Science & Technology Development Journal – Engineering and Technology, 3(S1):SI17-SI27

Open Access Full Text Article

Research Article

Applying attribute-based encryption on mobile devices

Nhan Tam Dang[1], Hai Duong Le[2], Son Thanh Le[1], Ha Manh Tran[2,*]

ABSTRACT

The 21st century has witnessed the rapid development of small and convenient mobile devices such as smartphones, tablets, game players, sensor nodes, etc. The rise of such mobile devices indicates the increase of colossal data transmission through the Internet and online services along with the challenges of data security. It is common to think of a solution to protect sensitive data from unauthorized users, and the most popular solution is to use encryption. While many research activities in functional encryption have widely been applied to network devices, computers, and applications, mobile devices still attract much attention to security issues due to the limitations of system resources, connectivity, data transmission and power consumption that malicious users can exploit to launch attacks. Especially, mobile devices have become a principal tool to share data on the Internet through online services, such as Facebook, Youtube, DropBox, Amazon, Online Games, etc. This paper presents a study of the Attribute-Based Encryption (ABE) scheme that exploits user attributes to build the secret key and the ciphertext. ABE encryption is specified by a set of attributes or a policy defining attributes that users possess. The paper also describes a few implementations of ABE applied in the cryptography community and the challenges of integrating ABE into real-world applications. Finally, the paper proposes an implementation of ABE for Android mobile devices. This implementation associated with the Kerberos protocol can be applied to secured data sharing applications. The Kerberos protocol aims at providing mutual authentication for the client server model. Experiments have evaluated the proposed ABE implementation on Android mobile devices along with the Kerberos system. The evaluation also includes ABE performance with discussions and lessons learned.

Key words: Applied Cryptography, Attribute-Based Encryption, Secured Data Sharing, Mobile Devices, Mobile Computing

[1]*International University, Vietnam National University Ho Chi Minh City, Vietnam*

[2]*Hong Bang International University, Ho Chi Minh City, Vietnam*

Correspondence

Ha Manh Tran, Hong Bang International University, Ho Chi Minh City, Vietnam

Email: hatm@hiu.vn

History

- Received: 30-7-2019
- Accepted: 26-8-2019
- Published: 17-10-2020

DOI :10.32508/stdjet.v3iSI1.518

INTRODUCTION

Many cryptographic schemes are based on the notion of the secret/private key within asymmetric cryptography or symmetric cryptography. Most applications use solely only these two methods because of many implementations with well-documented libraries, tools available. However, they fall short on the scalable factor of humongous systems where there is a large pool of users or small devices that have low computational power. New cryptographic schemes are born, including homomorphic encryption and functional encryption [1], which are said to be more secure and have much better performance compared to both asymmetric and symmetric cryptography.

With the rapid development of mobile computing technology, mobile devices become famous and get more sophisticated with a lower cost every year. The mobile software market offers a variety of applications ranging from communication, data storage, entertainment to education, economy, business. People tend to depend more and more on mobile devices and applications. Unfortunately, this also brings much attraction from attackers hoping to exploit these personal devices. Besides, the increasing popularity of services such as Dropbox, Google Drive, One Drive, Cloud Storage makes sharing data very convenient and easy. We consider these services or servers as partially trustworthy entities only. Users use mobile devices to send data to such services or servers, and thus, it is recommended for users to encrypt data before sending it. For both asymmetric and symmetric cryptography mechanisms, the receiving entity must have a secret/private key to decrypt data that is intentionally encrypted for that entity. The processes of key exchanging and identity authentication must be done securely and precisely to guarantee secured data exchange. The theoretical ABE scheme resolves these processes by exploiting user attributes when constructing the private key and the ciphertext. This scheme can also be applied to protecting communication between mobile devices and a server. In this paper, we investigate the ABE encryption scheme and apply this scheme to mobile devices for secured data sharing. The contribution of this study is thus twofold:

- Study the ABE encryption scheme and experiment few ABE implementations
- Provide an ABE implementation for Android mobile devices and evaluate the implementation on a mobile computing application using Kerberos.

We evaluated the application with multiple security features and provided discussions for the problems of the ABE scheme. The rest of the paper is structured as follows: the next section presents the overview and existing implementations of ABE. Section III presents the implementation of the ABE scheme that can be applied for Android mobile devices. Section IV reports the preliminary results with some lessons learned before the paper is concluded in Section V.

ATTRIBUTED-BASED ENCRYPTION

Attributed-Based Encryption (ABE) first took shape by Amit Sahai and Brent Waters in [2] and then later in [3]. ABE is a form of asymmetric cryptography; messages are encrypted under an arbitrary number of attributes or a policy decided by users. Users can encrypt different parts of data with different sets of attributes or policies, so the owner can now selectively share data with other users in a fine-grained way.

A policy can be interpreted as a set of rules needed to be satisfied to guarantee a successful encryption and decryption process. It is easier to understand the term of the attribute by referring to the notion in software engineer where the system actor specifies the role played by a user or any other system interacted with the main one. These actors or objects have their properties defined by using attributes. Ultimately, we could say that the according attributes signify the according to a group of people; for example, doctors have their doctor license number and name of hospital attributes. Having revolved around attributes is probably the reason why we have the name Attribute-Based Encryption.

In ABE, encryption of data is specified by a set of attributes or a policy that defines the attributes that users need to possess. For example, the policy ((doctor or nurse) and hospital = ”Mayo Clinic”) indicates that only users who are either a doctor or nurse working at Mayo Clinic can successfully access the plaintext of patients’ health records.

There are mainly two methods of ABE: Ciphertext-Policy ABE (CP-ABE) and Key-Policy (KP-ABE). Four main essential functions of CP-ABE are:

- Parameter setup: This is a randomized algorithm that takes no input other than an implicit security parameter. This function generates a random public key (PubK) and an associated secret master key (MK).
- Encryption: This is also a randomized algorithm that takes PK, the access structure (number of policies to be met for the decryption, and the message to be encrypted).
- Key generation: This function generates a private key (PrvK) by using the list of attributes that must satisfy the access structure tree to successfully decrypt a message, and generated MK during the parameter setup function.
- Decryption: The algorithm takes the ciphertext of encryption, the PubK, and the PrvK as inputs. The decryption process happens successfully if and only if the list of attributes of the decryption key satisfies the enforcement policy.

The KP-ABE is also the same as CP-ABE except for the message encrypted together with the predefined set of attributes, and the decryption key generated for the whole access structure tree used in encryption.

The difference between the two methods is whether the attributes are embedded in the data (KP-ABE) or the access structure is embedded in the data (CP-ABE) for the encryption process as shown in Figure 1. Access structure can be viewed as policy. There are some different types of access structures: threshold, tree, Linear Secret Sharing Scheme (LSSS). Sahai and Waters [2] introduced threshold, where the encryption/decryption process has the cipher-text, and the key is possibly associated with different sets of attributes. The decryption process can only happen successfully if and only if two sets of attributes overlap at least large enough as a globally defined threshold during the setup process. Goyal et al. [3] introduced tree, which is a way to secretly share the attributes of policy and then be reconstructed using Lagrange’s interpolation. Tree supports AND gates, OR gates, and arbitrary threshold. LSSS works on a matrix that describes the attributes of policy in a row-column manner.

There are fully tested ABE implementations: AndrABEn [4], DET-ABE [5], cpabe-java [6]. AndrABEn was implemented and tested on Android devices, with both Java and C programming language combined to optimize the performance on mobile devices. AndrABEn has some dependencies that are only available in Unix-based operating systems; therefore, it does not work in Windows. AndrABEN was also reported to have some failed cases when installing and running the library in Android devices. Junwei Wang developed cpabe-java as part of his Ph.D. thesis. The project was based on the cpabe, which was developed

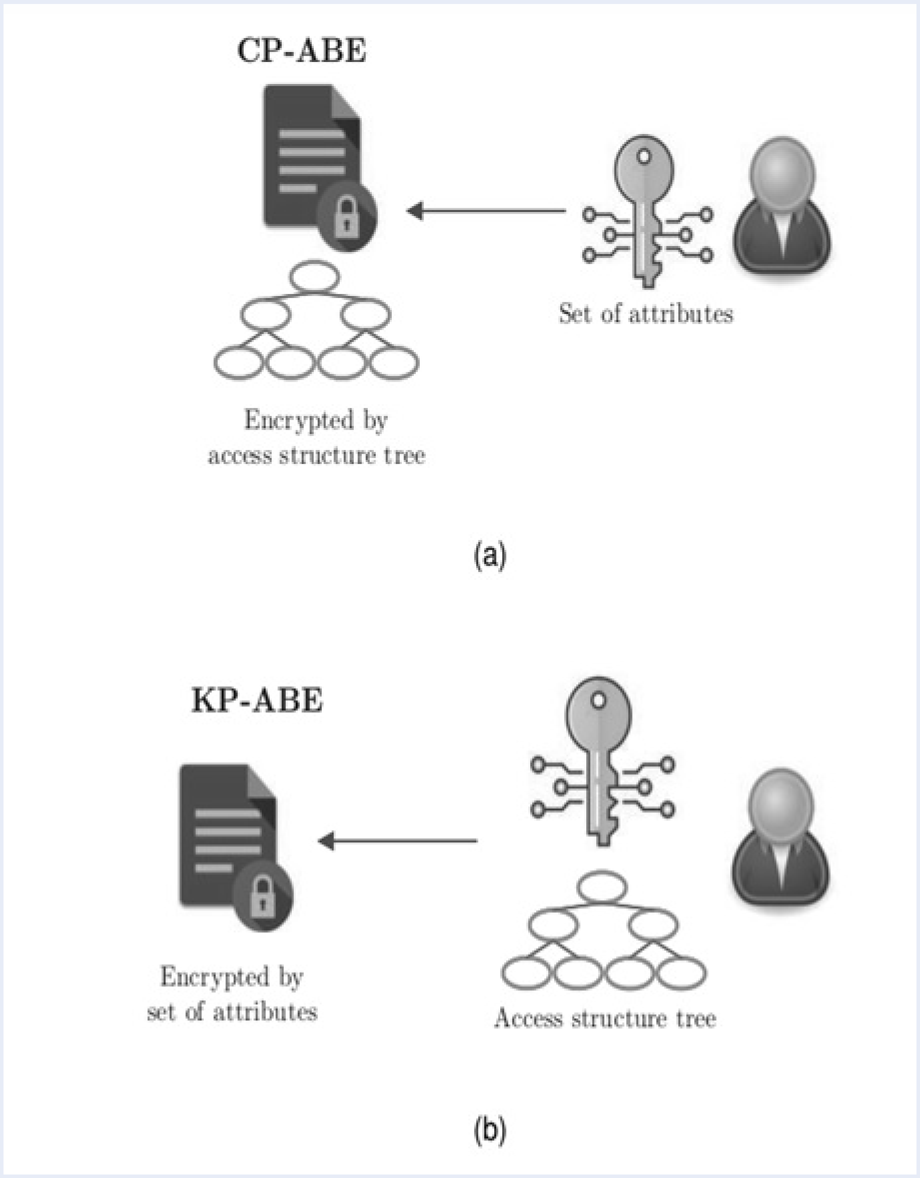
CP-ABE
Set of attributes
Encrypted by access structure tree
(a)
KP-ABE
Encrypted by set of attributes
Access structure tree
(b)

Figure 1: Comparison between CP-ABE and KP-ABE. (a). CP-ABE. (b). KP-ABE

by Bethencourt et al.[7], especially in the policy and the structure of cpabe. DET-ABE was implemented by Miguel Morales-Sandoval, which encrypted data using AES and, in turn, using CP-ABE to protect the AES key. DET-ABE provides only compiled classes as a framework and lacks the functionality and utility to deploy on a server or client-side.

Since its first introduction in 2008, the Android platform has been at the forefront of the mobile revolution and gained enormous positive attention of worldwide users around 85 percent to the global share, as shown in Figure 2. That is why most research activities focus solely on Android, and this paper also does the same using the Android device as a client to communicate with a server for sending encrypted data and decrypting retrieved data for the experiments. Android devices must store the private key and can encrypt/decrypt data locally. We use cpabe-java implemented by Junwei Wang to do the experiments, cpabe-java is open-source software so we can inspect their code and provide improvement if necessary. cpabe-java only needs JPBC library[8] as a primitive dependency to run and can run fine on any machine running Java. cpabe-java was tested on JPBC-

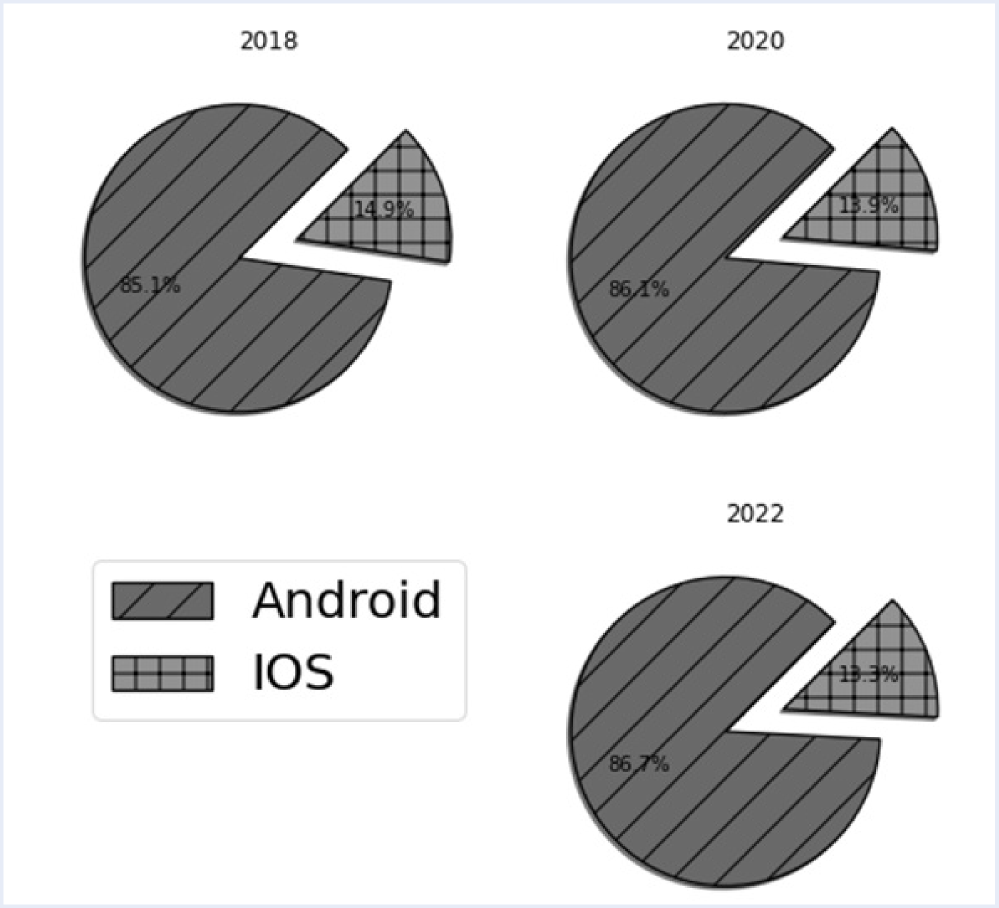

Figure 2: IMPLEMENTATION

1.2.1 while the latest version is JPBC-2.0.0. We tested the availability of the implemented project again on the newer version of the library and modified the code to suit our needs. Since Android uses Java as the native programming language, we bring all the source to Android devices and test its feasibility. We only set up experiments to use four fundamental functions: parameters setup, encryption, key generation, decryption with the flow of processes is described in Figure 3. With A is the access structure tree, M is the message, PK is the public key, MK is the secret master key, SK is the private key, CT is the ciphertext, S is the set of attributes. A and S are input from users.

For simplicity, we will not consider any factors that could affect the network, which results in problems of network traffic or interception from the third party in the experiments. We decide to use cpabe-java to deploy on both server and mobile client, so the communication between the two will be much easier to manage. We are also taking the high popularity of Windows and Unix-based operating systems into consideration; for that, cpabe-java is a much better option compared to AndrABEn and DET-ABE.

ABE is unique in a way that it is associated with three types of keys instead of two like asymmetric encryption. Usually, when generating the unique pair public-private, we can remove any trace of the private key on the server-side without any thoughts and only care about publishing the public key. However, with ABE having associated secret master key in the generation process of a public key, we must store the secret master key securely. We discuss two approaches for setting up the system to generate cryptographic keys.

Our work follows the model shown in Figure 4, which employs a trusted third party called, the client sends a request for private key generation to the server and, in turn, the server redirects the request to Trusted Authority (TA). TA creates a public key, master key, and then uses them to proceed private key generation process. TA must securely store the master key, send back the private key to the server, and the server replies to the client. TA and server must not perform any

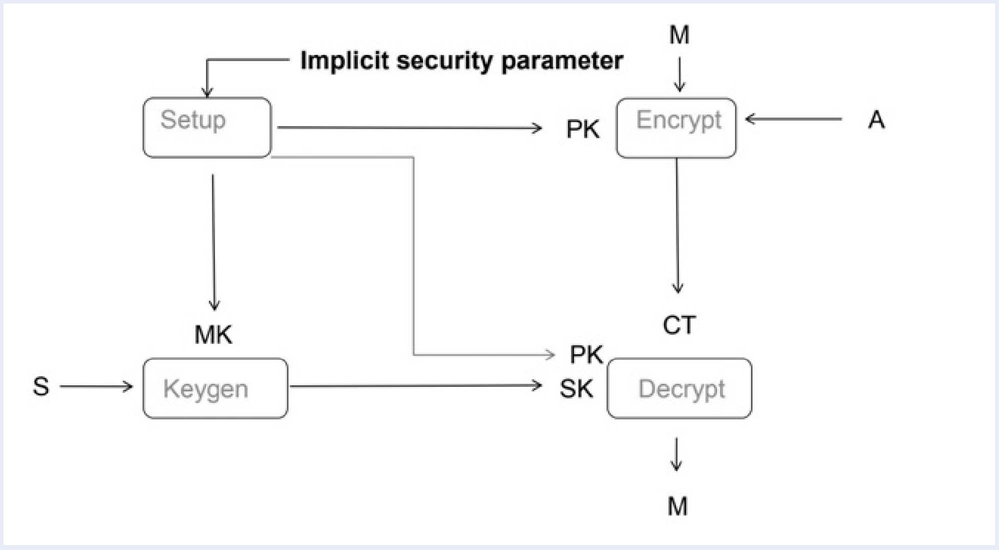

Figure 3: CP-ABE fundamental functions

actions which are related to storing or sending the private key to any other entities besides a client who makes a request.

Unfortunately, involving a third party always invites unforeseen security loopholes. We decided to deploy Kerberos (a network authentication protocol which was purposely designed to provide strong authentication) on the server-side to restrict the access of the storage file even if it is for the administrator account of the system as depicted in Figure 5. With this, we can assume it is safe to store the master key securely on the server-side.

We observe that the storage of the master key is critical; this is probably the reason why it is called the secret master key. Kerberos protocol inspired by the images of mythical Greek beast: three-headed guard dog stands firm acting as a gatekeeper of Hades place to ensure nobody who enters will ever leave. It is fitting since the Kerberos protocol required third-party (Key Distribution Center) to authenticate between client and service or host machine. The third-party mentioned in the Kerberos protocol is entirely different from the one in Figure 4. Kerberos is only responsible for authentication while the other one responsible for storing and generating keys. The process of Kerberos protocol can be summarized as in Figure 6.

In a nutshell, Kerberos consists of the following features:

- A protocol for authentication
- Using tickets as a mean to prove users' identity
- Avoiding storing the password locally or sending them over a network
- Using trusted the third party for the procedure
- Using symmetric cryptography

Kerberos protocol is interesting because we can easily understand it just from the viewpoint of non-technical people. The concept of the ticket is very similar to the ticket we buy in a theme park or amusement park. We must first pay the fee for the entrance ticket. Then we can go in using the ticket to request services the park offered.

We use the Kerberos protocol to enforce the access control policy of users and services effectively. We assume that our server composed of many host machines connected. Administrators can connect to the server for maintenance and perform some tasks. The authentication and authorization processes are critical. Looking at some applications or frameworks like Hadoop default model of authentication, when it is presented with a username, Hadoop believes whatever we say and make sure every machine in the cluster thinks the same. Analogously, when a person is at a party or workshop approaches and introduces himself as 'A', we naturally believe he is truly 'A'. Hadoop default model of authentication pretty much behaves the same. By using the Kerberos protocol, with the same analogy, we would, in response, ask to see his/her card, verify it by checking against the database. In industry, people use Hadoop with Kerberos protocol for authentication.

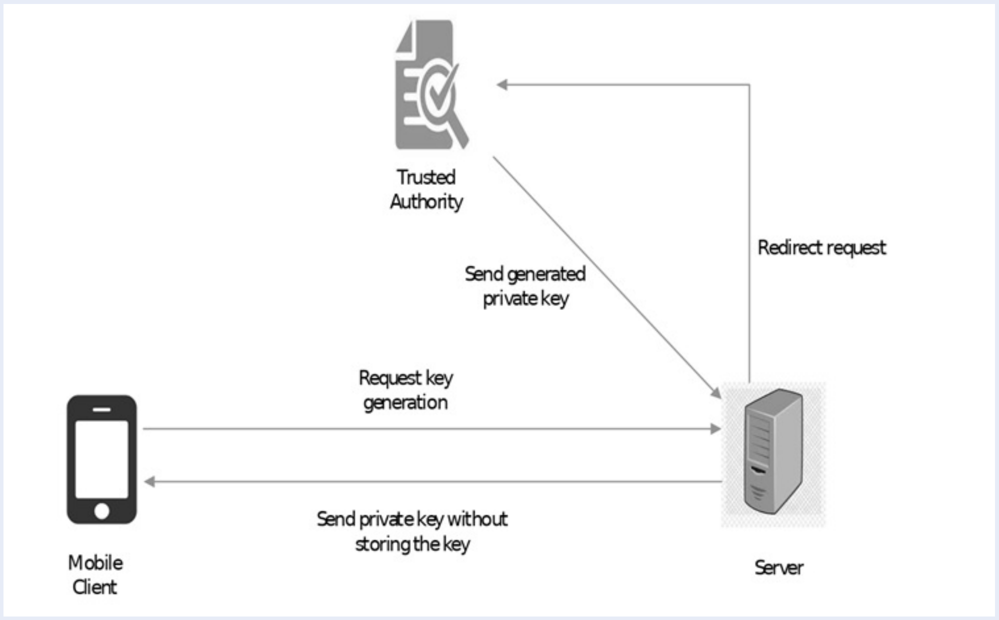

Figure 4: Simple Client-Server with Trusted Authority in Key Generation

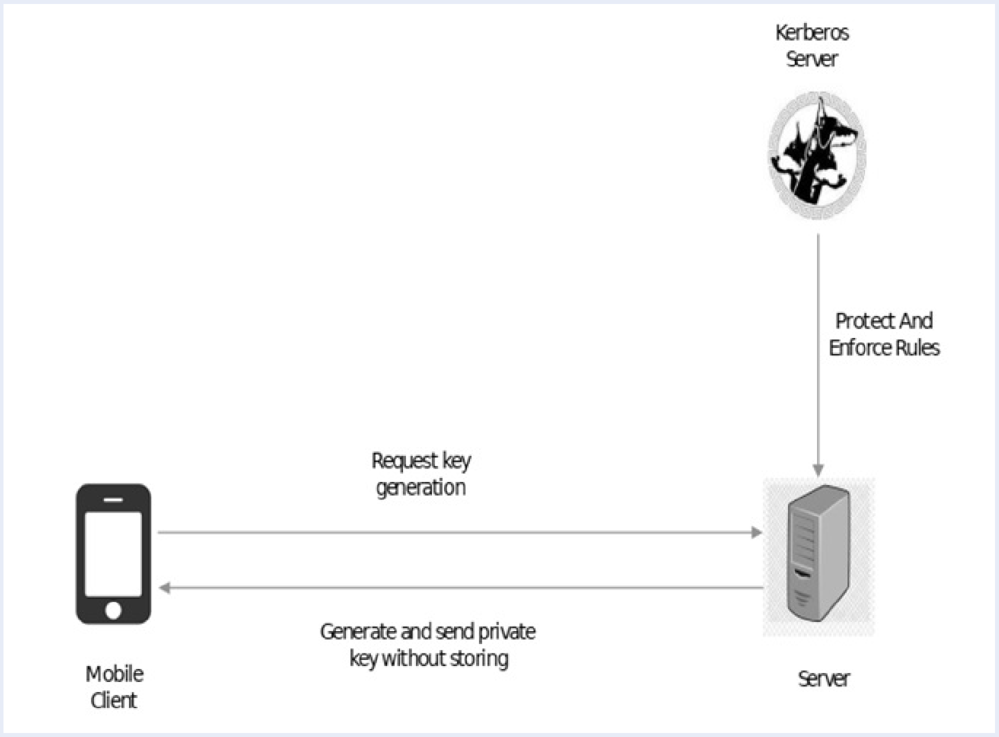

Figure 5: Simple Client-Server with Kerberos Server In Key Generation

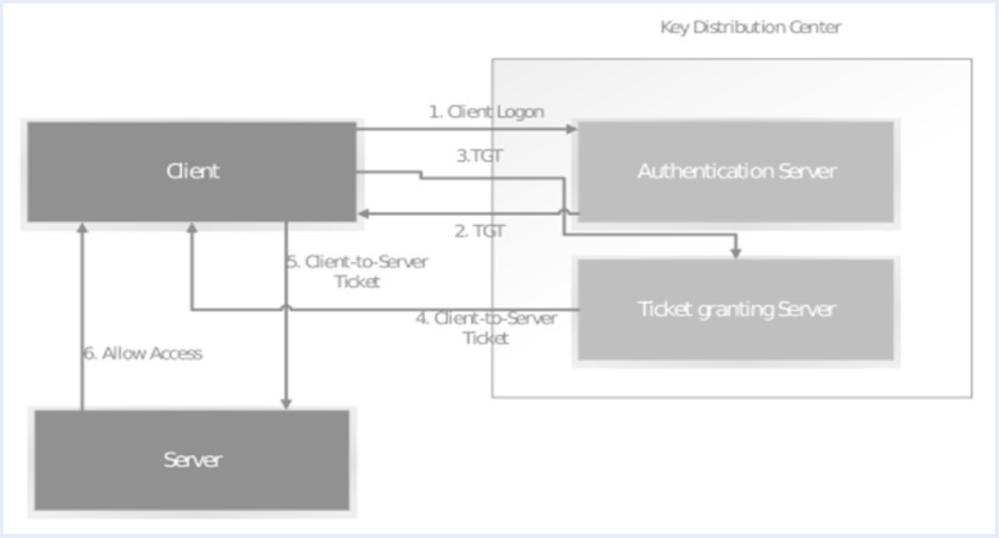

Figure 6: Kerberos Server Process

Kerberos protocol does not improve security strength. However, the Kerberos protocol is advantageous in the authentication process. The system can only authorize users to set rules after successfully authenticate users. With the help of the Kerberos protocol, we can authenticate users and enforce the regulations on specific groups of users.

Typically, Kerberos protocol is used within corporate/internal environments. Rather than typing the credential, again and again, to access the internal payroll site to review the payment and bonus. A ticket (cached on the system) is used for authentication.

We apply the ABE implementation of Android mobile devices to health care applications like[9]. We use a mobile application for users to scan around the vicinity of the current location for the nearest doctors. Users can upload their encrypted health records with a policy that specifies the doctors of specialized fields depending on their particular symptoms. Only doctors with specialized skill attributes can decrypt and read the content. A doctor can decide whether to accept the patient's appointment. The motivation is to allow both doctors and patients to search actively for each other. Health records can be stored on the Cloud.

RESULT AND DISCUSSION

Our proposed method focuses more on the deployment of the Kerberos protocol to enhance the security of the system and securely store the secret key instead of the improvement of the ABE scheme.

The most severe issue with the ABE is central trust. The ABE in our setup requires faith in a central authority – the private key generator. The private key generator can also be installed on users' devices. However, users have to manage their keys and security, and the performance of a mobile device will affect the key generation significantly. Central authority can be heavily invested to become secure and trusted. Generally, the centralized server has more computational power compared to users' devices. This makes it most appropriate in enterprise settings.

Another problem is the performance. ABE is slow because it involves creating a policy tree. ABE is most expensive on decryption, which is the worst place to be slow because decryption is most likely the most performing operation from users. We test the performance of ABE on various file sizes to see what data can be applied in the ABE scheme. The result is shown in Figure 7.

Communication is vital because of the request of key generation to trust authority. During the commission, attackers can sniff and get the private key. It is imperative to encrypt the channel between users and the centralized server. We can use TLS to help secure communication.

Overall, the cpabe-java worked well with the latest version of JPBC and Android devices. It was shown that the more attributes implemented, the longer the execution time is. The strength of the encryption algorithms is compared for the key size because the number of tasks needed to break the algorithms or to establish the key is approximately the same using a given source. The strength of an algorithm can be

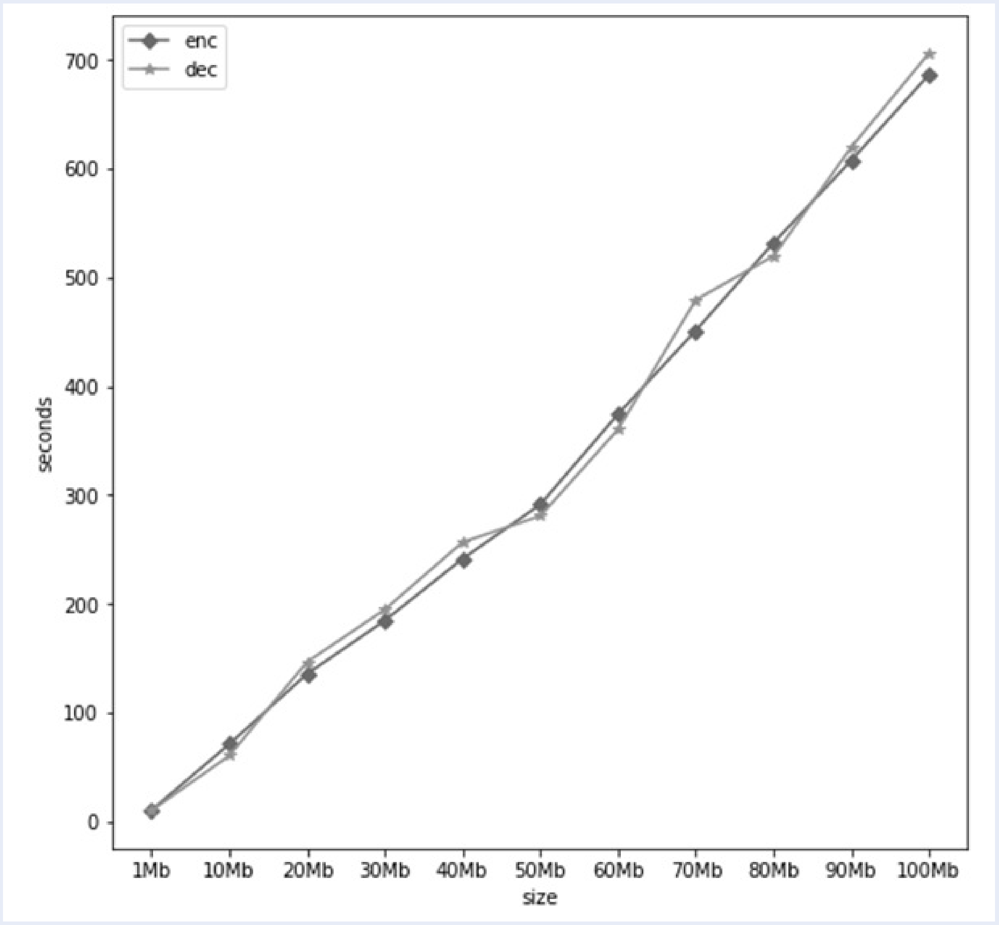

Figure 7: ABE performance

viewed as the number of works needed to try all possible keys for that algorithm. The comparison of the security strength of cpabe using Type A with RSA in Table 1 and Table 2 [10] shows that ABE is better in performance wise of the same security strength level.

Table 1: RSA strength level in bits

Security level in bits	RSA modulus size
80	1024
112	2048
128	3076
192	7680
256	15360

ABE is proven to achieve fine-grained access and management but not the first one. Many types of research about scalable encryption based on symmetric and asymmetric encryption have been done

Table 2: ABE strength level in bits with type a pairing in jpbc

Security level in bits	80	112	128
Bit length of r (q prime)	160	224	256
Bit length of q (field size)	512	1024	1536

before. Both asymmetric and symmetric cryptography perform in one-to-one mapping while ABE works in a one-to-many manner. To achieve one-to-many mapping, they approached a group key management agreement. While this approach works, it also brings shortcomings. Symmetric key cryptography solutions: based on the symmetric key cryptography derivation methods, which can achieve fine-grained data access. This approach can easily be applied to the group without any modifications. Unfortunately, the symmetric key cryptography based solutions have many drawbacks. The most obvious problem is the key distribution due to the nature of symmetric key cryptography, which employs identical keys for both encryption and decryption. Users either have to either manually meet face-to-face for trustworthy secret sharing of the key or require a secure key agreement protocol like[11]. The complicated process leads to high management overhead and time consuming when there is a large number of users. With this kind of solution, we can see that user revocation of privilege level access is not supported since the keys have already been known to all. In case the user revocation is a must, this can be done. However, it is very inefficient; upon the dismissal of one user, all the remaining users are also affected and have to generate a different set of keys, data also need to be re-encrypted.

Public key cryptography based solutions: based on the asymmetric cryptography derivation methods, using for group key, also have many drawbacks. The key distribution is not a big problem now. Unfortunately, asymmetric cryptography requires the keys must be many times longer than key in symmetric cryptography counterpart to boost the equivalent security level, which is more computationally costly. The key management overhead is still potentially high and is vulnerable to a collusion attack. Collusion attack is the execution of operations to combine many parts of the known keys to create a new key capable of decrypting the file. Jikai Teng and Chuanku Wu researched the collusion attack on asymmetric group key[12].

In conclusion, both asymmetric and symmetric have high overhead key management as the complexities of key creation, and user revocation to the number of users is a positive correlation (the higher number of users is, the more complexity in key generation and user revocation). ABE is proven to overcome this adversity. Questions raised ABE is susceptible to collusion attack by multiple users collect sufficient information and combined many private keys to decrypt data. Fortunately, ABE is resistant to this attack, as described in[7]. We observe that in the key generation process, each user is assigned a random parameter value, which is then embedded in the private key. So, using different private keys means different parameter values in the decryption process, thus yields in failure. ABE also has a fair share of doubt about susceptible to insider attack by investigating required attributes then create a new key. Creating a new key from scratch with knowledge of exact attributes also yields in failure as the associated master key is needed to generate a private key. A server or Trusted Authority depended on the paradigm securely stores the master key. Typically, users do not have any means to grasp the master key to generate a new private key.

ABE is best suited in situations where an encrypted file involved multiple parties, for example, nurses, doctors, family members can gain access to a patient's health record but with different privilege levels; or when to broadcast without regards to recipients: military operations, a Facebook personal circle of friends. Many types of research also apply ABE in IoT[13].

ABE should not be used for any applications that require identity ensuring, for example, Blockchain. However, the ABE scheme can represent any individual by using many personal attributes of an individual that are very difficult to forge: fingerprint, retina, face, voice, hardware id, etc. Currently, ABE implementations only support string and numerical data types. Converting these unique attributes requires in-depth knowledge, specialized skill, and complicated process. Besides, determining the number of attributes for a specific application is another problem that needs to be addressed adequately.

CONCLUSION

We have provided an implementation of the ABE scheme for Android mobile devices with the Kerberos protocol and evaluated several security features for secured data sharing and performance of ABE on various file sizes. With the increasing expansion of cloud computing, IoT, mobile devices, this study can be applied for data security and privacy protection. ABE has proved its advantages in many practical applications. ABE can also be applied to mobile devices, but soon becomes more and more popular in mobile computing applications. Libraries and frameworks are implemented to help visualize this scheme. Future work focuses on selecting several appropriate attributes for the ABE scheme.

ACKNOWLEDGEMENT

This research activity is funded by Vietnam National University in Ho Chi Minh City (VNU-HCM) under the grant number C2019-28-06.

ABBREVIATION

ABE	Attribute-Based Encryption
AES	AES Encryption Algorithm
CP-ABE	Ciphertext-Policy ABE
DET-ABE	Digital Envelop Technique ABE
IoT	Internet of Things
KP-ABE	Key-Policy ABE
LSSS	Linear Secret Sharing Scheme
RSA	RSA Encryption Algorithm
TA	Trusted Authority
TLS	Transported Layer Security

CONFLICT OF INTEREST

The authors declare that there is no conflict of interest

AUTHORS' CONTRIBUTIONS

N. T. Dang wrote the manuscript and provided data for Tables I and II. H. D. Le checked the ABE algorithm with the Kerberos protocol. S. T. Le conducted ABE implementation on Android mobile devices. H. M. Tran conducted secured data sharing scenarios for the ABE scheme and provided evaluation analysis. All authors reviewed the final manuscript.

REFERENCES

1. Boneh D, Sahai A, Waters B. Functional encryption: a new vision for public-key cryptography. Commun ACM. 2012;55:56–64. Available from: https://doi.org/10.1145/2366316.2366333.
2. Sahai A, Waters B. Fuzzy identity based encryption. In IACR Cryptology ePrint Archive. 2004;Available from: https://doi.org/10.1007/11426639_27.
3. Goyal VK, Pandey O, Sahai A, Waters B. Attribute-based encryption for fine-grained access control of encrypted data. IACR Cryptology ePrint Archive. 2006;309:2006. Available from: https://doi.org/10.1145/1180405.1180418.
4. Ambrosin M, Conti M, Dargahi T. On the feasibility of attribute- based encryption on smartphone devices. In IoT-Sys@MobiSys. 2015;Available from: https://doi.org/10.1145/2753476.2753482.
5. Morales-Sandoval M, Diaz-Perez A. Det-abe: A java api for data confidentiality and fine-grained access control from attribute based encryption. In Naeem Raja Akram and Sushil Jajodia, editors, Infor- mation Security Theory and Practice: 9th IFIP WG 11.2 International Conference, WISTP 2015, pages 104-119, Heraklion, Crete, Greece, 2015. Springer International Publishing;Available from: https://doi.org/10.1007/978-3-319-24018-3_7.
6. Wang J. Java realization for ciphertext-policy attribute-based encryp- tion. 2012;.
7. Bethencourt J, Sahai A, Waters B. Ciphertext-policy attribute-based encryption. 2007 IEEE Symposium on Security and Privacy (SP '07). 2007;p. 321–334. Available from: https://doi.org/10.1109/SP.2007.11.
8. Caro AD, Iovino V. jpbc: Java pairing based cryptography. In Proceedings of the 16th IEEE Symposium on Computers and Commu- nications, ISCC 2011, pages 850-855, Kerkyra, Corfu, Greece, June 28-July 1, 2011. IEEE;Available from: https://doi.org/10.1109/ISCC.2011.5983948.
9. Balapure SR, Khodke PA. Privacy preservation of e-health care system in cloud. 2017;.
10. Hemalatha SB, Manickachezian R. Security strength of rsa and attribute-based encryption for data security in cloud computing. 2014;.
11. Teng J, Wu C. A collusion attack on asymmetric group key exchange. Security and Communication Networks. 2015;8:2189–2193. Available from: https://doi.org/10.1002/sec.1163.
12. Teng J, Wu C. A collusion attack on asymmetric group key exchange. Security and Communication Networks. 2015;8:2189–2193. Available from: https://doi.org/10.1002/sec.1163.
13. Ambrosin M, Anzanpour A, Conti M, Dargahi T, Moosavi SR, Rahmani A, et al. On the feasibility of attribute-based encryption on internet of things devices. IEEE Micro. 2016;36:25–35. Available from: https://doi.org/10.1109/MM.2016.101.

Tạp chí Phát triển Khoa học và Công nghệ – Kĩ thuật và Công nghệ, 3(S1):SI17-SI27

Open Access Full Text Article

Bài Nghiên cứu

Ứng dụng mã hóa dựa trên thuộc tính cho các thiết bị di động

Đặng Tâm Nhân[1], Lê Hải Dương[2], Lê Thanh Sơn[1], Trần Mạnh Hà[2,*]

TÓM TẮT

Thế kỷ 21 chứng kiến sự phát triển nhanh chóng của các thiết bị di động nhỏ và tiện lợi như điện thoại thông minh, máy tính bảng, máy chơi điện tử, nốt cảm biến, v.v. Sự gia tăng của các thiết bị di động như vậy dẫn đến sự bùng nổviệc truyền tải dữ liệu khổng lồ thông qua Internet và các dịch vụ trực tuyến cùng với những thách thức về an toàn dữ liệu. Người ta luôn suy nghĩ đến một giải pháp để bảo vệ dữ liệu nhạy cảm khỏi người dùng trái phép và giải pháp phổ biến nhất là sử dụng mã hóa. Trong khi nhiều hoạt động nghiên cứu về mã hóa chức năng đã được áp dụng rộng rãi cho các thiết bị mạng, máy tính và ứng dụng, thiết bị di động vẫn thu hút nhiều sự chú ý về các vấn đề bảo mật do một số hạn chế về tài nguyên hệ thống, tín hiệu kết nối, tiêu thụ điện năng mà người dùng trái phép có thể khai thác để thực hiện các tấn công. Đặc biệt, thiết bị di động trở thành công cụ chính để chia sẻ dữ liệu trên Internet thông qua các dịch vụ trực tuyến, chẳng hạn như các dịch vụ lưu trữ và chia sẽ thông tin được cung cấp bởi Facebook, Youtube, Amazon, DropBox, Online Games, v.v. Bài viết này trình bày một nghiên cứu về lược đồ mã hóa dựa trên thuộc tính (ABE) khai thác các thuộc tính người dùng để xây dựng khóa bí mật và dữ liệu mã hóa. Mã hóa ABE được chỉ định bởi một tập hợp các thuộc tính hoặc một chính sách xác định các thuộc tính mà người dùng sở hữu. Bài viết cũng mô tả một vài ứng dụng triển khai ABE được áp dụng trong cộng đồng mật mã và những thách thức của việc tích hợp ABE vào các ứng dụng thực tiễn. Cuối cùng, bài viết đề xuất ứng dụng triển khai ABE cho thiết bị di động Android. Ứng dụng này kết hợp với giao thức Kerberos có thể áp dụng cho nhiều ứng dụng chia sẻ dữ liệu an toàn. Giao thức Kerberos nhắm đến việc cung cấp xác thực qua lại cho mô hình máy chủ và máy người dùng. Các thực nghiệm đánh giá ứng dụng ABE đề xuất trên thiết bị di động Android cùng với hệ thống Kerberos. Phần đánh giá cũng bao gồm hiệu suất của ABE với một số thảo luận và rút ra bài học kinh nghiệm.

Từ khoá: Mật mã học ứng dụng, Mã hóa dựa trên thuộc tính, Chia sẻ dữ liệu an toàn, Thiết bị di động, Điện toán di động

[1]*Trường Đại học Quốc tế, Đại học Quốc gia Thành Phố Hồ Chí Minh, Thành Phố Hồ Chí Minh, Việt Nam*

[2]*Trường Đại học Quốc tế Hồng Bàng, Thành Phố Hồ Chí Minh, Việt Nam*

Liên hệ

Trần Mạnh Hà, Trường Đại học Quốc tế Hồng Bàng, Thành Phố Hồ Chí Minh, Việt Nam

Email: hatm@hiu.vn

Lịch sử

- Ngày nhận: 30-7-2019
- Ngày chấp nhận: 26-8-2019
- Ngày đăng: 17-10-2020

DOI :10.32508/stdjet.v3iSI1.518

Trích dẫn bài báo này: Nhân D T, Dương L H, Sơn L T, Hà T M. **Ứng dụng mã hóa dựa trên thuộc tính cho các thiết bị di động**. *Sci. Tech. Dev. J. - Eng. Tech.;* 3(S1):SI17-SI27.