhead are the encryption bandwidth overhead and the key management bandwidth overhead. This applies to the bandwidth overhead of the consumer as well. Two of the three strategies described in Section VII that improve the producer bandwidth efficiency are also good for improving the consumer bandwidth efficiency: 1) using schemes with constant-size ciphertexts and 2) using small group elements. Furthermore, we identified in the literature two additional strategies to improve the efficiency of the consumer bandwidth: 1) partially outsourcing the decryption to the data storage [72], [78] and 2) using a direct key revocation mechanism [70], [81].

In the first additional strategy, since the consumer must download the ciphertext from the data storage, we can use this intermediary to lighten the burden on the consumer. Oftentimes, the data storage (typically in the form of a cloud server) manipulates the stored ciphertexts in some way before sending them to the consumers. Indeed, in schemes like CDLQ16, QZZC17, and CM21, the cloud storage does shrink the ciphertext size before sending it to a consumer: from an arbitrary size (that grows linearly with the size of the access policy) to a constant size of $|\mathbb{G}_T| + 2|\mathbb{G}|$. This strategy can be effective for the consumer's bandwidth. However, we must point out that not always such a manipulation reduces the number of bytes that the consumer has to download. In fact, HN11 requires the data storage to do some operations over a ciphertext before sending it to the requesting consumer. Among other operations, the data storage has to prepend some additional information to the ciphertext, therefore *increasing* the number of bytes that the consumer must download.

The second additional strategy is to use a direct revocation mechanism. Indeed, the direct revocation mechanism (e.g., LYZL18) is the most effective since the consumer does not have to download any cryptographic quantity to update its decryption key. In contrast, indirect revocation and attribute-wise revocation impact the consumer, which must download key update material, either periodically or at each revocation event.

X. Conclusion & Future Directions

In this article, we surveyed ABE schemes and solutions suitable for IoT applications. We analyzed various schemes under the three KPIs: producer CPU efficiency, producer bandwidth efficiency, and key authority bandwidth efficiency. We then identified and described the strategies that the state-of-the-art schemes adopt to improve these performance indicators. Finally, we assessed the efficiency of some prominent ABE schemes by thorough simulations.

From this survey, many challenges emerged in designing an ABE scheme suitable for IoT. To address such challenges, researchers can investigate the several strategies that we described to improve the identified KPIs and APIs. We believe that those strategies should be used as a design pattern in new IoT-oriented ABE schemes. Alternatively, research should push to find novel strategies to improve the proposed performance indicators.

In addition, we emphasized that the overall performance of a system could be significantly improved by using IoT devices with hardware acceleration specific for ABE cryptographic operations. We note that some devices, such as the Zolertia RE-Mote, are already equipped with hardware accelerators for elliptic-curve operations. However, such ECC hardware accelerators are not designed for pairing-based cryptography. Although they allow for point-scalar multiplications and group exponentiations, they do not provide pairing acceleration support. Moreover, some accelerators only support NIST-standardized curves, which are not suitable for ABE schemes. We believe the research should also push toward pairing-based crypto hardware accelerators to increase IoT devices' performance both in encryption and in decryption.

Finally, we note that schemes with additional features, such as traceability, accountability, puncturability, and post-quantum security, generally add extra costs and are often to burdensome for IoT scenarios. Indisputably, all these features are desirable for a cryptographic scheme, and we believe the research should also push toward improving such schemes and propose new ones that keep the computation and communication overhead low, particularly for the data producer.

As a final remark, we note that quantum-resistant ABE schemes are currently infeasible on IoT devices. Indeed, such schemes are built on lattices, oftentimes relying on the learning with errors (LWEs) problem. This means that the size of encryption and decryption keys are too large (typically in the order of gigabytes [25], [26]) to fit IoT devices. Since quantum computers will be available to major actors in the next decades, we believe the research should push to find more efficient ABE quantum-resistant schemes. In this way, all the frameworks built upon ABE can still be used without concerns.

References

[1] S. Yu, K. Ren, and W. Lou, "FDAC: Toward fine-grained distributed data access control in wireless sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 4, pp. 673–686, Apr. 2011.

[2] M. Aazam, S. Zeadally, and K. A. Harras, "Deploying fog computing in Industrial Internet of Things and industry 4.0," *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4674–4682, Oct. 2018. [Online]. Available: https://doi.org/10.1109/TII.2018.2855198

[3] M. La Manna, P. Perazzo, M. Rasori, and G. Dini, "FABElous: An attribute-based scheme for Industrial Internet of Things," in *Proc. IEEE Int. Conf. Smart Comput. (SMARTCOMP)*, Washington, DC, USA, 2019, pp. 33–38.

[4] Y. Li, X. Cheng, Y. Cao, D. Wang, and L. Yang, "Smart choice for the smart grid: Narrowband Internet of Things (NB-IoT)," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 1505–1515, Jun. 2018. [Online]. Available: https://doi.org/10.1109/JIOT.2017.2781251

[5] M. Rasori, P. Perazzo, and G. Dini, "A lightweight and scalable attribute-based encryption system for smart cities," *Comput. Commun.*, vol. 149, pp. 78–89, Jan. 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0140366419303913

[6] Y. Zhang, D. Zheng, and R. H. Deng, "Security and privacy in smart health: Efficient policy-hiding attribute-based access control," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 2130–2145, Jun. 2018. [Online]. Available: https://doi.org/10.1109/JIOT.2018.2825289

[7] S. Sicari, A. Rizzardi, G. Dini, P. Perazzo, M. La Manna, and 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," *Int. J. Inf. Security*, vol. 20, pp. 695–713, Nov. 2020.

[8] M. Baza, M. Nabil, N. Lasla, K. Fidan, M. Mahmoud, and M. M. Abdallah, "Blockchain-based firmware update scheme tailored for autonomous vehicles," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Marrakesh, Morocco, Apr. 2019, pp. 1–7. [Online]. Available: https://doi.org/10.1109/WCNC.2019.8885769

[9] M. La Manna, L. Treccozzi, P. Perazzo, S. Saponara, and G. Dini, "Performance evaluation of attribute-based encryption in automotive embedded platform for secure software over-the-air update," *Sensors*, vol. 21, no. 2, p. 515, 2021.

[10] Q. Wen, Y. Gao, Z. Chen, and D. Wu, "A blockchain-based data sharing scheme in the supply chain by IIoT," in *Proc. IEEE Int. Conf. Ind. Cyber Phys. Syst. (ICPS)*, Taipei, Taiwan, May 2019, pp. 695–700. [Online]. Available: https://doi.org/10.1109/ICPHYS.2019.8780161

[11] A. I. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A survey on enabling technologies, protocols, and applications," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2347–2376, 4th Quart., 2015. [Online]. Available: https://doi.org/10.1109/COMST.2015.2444095

[12] P. Kocher *et al.*, "Spectre attacks: Exploiting speculative execution," *Commun. ACM*, vol. 63, no. 7, pp. 93–101, 2020. [Online]. Available: https://doi.org/10.1145/3399742

[13] M. Lipp *et al.*, "Meltdown," 2018, *arXiv:1801.01207.*

[14] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn.*, 2005, pp. 457–473.

[15] P. S. K. Oberko, V.-H. K. S. Obeng, and H. Xiong, "A survey on multi-authority and decentralized attribute-based encryption," *J. Ambient Intell. Humanized Comput.*, vol. 13, pp. 515–533, Mar. 2021.

[16] R. R. Al-Dahhan, Q. Shi, G. M. Lee, and K. Kifayat, "Survey on revocation in ciphertext-policy attribute-based encryption," *Sensors*, vol. 19, no. 7, p. 1695, 2019.

[17] K. Edemacu, H. K. Park, B. Jang, and J. W. Kim, "Privacy provision in collaborative eHealth with attribute-based encryption: Survey, challenges and future directions," *IEEE Access*, vol. 7, pp. 89614–89636, 2019.

[18] C.-C. Lee, P.-S. Chung, and M.-S. Hwang, "A survey on attribute-based encryption schemes of access control in cloud environments," *IJ Netw. Security*, vol. 15, no. 4, pp. 231–240, 2013.

[19] C.-W. Liu, W.-F. Hsien, C. C. Yang, and M.-S. Hwang, "A survey of attribute-based access control with user revocation in cloud data storage," *IJ Netw. Security*, vol. 18, no. 5, pp. 900–916, 2016.

[20] S. Moffat, M. Hammoudeh, and R. Hegarty, "A survey on ciphertext-policy attribute-based encryption (CP-ABE) approaches to data security on mobile devices and its application to IoT," in *Proc. Int. Conf. Future Netw. Distrib. Syst.*, 2017, p. 34.

[21] L. Pang, J. Yang, and Z. Jiang, "A survey of research progress and development tendency of attribute-based encryption," *Sci. World J.*, vol. 2014, Jul. 2014, Art. no. 193426.

[22] Z. Qiao, S. Liang, S. Davis, and H. Jiang, "Survey of attribute based encryption," in *Proc. 15th IEEE/ACIS Int. Conf. Softw. Eng. Artif. Intell. Netw. Parallel/Distrib. Comput. (SNPD)*, Las Vegas, NV, USA, 2014, pp. 1–6.

[23] Y. Zhang, R. H. Deng, S. Xu, J. Sun, Q. Li, and D. Zheng, "Attribute-based encryption for cloud computing access control: A survey," *ACM Comput. Surv.*, vol. 53, no. 4, pp. 1–41, 2020.

[24] Y. Zhang, X. Chen, J. Li, H. Li, and F. Li, "Attribute-based data sharing with flexible and direct revocation in cloud computing," *KSII Trans. Internet Inf. Syst.*, vol. 8, no. 11, pp. 4028–4049, 2014.

[25] J. Li, C. Ma, and K. Zhang, "A novel lattice-based ciphertext-policy attribute-based proxy re-encryption for cloud sharing," in *Security and Privacy in Social Networks and Big Data*, W. Meng and S. Furnell, Eds. Singapore: Springer, Jul. 2019, pp. 32–46.

[26] W. Dai *et al.*, "Implementation and evaluation of a lattice-based key-policy ABE scheme," *IEEE Trans. Inf. Forensics Security*, vol. 13, pp. 1169–1184, 2018. [Online]. Available: https://doi.org/10.1109/TIFS.2017.2779427

[27] X. Li, D. Gu, Y. Ren, N. Ding, and K. Yuan, "Efficient ciphertext-policy attribute based encryption with hidden policy," in *Proc. Int. Conf. Internet Distrib. Comput. Syst.*, 2012, pp. 146–159.

[28] W. Viriyasitavat, T. Anuphaptrirong, and D. Hoonsopon, "When blockchain meets Internet of Things: Characteristics, challenges, and business opportunities," *J. Ind. Inf. Integr.*, vol. 15, pp. 21–28, Sep. 2019.

[29] W. Viriyasitavat, L. Da Xu, Z. Bi, and D. Hoonsopon, "Blockchain technology for applications in Internet of Things—Mapping from system design perspective," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8155–8168, Oct. 2019.

[30] A. Arena, P. Perazzo, and G. Dini, "Virtual private ledgers: Embedding private distributed ledgers over a public blockchain by cryptography," in *Proc. 23rd Int. Database Appl. Eng. Symp.*, New York, NY, USA, 2019, pp. 1–9. [Online]. Available: https://doi.org/10.1145/3331076.3331083

[31] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in *Proc. IEEE Infocom*, San Diego, CA, USA, 2010, pp. 1–9.

[32] E. Rescorla and N. Modadugu, "Datagram transport layer security version 1.2," IETF, RFC 6347, 2012, pp. 1–32. [Online]. Available: https://doi.org/10.17487/RFC6347

[33] "Espressif ESP32 platform datasheet," Data Sheet, Espressif Syst., Shanghai, China, 2021. [Online]. Available: https://bit.ly/2qW8yj1

[34] "Zolertia RE-Mote Platform Datasheet." 2017. [Online]. Available: https://bit.ly/2OkilYY

[35] H. Li, G. Chen, Y. Wang, Y. Gao, and W. Dong, "Accurate performance modeling of uplink transmission in NB-IoT," in *Proc. IEEE 24th Int. Conf. Parallel Distrib. Syst. (ICPADS)*, Singapore, 2018, pp. 910–917.

[36] J. Tosi, F. Taffoni, M. Santacatterina, R. Sannino, and D. Formica, "Performance evaluation of Bluetooth low energy: A systematic review," *Sensors*, vol. 17, no. 12, p. 2898, 2017.

[37] B. Latré, P. De Mil, I. Moerman, N. Van Dierdonck, B. Dhoedt, and P. Demeester, "Maximum throughput and minimum delay in IEEE 802.15.4," in *Proc. Int. Conf. Mobile Ad-Hoc Sens. Netw.*, 2005, pp. 866–876.

[38] O. Georgiou and U. Raza, "Low power wide area network analysis: Can LoRa scale?" *IEEE Wireless Commun. Lett.*, vol. 6, no. 2, pp. 162–165, Apr. 2017.

[39] K. L. Lueth *et al.*, "IoT Analytics Research—State of IoT summer 2021," IoT Analytics GmbH, Hamburg, Germany, Rep., Mar. 2022. [Online]. Available: https://iot-analytics.com/product/state-of-iot-summer-2021/

[40] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," in *Proc. Annu. Int. Cryptol. Conf.*, 2001, pp. 213–229.

[41] S. D. Galbraith, K. G. Paterson, and N. P. Smart, "Pairings for cryptographers," *Discrete Appl. Math.*, vol. 156, no. 16, pp. 3113–3121, 2008.

[42] A. Menezes, P. Sarkar, and S. Singh, "Challenges with assessing the impact of NFS advances on the security of pairing-based cryptography," in *Proc. Int. Conf. Cryptol. Malaysia*, 2016, pp. 83–108.

[43] A. Lewko and B. Waters, "Decentralizing attribute-based encryption," in *Advances in Cryptology (EUROCRYPT)*, K. G. Paterson, Ed. Berlin, Germany: Springer, May 2011, pp. 568–588.

[44] Z. Liu, Z. Cao, and D. S. Wong, "Efficient generation of linear secret sharing scheme matrices from threshold access trees," IACR Cryptol. ePrint Arch., Lyon, France, Rep. 2010/374, 2010.

[45] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE Symp. Security Privacy (SP)*, Berkeley, CA, USA, 2007, pp. 321–334.

[46] E. Fujisaki and T. Okamoto, "Secure integration of asymmetric and symmetric encryption schemes," in *Advances in Cryptology (CRYPTO)*, M. Wiener, Ed. Berlin, Germany: Springer, Aug. 1999, pp. 537–554.

[47] T. V. X. Phuong, G. Yang, and W. Susilo, "Poster: Efficient ciphertext policy attribute based encryption under decisional linear assumption," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2014, pp. 1490–1492.

[48] Z. Zhou, D. Huang, and Z. Wang, "Efficient privacy-preserving ciphertext-policy attribute based-encryption and broadcast encryption," *IEEE Trans. Comput.*, vol. 64, no. 1, pp. 126–138, Jan. 2015.

[49] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," IACR Cryptol. ePrint Archive, Lyon, France, Rep. 2006/309, 2006. [Online]. Available: https://eprint.iacr.org/2006/309

[50] R. Ostrovsky, A. Sahai, and B. Waters, "Attribute-based encryption with non-monotonic access structures," in *Proc. 14th ACM Conf. Comput. Commun. Security*, 2007, pp. 195–203.

[51] A. Boldyreva, V. Goyal, and V. Kumar, "Identity-based encryption with efficient revocation," in *Proc. ACM Conf. Comput. Commun. Security (CCS)*, Alexandria, VA, USA, Oct. 2008, pp. 417–426. [Online]. Available: https://doi.org/10.1145/1455770.1455823

[52] N. Attrapadung and H. Imai, "Attribute-based encryption supporting direct/indirect revocation modes," in *Cryptography and Coding*, M. G. Parker, Ed. Berlin, Germany: Springer, Dec. 2009, pp. 278–300.

[53] M. La Manna, P. Perazzo, and G. Dini, "SEA-BREW: A scalable attribute-based encryption revocable scheme for low-bitrate IoT wireless networks," *J. Inf. Security Appl.*, vol. 58, May 2021, Art. no. 102692. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2214212620308413

[54] M. Rasori, P. Perazzo, G. Dini, and S. Yu, "Indirect revocable KP-ABE with revocation undoing resistance," *IEEE Trans. Services Comput.*, early access, Apr. 8, 2021, doi: 10.1109/TSC.2021.3071859.

[55] K. Emura, A. Miyaji, A. Nomura, K. Omote, and M. Soshi, "A ciphertext-policy attribute-based encryption scheme with constant ciphertext length," in *Proc. Int. Conf. Inf. Security Pract. Exp.*, 2009, pp. 13–23.

[56] J. Herranz, F. Laguillaumie, and C. Ràfols, "Constant size ciphertexts in threshold attribute-based encryption," in *Proc. Int. Workshop Public Key Cryptogr.*, 2010, pp. 19–34.

[57] A. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters, "Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption," in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn.*, 2010, pp. 62–91.

[58] Z. Zhou and D. Huang, "On efficient ciphertext-policy attribute based encryption and broadcast encryption," IACR Cryptol. ePrint Arch., Lyon, France, Rep. 2010/395, 2010. [Online]. Available: https://eprint.iacr.org/2010/395

[59] N. Attrapadung, B. Libert, and E. de Panafieu, "Expressive key-policy attribute-based encryption with constant-size ciphertexts," in *Public Key Cryptography (PKC)*, D. Catalano, N. Fazio, R. Gennaro, and A. Nicolosi, Eds. Berlin, Germany: Springer, Mar. 2011, pp. 90–108.

[60] J. Hur and D. K. Noh, "Attribute-based access control with efficient revocation in data outsourcing systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 7, pp. 1214–1221, Jul. 2011. [Online]. Available: https://doi.org/10.1109/TPDS.2010.203

[61] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *Proc. Int. Workshop Public Key Cryptogr.*, 2011, pp. 53–70.

[62] A. Ge, R. Zhang, C. Chen, C. Ma, and Z. Zhang, "Threshold ciphertext policy attribute-based encryption with constant size ciphertexts," in *Proc. Aust. Conf. Inf. Security Privacy*, 2012, pp. 336–349.

[63] A. Sahai, H. Seyalioglu, and B. Waters, "Dynamic credentials and ciphertext delegation for attribute-based encryption," IACR Cryptol. ePrint Arch., Lyon, France, Rep. 2012/437, 2012. [Online]. Available: https://eprint.iacr.org/2012/437

[64] Y. Rouselakis and B. Waters, "Practical constructions and new proof methods for large universe attribute-based encryption," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2013, pp. 463–474.

[65] N. Doshi and D. C. Jinwala, "Fully secure ciphertext policy attribute-based encryption with constant length ciphertext and faster decryption," *Security Commun. Netw.*, vol. 7, no. 11, pp. 1988–2002, 2014.

[66] S. Hohenberger and B. Waters, "Online/offline attribute-based encryption," IACR Cryptol. ePrint Arch., Lyon, France, Rep. 2014/021, 2014. [Online]. Available: https://eprint.iacr.org/2014/021

[67] L. Touati, Y. Challal, and A. Bouabdallah, "C-CP-ABE: Cooperative ciphertext policy attribute-based encryption for the Internet of Things," in *Proc. Int. Conf. Adv. Netw. Distrib. Syst. Appl. (INDS)*, Bejaia, Algeria, 2014, pp. 64–69.

[68] Y. Zhang, D. Zheng, X. Chen, J. Li, and H. Li, "Computationally efficient ciphertext-policy attribute-based encryption with constant-size ciphertexts," in *Proc. Int. Conf. Provable Security*, 2014, pp. 259–273.

[69] J. Chen, R. Gay, and H. Wee, "Improved dual system ABE in prime-order groups via predicate encodings," IACR Cryptol. ePrint Arch., Lyon, France, Rep. 2015/409, 2015. [Online]. Available: https://eprint.iacr.org/2015/409

[70] T. V. X. Phuong, G. Yang, W. Susilo, and X. Chen, "Attribute based broadcast encryption with short ciphertext and decryption key," in *Computer Security (ESORICS)*, G. Pernul, P. Y. A. Ryan, and E. Weippl, Eds. Cham, Switzerland: Springer Int., Sep. 2015, pp. 252–269.

[71] X. Yao, Z. Chen, and Y. Tian, "A lightweight attribute-based encryption scheme for the Internet of Things," *Future Gener. Comput. Syst.*, vol. 49, pp. 104–112, Aug. 2015.

[72] H. Cui, R. H. Deng, Y. Li, and B. Qin, "Server-aided revocable attribute-based encryption," in *Proc. 21st Eur. Symp. Res. Comput. Security*, vol. 9879. Heraklion, Greece, Sep. 2016, pp. 570–587. [Online]. Available: https://doi.org/10.1007/978-3-319-45741-3_29

[73] V. Odelu and A. K. Das, "Design of a new CP-ABE with constant-size secret keys for lightweight devices using elliptic curve cryptography," *Security Commun. Netw.*, vol. 9, no. 17, pp. 4048–4059, 2016. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/sec.1587

[74] L. Touati and Y. Challal, "Collaborative KP-ABE for cloud-based Internet of Things applications," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Kuala Lumpur, Malaysia, 2016, pp. 1–7.

[75] S. Agrawal and M. Chase, "FAME: Fast attribute-based message encryption," IACR Cryptol. ePrint Arch., Lyon, France, Rep. 2017/807, 2017. [Online]. Available: https://eprint.iacr.org/2017/807

[76] J. Li, W. Yao, J. Han, Y. Zhang, and J. Shen, "User collusion avoidance CP-ABE with efficient attribute revocation for cloud storage," *IEEE Syst. J.*, vol. 12, no. 2, pp. 1767–1777, Jun. 2018.

[77] V. Odelu, A. K. Das, M. K. Khan, K.-K. R. Choo, and M. Jo, "Expressive CP-ABE scheme for mobile devices in IoT satisfying constant-size keys and ciphertexts," *IEEE Access*, vol. 5, pp. 3273–3283, 2017.

[78] B. Qin, Q. Zhao, D. Zheng, and H. Cui, "Server-aided revocable attribute-based encryption resilient to decryption key exposure," in *Proc. Int. Conf. Cryptol. Netw. Security*, 2017, pp. 504–514.

[79] Q. Huang, L. Wang, and Y. Yang, "DECENT: Secure and fine-grained data access control with policy updating for constrained IoT devices," *World Wide Web*, vol. 21, no. 1, pp. 151–167, 2018.

[80] Y. Jiang, W. Susilo, Y. Mu, and F. Guo, "Ciphertext-policy attribute-based encryption supporting access policy update and its extension with preserved attributes," *Int. J. Inf. Security*, vol. 17, no. 5, pp. 533–548, 2018.

[81] J. K. Liu, T. H. Yuen, P. Zhang, and K. Liang, "Time-based direct revocable ciphertext-policy attribute-based encryption with short revocation list," in *Proc. Int. Conf. Appl. Cryptogr. Netw. Security*, 2018, pp. 516–534.

[82] S. Xu, G. Yang, and Y. Mu, "Revocable attribute-based encryption with decryption key exposure resistance and ciphertext delegation," *Inf. Sci.*, vol. 479, pp. 116–134, Apr. 2019.

[83] L. Cheng and F. Meng, "Server-aided revocable attribute-based encryption revised: Multi-user setting and fully secure," in *Proc. Eur. Symp. Res. Comput. Security*, 2021, pp. 192–212.

[84] C. F. Kerry, *Digital Signature Standard (DSS)*, FIPS Standard 186-4, 2013.

[85] S.-Y. Tan, K.-W. Yeow, and S. O. Hwang, "Enhancement of a lightweight attribute-based encryption scheme for the Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 4, pp. 6384–6395, Aug. 2019. [Online]. Available: https://doi.org/10.1109/JIOT.2019.2900631

[86] J. Herranz, "Attacking pairing-free attribute-based encryption schemes," *IEEE Access*, vol. 8, pp. 222226–222232, 2020. [Online]. Available: https://doi.org/10.1109/ACCESS.2020.3044143

[87] J. Herranz, "Attribute-based encryption implies identity-based encryption," *IET Inf. Security*, vol. 11, no. 6, pp. 332–337, 2017.

[88] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Advances in Cryptology*, G. R. Blakley and D. Chaum, Eds. Berlin, Germany: Springer, Aug. 1984, pp. 47–53.

[89] A. Joux, "A one round protocol for tripartite Diffie–Hellman," in *Proc. 4th Int. Symp. Algorithmic Number Theory (ANTS-IV)* (Lecture Notes in Computer Science), vol. 1838. Leiden, The Netherlands, Jul. 2000, pp. 385–394. [Online]. Available: https://doi.org/10.1007/10722028_23

[90] J. A. Akinyele, C. Garman, and S. Hohenberger, "Automating fast and secure translations from type-I to type-III pairing schemes," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Security*, New York, NY, USA, 2015, pp. 1370–1381. [Online]. Available: https://doi.org/10.1145/2810103.2813601

[91] A. Sahai, H. Seyalioglu, and B. Waters, "Dynamic credentials and ciphertext delegation for attribute-based encryption," in *Proc. Annu. Cryptol. Conf.*, 2012, pp. 199–217.

[92] J. H. Seo and K. Emura, "Revocable identity-based encryption revisited: Security model and construction," in *Proc. Int. Workshop Public Key Cryptogr.*, 2013, pp. 216–234.

[93] C. Feng, K. Yu, M. Aloqaily, M. Alazab, Z. Lv, and S. Mumtaz, "Attribute-based encryption with parallel outsourced decryption for edge intelligent IoV," *IEEE Trans. Veh. Technol.*, vol. 69, no. 11, pp. 13784–13795, Nov. 2020.

Marco Rasori received the master's degree in computer engineering from the University of Pisa, Pisa, Italy, in 2016, the Ph.D. degree in smart computing, a joint program from the University of Florence, Florence, Italy; the University of Pisa; and the University of Siena, Siena, Italy, in 2020.

During his Ph.D. studies, he was a Visiting Scholar with the Stevens Institute of Technology, Hoboken, NJ, USA, studying Attribute-Based Encryption Solutions for the IoT Environment. He is currently a Research Associate with the Institute of Informatics and Telematics, National Research Council, Pisa. His main research interests are in the area of cybersecurity with particular focus on design and performance evaluation of cybersecurity solutions for smart cyber–physical systems.

Michele La Manna received the bachelor's and master's degrees in computer engineering from the University of Pisa, Pisa, Italy, in 2014 and in 2017, respectively, and the Ph.D. degree in smart computing, a joint program from the University of Florence, Florence, Italy; the University of Pisa; and the University of Siena, Siena, Italy, in 2022.

He is currently a Research Scholar with the University of Pisa. His main research interests are design and performance evaluation of cybersecurity solutions for smart cyber–physical systems, focusing on solutions based on attribute-based encryption or post-quantum cryptography.

Pericle Perazzo received the master's degree (*cum laude*) in computer engineering and the Ph.D. degree in information engineering from the University of Pisa, Pisa, Italy, in 2010 and 2014, respectively.

During his Ph.D. studies, he was a Visiting Researcher with the Institute for Parallel and Distributed Systems, Stuttgart, Germany. Since 2017, he has been a Researcher with the Department of Information Engineering, University of Pisa. His research interests include the area of security and privacy in the Internet of Things, with special emphasis on attribute-based encryption, post-quantum cryptography, and blockchain technologies.

Gianluca Dini received the master's degree in electronics engineering from the University of Pisa, Pisa, Italy, in 1990, and the Ph.D. degree in computer engineering from Scuola Superiore Sant'Anna, Pisa, in 1995.

From 1993 to 1994, he was a Research Fellow with the Department of Computer Science, University of Twente, Enschede, The Netherlands. Since 1993, he has been with the Department of Information Engineering, University of Pisa, where he is currently a Full Professor. Currently, he is working on security in the Internet of Things. He has published more than 100 papers in international conferences, books, and journals and has participated, even with coordination roles, to many projects funded by the Commission of the European Community, the Italian Government, and private companies. His research interests are in the field of distributed computing systems, with particular reference to security.