# Performance Evaluation of Attribute-Based Encryption: Toward Data Privacy in the IoT

Xinlei Wang
University of California, Davis
xlwang@ucdavis.edu

Jianqing Zhang
Intel Labs
jianqing.zhang@intel.com

Eve M. Schooler
Intel Labs
eve.m.schooler@intel.com

Mihaela Ion
University of Trento
mion@google.com

***Abstract*—With the ever increasing number of connected devices and the over abundance of data generated by these devices, data privacy has become a critical concern in the Internet of Things (IoT). One promising privacy-preservation approach is Attribute-Based Encryption (ABE), a public key encryption scheme that enables fine-grained access control, scalable key management and flexible data distribution. This paper presents an in-depth performance evaluation of ABE that focuses on execution time, data and network overhead, energy consumption, and CPU and memory usage. We evaluate two major types of ABE, Key-Policy Attribute-Based Encryption (KP-ABE) and Ciphertext-Policy Attribute-Based Encryption (CP-ABE), on different classes of mobile devices including a laptop and a smartphone. To the best of our knowledge, this is the first comprehensive study of ABE dedicated solely to its performance. Our results provide insights into important practical issues of ABE, including what computing resources ABE requires in heterogeneous environments, at what cost ABE offers benefits, and under what situations ABE is best suited for use in the IoT.**

## I. Introduction

The sheer number of devices connecting to the Internet of Things (IoT) continues to grow exponentially, as does the amount of data produced. Data created in one domain (e.g., Smart Meter data in the Energy domain periodically measures total household energy usage) is now generated at high enough frequency that it is vulnerable to being mined for sensitive or confidential information (e.g., presence, behavior patterns) beyond the original purpose. Therefore data privacy is of critical importance. However traditional privacy-preserving techniques fall short in a number of ways: fine-grain access control, scalable key management, and flexible data distribution.

In traditional encryption schemes, a sender (data publisher) usually needs to know the identities of the intended recipients (data subscribers) and needs to pre-share credentials with them. The intent is that a sender encrypts data that can only be decrypted and read by an exact recipient. However, in some IoT scenarios, such as the Smart Home [1] or Smart City [2], senders may not know the identify of recipients, nor know all possible recipients who may need access to the data in the near term or the longer term. For example, a witness may publicly report a car accident via a smartphone app that submits a photo to a rescue service. Unbeknownst to the witness, the app both shares and limits distribution of the eye-witness photos with police and emergency personnel. With traditional public key mechanisms, it is also difficult to supply data access policies that include a richer set of attributes, beyond simply a specific set of recipients. Yet a Smart City may want to set a broader policy that publicizes regional accidents reports, for example to those in a particular area within a specified time period.

Attribute-Based Encryption (ABE) [4], [5] is a promising approach that can address these issues. ABE enables expressive, flexible and fine-grained data access control policies, which are the conditional statements or rules built from attributes (features, descriptors, meta-data). Although a trusted key authority is required for policy decisions, no third party is required for policy enforcement. Moreover, senders and recipients are decoupled because they do not need to pre-share secrets, which simplifies key management for large-scale and dynamic systems and which makes data distribution more flexible. In particular, in group-oriented publish-subscribe systems found in the IoT, ABE does not require a shared group key to be updated for every new group member who joins, significantly improving scalability. What's more, ABE is more strongly resistant to collusion attacks than traditional public key encryption schemes for group-based access control [5].

Given its unique benefits, ABE has recently gained much attention and has been adopted by many cloud computing applications and large-scale dynamic systems [6]–[9]. However, to the best of our knowledge, there has not been a comprehensive study on ABE performance especially focused on mobile devices, which often are clients of cloud applications that play a key role in the IoT. Due to limited computing resources and power constraints, it is important to study the actual performance of ABE on mobile devices. It is critical to understand at what cost ABE offers its benefits, and under what situations ABE is best suited for use in the IoT.

We implemented and evaluated two major types of ABE, Ciphertext-Policy Attribute-Based Encryption (CP-ABE) and Key-Policy Attribute-Based Encryption (KP-ABE), in Java. The evaluation focuses on the metrics of execution time, data and network overhead, energy consumption as well as CPU and memory usage. Since ABE aims to be used in large-scale systems with mobile devices, we tested our library on both a PC-class laptop and an Android-based smartphone, which would help us understand the requirements of ABE on platforms with different levels of computing power.

In the rest of the paper, Section II provides an overview of the essential aspects of ABE and Section III presents the performance evaluation methodology and results. We discuss our findings and future work in Section IV, highlight related work in Section V and conclude the paper in Section VI.

## II. Attribute-Based Encryption: Overview

### A. ABE Fundamentals & Implementation

An ABE system usually consists of a key authority, publishers (senders) and subscribers (recipients). The key authority authenticates publishers and subscribers (verifies they are who they say they are, as well as their attributes), generates public/private keys, and issues the keys to publishers and subscribers. There are four major algorithms in ABE: *Setup*, *KeyGen*, *Encryption* and *Decryption*. During the *Setup* stage, the key authority generates a pair of public key and master

Dr. Mihaela Ion is now affiliated with Google Inc.

978-1-4799-2003-7/14/$31.00 ©2014 IEEE

secret key based on parameters initialized from pairing-based cryptography [10], [11]. It keeps the master secret key and distributes the public key to each user in the system. The key authority then runs the *KeyGen* algorithm to generate subscribers' private keys per request. The *Encryption* and the *Decryption* algorithms are executed by publishers and subscribers respectively.

In this work, we implement and evaluate two ABE schemes: KP-ABE [4] and CP-ABE [5]. In KP-ABE, a set of data attributes are embedded in the cipher text of encrypted data, and the access policy to the data is embedded in the subscriber's private key. Only if the access policy embedded in the private key matches that of the encrypted data, then the subscriber can decrypt the data. In the *regional accident report* example in Section I, a requester is assigned a private key by the key authority, which embeds the access policy, say ***year***=*2013* **AND** ***zip***=*95054*. The city council can limit access to data by encrypting reports with specific attributes (*year* and *zip*). Only reports with the specified *year* and *zip* can be decrypted by the requester. In CP-ABE, the publisher's data access policy is embedded in the ciphertext, and a subscriber's attributes are associated with its private keys. A subscriber can decrypt the ciphertext only if the attributes associated with its private key satisfy the access policy embedded in the encrypted data. In the *accident witness* example, the witness encrypts the data with a key, which is derived from a policy that the ciphertext can be decrypted only if the recipient is *police* **OR** *emergency-personnel*. Then only the recipient whose private key is associated with the roles (attributes) can decrypt the report. Generally speaking, CP-ABE is conceptually similar to Role-Based Access Control, while KP-ABE is more similar to Attribute-Based Access Control.

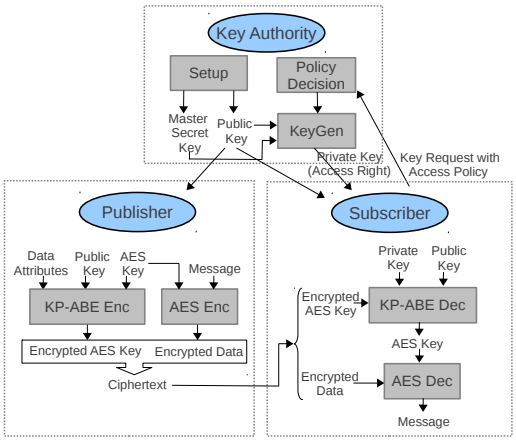
Figure 1 illustrates the major operations of KP-ABE and CP-ABE in our implementation. The essential difference between the two schemes is whether the attributes are used to describe data or subscribers. Note that the *Encryption* algorithm of CP-ABE actually consists of two steps: (1) convert a descriptive policy into an access tree; (2) encrypt the data by embedding the access tree into the ciphertext. Both steps involve expensive cryptographic computations. To accurately analyze how much computing resource each demands, we separate them in our implementation and the performance evaluation. The first step is referred to as *EncKeyGen* since the access tree can be considered an encryption key.

Since ABE is expected to be considerably more expensive than a symmetric key encryption, the data payload is never directly encrypted with ABE. Instead, we use AES symmetric encryption with 128-bit keys for data encryption, and the AES keys are in turn encrypted/decrypted with ABE and sent together with the cipher text. This also is a common practice for most public key encryption schemes. The details of our implementation can be found in [12].

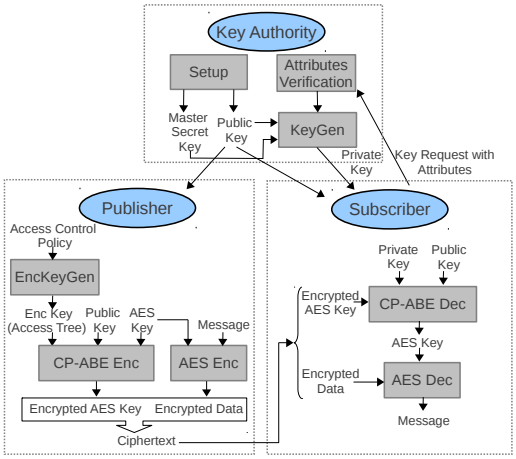
### B. Experiment Factors: Attributes & Security Strength

Two important factors largely affect the performance of ABE: *number of attributes* and *security strength*. We studied their impact on the performance metrics in our evaluation.

*1) Attributes number:* Attributes form the basis of ABE. To understand the impact of the number of attributes on the performance, we obtained testing results by varying the number of attributes. In a general IoT application, *e.g.*, a




(a) KP-ABE




(b) CP-ABE

Fig. 1: Operations of KP-ABE and CP-ABE

Smart Home or Smart City, we believe that the number of attributes involved in each instance of data encryption does not exceed 30 in most cases. Hence, our testing was done for up to 30 attributes. Our results are sufficient to project how the performance would change if more attributes are involved.

In addition, the length of a single attribute and the representation of attributes also have impact on the ABE performance. However, since attributes are hashed before being used in the expensive computations, that impact is extremely subtle, so we do not explicitly show it in our work.

*2) Security strength:* As one can imagine, stronger security requires longer primitives in cryptographic computations, which incurs more overhead. To measure the correlation between security level and performance, we examine the security levels ABE offers and what determines the ABE security level.

Because ABE is built on top of pairing-based cryptography, its security strength is determined by the underlying pairing-based cryptographic algorithm, which in turn is determined by the bit-length of the parameters [10], [11]. Suppose $E$ is the underlying elliptic curve used by the pairing algorithm, and $E$ is defined over a finite field $\mathbb{F}_q$. The parameters that determine the security strength of the pairing include the *field size* $q$ and the *prime order* $r$ of the base-point $P \in E(\mathbb{F}_q)$ ($r$ does not divide $q$) as well as the *embedding degree* $k$, which is the multiplicative order of $q$ modulo $r$. We use Type A pairing in the jPBC library [13] which has a fixed $k = 2$ [11].

TABLE I: Security Levels of ABE

| Security level (in bits) | 80 | 112 | 128 |
|---|---|---|---|
| Bit-length of $r$ for ABE | 160 | 224 | 256 |
| Bit-length of $q$ for ABE | 512 | 1024 | 1536 |
| Equivalent RSA key size (in bits) | 1024 | 2048 | 3072 |

It is considered one of the most efficient pairing algorithms available in the jPBC library. Therefore, the two parameters we can adjust to achieve different security levels are $r$ and $q$.

According to [10] and [14], with a fixed $k = 2$, the bit-length choices of $r$ and $q$ and their equivalent security levels in terms of symmetric key encryption and RSA are summarized in Table I. Starting with the 80-bit security level, which is sufficient for most medium-security purposes [15], we test three common security levels in our evaluation.

## III. Performance Evaluation

The performance metrics evaluated are execution time, data and network overhead, energy consumption, CPU and memory usage. Number of attributes and security strength levels are the two main factors considered. Generally, ABE is used to encrypt AES keys, which in turn encrypt data content. Thus, our evaluation results are based on ABE operations over a 128-bit AES key. Since mobile devices play a key role in the IoT, we test ABE on both a PC-class laptop and an Android smartphone. The laptop runs 64-bit Ubuntu 12.04 with a 1.60GHz Intel Quad-Core i7 2677M CPU and 4GB RAM. The smartphone runs Android 4.04 with a 1.60GHz Intel Atom Z2460 and 1GB RAM.

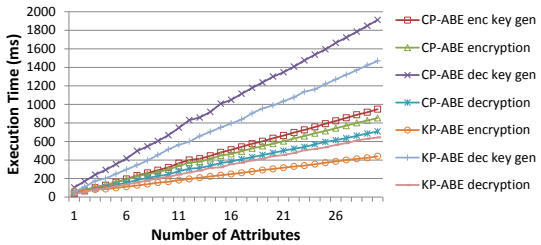
### A. Execution Time

The most important metric is the execution time for each cryptographic operation. We tested the execution time of the three major cryptographic operations of KP-ABE and CP-ABE, which are run by a key authority or a user: the *decryption key generation* (*i.e.*, the *KeyGen* algorithm), *encryption* and *decryption*. As mentioned previously, for CP-ABE, the *encryption key generation* operation is separated from *encryption*, to clearly differentiate the computing resource demands.

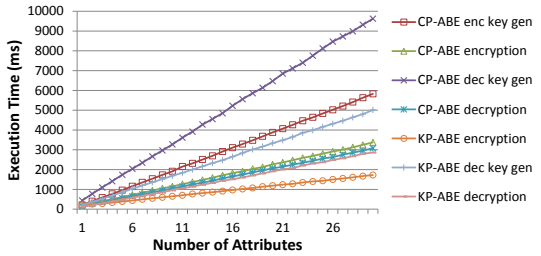
*1) Laptop:* Figure 2 shows the execution time of the major ABE operations with three different security levels. As expected, the execution time of each operation increases linearly with the number of attributes. Another observation is that KP-ABE is faster than CP-ABE for all operations, because CP-ABE performs more exponentiation computations.

Even up to 30 attributes, most operations need less than 1 second at the 80-bit security level, except for the decryption key generation for both forms of ABE. Assuming fewer than 10 attributes will be used in applications, and considering that decryption key generation operations are performed by the key authority, ordinary users should not experience noticeable delay caused by ABE. However, the delay becomes considerable at higher security levels. At the 128-bit security level, it takes several seconds to run an operation for the general case with 10 attributes. Thus, we recommend ABE for applications that demand medium-security on average computing devices although performance could be slightly improved by additional implementation optimizations (see Section IV).

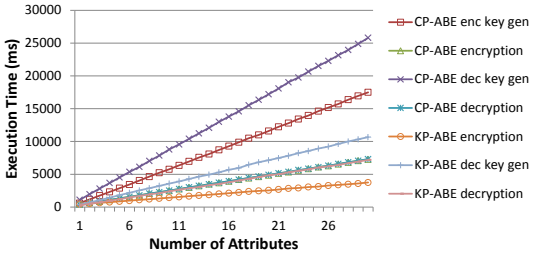
To provide an intuition how ABE compares with a public key encryption scheme in terms of computation overhead, we tested the execution time of the popular RSA scheme (textbook RSA) under the same computing environment with




(a) 80-bit security level




(b) 112-bit security level




(c) 128-bit security level

Fig. 2: Execution time of ABE on a laptop

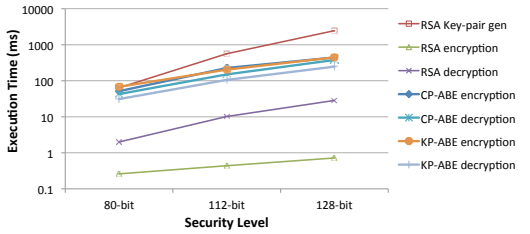
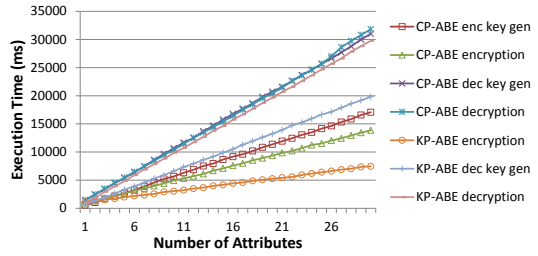


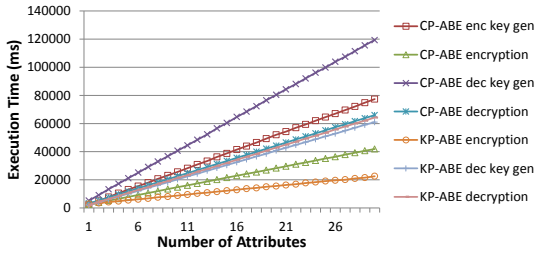

Fig. 3: Execution time of RSA compared to single-attribute ABE on a laptop

the corresponding security level. Results are shown in Figure 3 with a log-scale vertical axis. We also added the encryption and decryption delays of ABE with a single attribute on the figure for comparison. It is expected that RSA operations take less time to run because its modular exponentiation calculation is much lighter than pairing operations used in ABE. Notably, the encryption and decryption operations of RSA that take place on the client end are extremely light-weight compared to ABE, though the later provides more flexible access control.

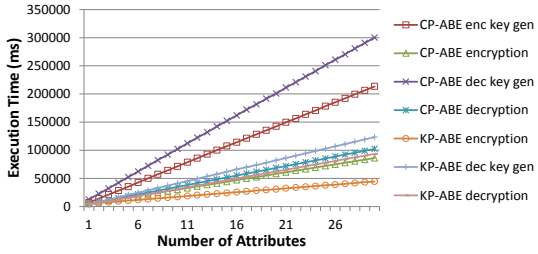
*2) Smartphone:* Figure 4 shows ABE execution time on an Atom-based Android phone. Due to the limited computing power available on the mobile phone, the performance is over 10 times worse than on the laptop. This delay becomes unacceptable even with the minimal security level. A potential solution is to re-use the AES encryption keys so that ABE encryption and decryption do not need to be executed for every message. These operations could be performed in background in an amortized manner when the AES key is refreshed. We




(a) 80-bit Security Level

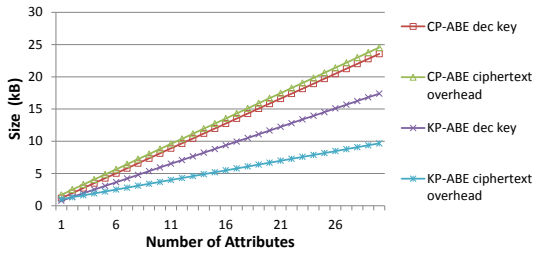



(b) 112-bit Security Level

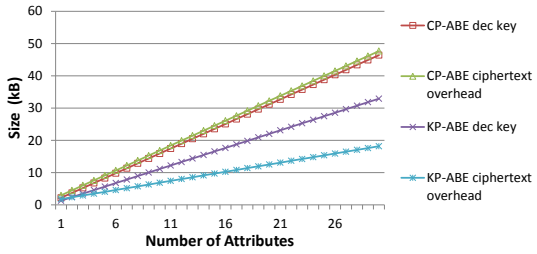



(c) 128-bit Security Level

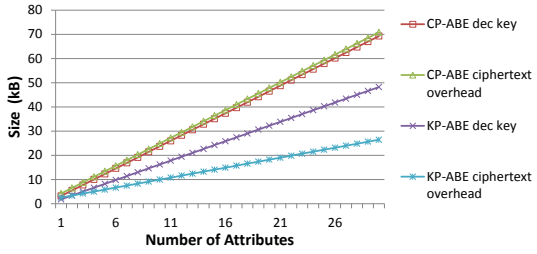
Fig. 4: Execution time of ABE on an Android phone




(a) 80-bit Security Level




(b) 112-bit Security Level




(c) 128-bit Security Level

Fig. 5: Data overhead of ABE

have implemented and are investigating the performance.

An unexpected observation is that the decryption execution time of both forms of ABE are slower than decryption key generations for 80-bit security level, while faster at the other two levels. This is echoed in the energy consumption (see Section III-C). Our hypotheses include possibly suboptimal curve selection and/or suboptimal compiler mapping of code to hardware. Further investigation is in progress.

### B. Data Overhead

There are two types of communication overhead. The first is the decryption key, which is transferred from the key authority to clients. The second is the additional payload for the encrypted AES key, which is transferred between publishers and subscribers. As described in Section II-A, to reduce the cost of ABE, the data payload is always encrypted with AES first and then the AES key is encrypted with ABE. As a block cipher with 128-bit key, AES in our implementation introduces at most 16 bytes of overhead. Therefore, we ignore the overhead caused by AES and simply consider the encrypted data as payload. Furthermore, since the AES key size is fixed in our evaluation, the data overhead introduced by encrypted ABE is independent of the data payload size.

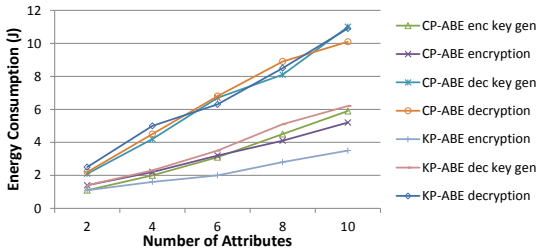
Figure 5 shows data overhead in terms of the numbers of attributes. As expected, both encryption key size and ciphertext overhead increase with the two factors: number of attributes and security level. Data overhead increases with the number of attributes because attributes are translated into an access tree, which grows in size as a function of attributes and which is embedded in the encryption key and the ciphertext. The impact of the security level relates to the larger primitives used in the computation for higher security levels and thus result in larger data overhead. In the figures, the resulting data size ranges from a few to tens of kilobytes for general use cases. The actual extra network delay caused by the data overhead is dependent on the network data rate, which in turn depends on specific networking technologies and the network conditions. Given most current networking technologies (*e.g.*, WiFi, Cellular, WSN, etc.), users should not experience a noticeable delay for getting a decryption key or transmitting data of such size. For a network with huge amounts of messages flowing, the total extra burden to the network may be considerable. For short messages that only contain brief text or content, the overhead could be larger than the messages themselves.

To make a comparison, we also obtained the data overhead of RSA. Our results indicate that RSA incurs minimal data overhead (always less than 1KB for all three security levels).

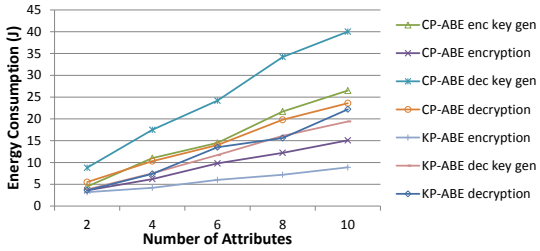
Note Java object serialization introduces additional padding data in our Java implementation, so the final size of the data objects sent over the network could be larger than the actual content and the size of such Java serialization overhead depends on the size of the object being serialized.

### C. Energy Consumption

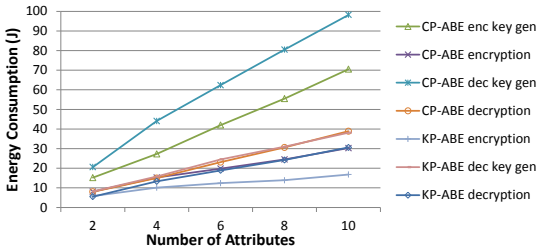
Due to battery limitations, energy consumption is a major concern for applications running on mobile phones or other IoT relevant devices. This section presents results of energy consumption testing on an Atom-based Android phone by measuring each operation of two ABE forms, using PowerTutor [16], a popular power monitor tool for Android devices.




(a) 80-bit Security Level




(b) 112-bit Security Level




(c) 128-bit Security Level

Fig. 6: Energy consumption of ABE on an Android phone

For simplicity, we measure the three security levels with 2, 4, 6, 8, and 10 attributes respectively (see Figure 6). We argue that it provides sufficient data to understand the proportional impact of larger numbers of attributes and security levels. Cross referencing these results with execution time results, we observe that energy consumption is approximately proportional to execution time. This is expected since all ABE cryptographic operations demand about the same computing power and thus execution time is linked to energy consumption. We know a fully charged mobile phone battery with a capacity of 1800mAh/3.7V can provide over 20,000J of energy. Our tests show opening a webpage on the phone consumes 2-5J of energy depending on network speed. We can see the energy consumption rate (energy consumption / execution time) of ABE is not particularly higher than other applications. Also note, as a client, mobile devices usually only need to perform encryption/decryption operations, whereas energy-consuming operations like decryption key generation may be performed by the key authority or a server, avoiding burden on smartphones.

### *D. CPU and Memory Utilization*

For CPU utilization on the laptop, we obtain the results programmatically by inserting measurement statements in the code and measure the CPU load when the ABE operations are being executed. We obtained fairly consistent results (average ∼30% load, ∼10% deviation) for each operation in Table II.

On Android, we measured the ABE library CPU load with the Dalvik Debug Monitor Server (DDMS). Again, we obtained consistent results for all ABE operations. Figure 7 shows a snapshot of the DDMS CPU monitoring result when

TABLE II: CPU load of ABE on a laptop

| Operation | | Ave. | Std. Dev. |
|---|---|---|---|
| CP-ABE | Enc. key gen. | 28.90% | 10.10% |
| | Dec. key gen. | 28.50% | 9.30% |
| | Encryption | 28.20% | 9.50% |
| | Decryption | 29.00% | 9.40% |
| KP-ABE | Dec. key gen. | 28.70% | 8.90% |
| | Encryption | 28.30% | 9.90% |
| | Decryption | 28.50% | 10.30% |

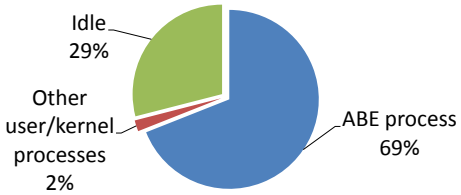



Fig. 7: CPU load of ABE on an Android phone

ABE is running. Apparently, the CPU load is platform-specific and it takes more CPU load to execute our ABE library when the computational power of the device is more limited.

We measured memory usage with DDMS as well. About 120K data objects are allocated and 12-14MB of memory are used during both ABE forms. Such memory usage is fairly acceptable considering the memory size of today's computing devices. For the same core Java code, the difference of memory usage on PC and Android is trivial.

## IV. Summary and Discussion

Our performance analysis has covered the execution time, data overhead, energy consumption and CPU/memory utilization of ABE on both a PC-class laptop and an Android smartphone. Table III highlights our evaluation results. According to the results, laptop users should not experience noticeable performance issues with ABE using the 80-bit security level, assuming a reasonable number of attributes (less than 10). However, ABE performance becomes unacceptable with higher security levels or when used on a smart phone. Thus, without significant improvement, the classic ABE algorithms are best used when the computing device has relatively high computing power and the applications demand low to medium security.

TABLE III: Summary of Evaluation Results

| Metric | Laptop | Smartphone |
|---|---|---|
| Execution Time | < 1s for most operations with 80-bit security level; > 10s with 10+ attr. and 128-bit security level. | ∼10 times slower than on laptop. |
| Data Overhead | Up to tens of KBs | |
| Energy Consumption | N/A | ∼2J/s (comparable to other regular applications). |
| CPU Load | ∼30% | ∼70% |
| Mem. Usage | 12-14MB | |

Compared with RSA, ABE's overall performance is more challenged. This is expected as pairing operations needed in ABE are roughly 20 times slower than modular exponentiation needed by RSA. However, we reiterate that the objectives and capabilities of ABE and RSA are very different. As there is no simple way to make a fair comparison, our intention is to provide an intuition about ABE performance compared to RSA. The motivation for ABE is to enable flexible and fine-grained access control and to offer scalable key management because senders and receivers are completely decoupled.

Our performance results incur extra overhead of program

execution on JVM, which is largely affected by code quality. Although we continue to optimize our ABE library code, because we want to use ABE on mobile devices without noticeable performance issues, we are focusing on key caching and re-use, background execution, outsourcing expensive computations to the cloud [17], [18], as well as tracking more efficient ABE schemes being proposed by the research community.

## V. Related Work

Besides the classic KP-ABE [4] and CP-ABE [5] used in this paper, there are a number of different extensions of ABE. Ostrovsky *et. al.* proposed an attribute-based encryption with non-monotonic access structure [19]. The access structure in the user's private key can represent any type of attributes such as negative ones, *e.g.*, "**NOT** *employee*". Muller *et. al.* introduced the concept of distributed ABE (DABE) [20]. In contrast to the classic ABE schemes where all secret keys are distributed by one central key authority, an arbitrary number of parties can be present to maintain attributes and their corresponding secret keys. Emura *et. al.* [21] proposed a CP-ABE construction with a constant ciphertext size. However, their scheme suffers from less expressive and less flexible access control policies. Wang *et. al.* proposed a hierarchical ABE (HABE) scheme [7] by integrating a hierarchical identity-based encryption scheme (HIBE) and CP-ABE, to achieve a full delegation mechanism between attribute authorities.

Most ABE research efforts are focused on describing cryptographic constructions rather than performance evaluation. Several works [7], [21]–[23] briefly discussed and compared the performance of different ABE schemes, but their discussions are only limited to Big-O analysis of execution time and data sizes in terms of number of attributes. A few other works [5], [12], [24] actually implemented KP-ABE and/or CP-ABE and presented the performance results. In contrast, our performance evaluation is much more comprehensive. In particular, we looked at the performance of ABE more from a practical point of view. To the best of our knowledge, it is the first study of the performance of ABE that considers possible security levels, on both laptop and mobile phone platforms.

## VI. Conclusion

We implemented a Java library for ABE and presented performance measurement and analysis results of its execution time, data overhead, energy consumption and CPU/memory usage. Both KP-ABE and CP-ABE were tested on a PC-class laptop and an Android-based smartphone. The performance of ABE closely correlates with the number of attributes used and the security parameters chosen. Our evaluation reveals that ABE yields acceptable results (less than a second execution time) in a laptop environment with low security strength, but becomes challenged (a few to tens of seconds execution time) for a mobile phone or with a higher security level. As a result, we propose potential solutions for using ABE on devices with limited computing power, such as key re-use and remote key generation. We believe ABE is crucial for achieving data privacy in the IoT, in contexts such as Smart Homes [1] and Smart Cities [2] due to its advantages over traditional public key encryption; fine-grain access control, scalable key management and flexible data distribution. As such, we strongly recommend further investigation into improved ABE performance, in hopes that ABE can be utilized more fully in diverse IoT computing environments with resource-constrained devices.

## References

[1] J. Zhang, Q. Li, and E. M. Schooler, "iHEMS: An Information-Centric Approach to Secure Home Energy Management," in *Proc. of IEEE SmartGridComm*, 2012.

[2] M. Ion, J. Zhang, and E. M. Schooler, "Toward content-centric privacy in icn: Attribute-based encryption and routing," in *Proc. 3rd ACM SIGCOMM Workshop on ICN*, ICN '13, pp. 39–40, 2013.

[3] T. Hardjono and B. Weis, "The Multicast Group Security Architecture." RFC 3740 (Informational), Mar. 2004.

[4] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. of ACM CCS*, pp. 89–98, 2006.

[5] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. of IEEE SP*, pp. 321–334, 2007.

[6] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in *Proc. of IEEE INFOCOM*, pp. 1–9, 2010.

[7] G. Wang, Q. Liu, and J. Wu, "Hierarchical attribute-based encryption for fine-grained access control in cloud storage services," in *Proc. of ACM CCS*, pp. 735–737, 2010.

[8] M. Li, S. Yu, K. Ren, and W. Lou, "Securing personal health records in cloud computing: Patient-centric and fine-grained data access control in multi-owner settings," in *Security and Privacy in Communication Networks*, pp. 89–106, Springer, 2010.

[9] Z. Wan, J. Liu, and R. H. Deng, "Hasbe: A hierarchical attribute-based solution for flexible and scalable access control in cloud computing," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 2, pp. 743–754, 2012.

[10] N. Koblitz and A. Menezes, *Pairing-based cryptography at high security levels*. Springer, 2005.

[11] B. Lynn, *On the implementation of pairing-based cryptosystems*. PhD thesis, Stanford University, 2007.

[12] M. Ion, *Security of Publish/Subscribe Systems*. PhD thesis, University of Trento, 2013. http://eprints-phd.biblio.unitn.it/993/.

[13] "The Java pairing based cryptography library (jPBC)." http://gas.dia.unisa.it/projects/jpbc/docs/curvegenerator.html.

[14] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid, "Recommendation for key management–part 1: General (revision 3)," *NIST special publication*, vol. 800, p. 57, 2011.

[15] J. W. Bos, M. E. Kaihara, T. Kleinjung, A. K. Lenstra, and P. L. Montgomery, "On the security of 1024-bit rsa and 160-bit elliptic curve cryptography.," *IACR Cryptology ePrint Archive*, p. 389, 2009.

[16] "PowerTutor - a power monitor for Android-based mobile platforms." http://ziyang.eecs.umich.edu/projects/powertutor/.

[17] M. Green, S. Hohenberger, and B. Waters, "Outsourcing the decryption of abe ciphertexts.," in *Proc. of USENIX Security*, p. 3, 2011.

[18] J. Li, C. Jia, J. Li, and X. Chen, "Outsourcing encryption of attribute-based encryption with mapreduce," in *Information and Communications Security*, pp. 191–201, Springer, 2012.

[19] R. Ostrovsky, A. Sahai, and B. Waters, "Attribute-based encryption with non-monotonic access structures," in *Proc. of ACM CCS*, pp. 195–203, ACM, 2007.

[20] S. Müller, S. Katzenbeisser, and C. Eckert, "Distributed attribute-based encryption," in *Information Security and Cryptology*, pp. 20–36, Springer, 2009.

[21] K. Emura, A. Miyaji, A. Nomura, K. Omote, and M. Soshi, "A ciphertext-policy attribute-based encryption scheme with constant ciphertext length," in *Information Security Practice and Experience*, pp. 13–23, Springer, 2009.

[22] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *Public Key Cryptography*, pp. 53–70, Springer, 2011.

[23] C.-C. Lee, P.-S. Chung, and M.-S. Hwang, "A survey on attribute-based encryption schemes of access control in cloud environments," *International Journal of Network Security*, vol. 15, pp. 231–240, 2013.

[24] Y. Zheng, *Privacy-Preserving Personal Health Record System Using Attribute-Based Encryption*. PhD thesis, Worcester Polytechnic Institute, 2011.