Fall 2014

User-differentiated hierarchical key management for the bring-your-own-device environments

Di Xie
Purdue University

Graduate School Form 30
(Revised 08/14)

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Di Xie

Entitled
USER-DIFFERENTIATED HIERARCHICAL KEY MANAGEMENT FOR THE BRING-YOUR-OWN-DEVICE ENVIRONMENTS

For the degree of Master of Science

Is approved by the final examining committee:

Baijian Yang

John A. Springer

Samuel P. Liles

To the best of my knowledge and as understood by the student in the Thesis/Dissertation Agreement, Publication Delay, and Certification/Disclaimer (Graduate School Form 32), this thesis/dissertation adheres to the provisions of Purdue University's "Policy on Integrity in Research" and the use of copyrighted material.

Baijian Yang

Approved by Major Professor(s): _____

_____

Approved by: Eugene H. Spafford 09/22/2014

Head of the Department Graduate Program Date

USER-DIFFERENTIATED HIERARCHICAL KEY MANAGEMENT FOR THE BRING-YOUR-OWN-DEVICE ENVIRONMENTS

A Thesis
Submitted to the Faculty
of
Purdue University
by
Di Xie

In Partial Fulfillment of the
Requirements for the Degree
of
Master of Science

December 2014
Purdue University
West Lafayette, Indiana

.

For my parents and those who love me

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor Dr. Baijian “Justin” Yang for his continued support. This thesis would never be finished without him.

I would like to thank the rest of my thesis committee members: Dr. Samuel P. Liles and Dr. John A. Springer, for their excellent guidance, patience, and tough questions that make this thesis better.

I would like to thank Dr. Giuseppe Ateniese, who amiably permitted me to do the research on the basis of his work.

I would like to thank Marlene Walls and Stacy Lane, who provide an extensive set of help on the progress of finishing the thesis.

Last but not the least, I would like to thank the help from Deborah Schwarte, who has been helpful during the proofreading and editing of my thesis. Besides her great work, I also would like to thank for her encouragement all the time.

TABLE OF CONTENTS

Page

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

LIST OF ABBREVIATIONS

IT	Information Technology
BYOD	Bring-Your-Own-Device
NAC	Network Access Control
MDM	Mobile Device Management
UDTLEBC	User-Differentiated Two-Layer Encryption-Based Construction
TLEBC	Two-Layer Encryption-Based Construction
AES	Advanced Encryption Standard
poset	Partially ordered Set
IND-ST	Key Indistinguishability with respect to Static Adversaries
IND-AD	Key Indistinguishability with respect to Adaptive Adversaries
REC-ST	Key Recovery with respect to Static Adversaries
REC-AD	Key Recovery with respect to Adaptive Adversaries
IND-P1-C0	Plaintext Indistinguishability against an adversary
CBC	Cipher-Block Chaining
CFB	Cipher Feedback
OFB	Output Feedback
CTR	Counter
ECB	Electronic Codebook

GLOSSARY

Advanced Encryption Standard - "... a FIPS-approved cryptographic algorithm that can be used to protect electronic data. The AES algorithm is a symmetric block cipher that can encrypt (encipher) and decrypt (decipher) information. Encryption converts data to an unintelligible form called ciphertext; decrypting the ciphertext converts the data back into its original form, called plaintext. The AES algorithm is capable of using cryptographic keys of 128, 192, and 256 bits to encrypt and decrypt data in blocks of 128 bits." [1]

Bring-Your-Own-Device - "the new mantra of employees who are empowered to innovate the way they work, using the technology tools they prefer."[2]

Hierarchical Key Management Scheme - a technique that provides access controls in multi-user systems by cryptographic keys.[3]

Mobile Device Management - "emerged recently in response to the ubiquitous adoption of handheld mobile devices in corporations. MDM focuses on the end device rather than the network, using client software on the devices. By communicating with the client software, the MDM server enables IT to manage these devices."[4]

Mode of Encryption - "In cryptography, a mode of operation is an algorithm that uses a block cipher to provide an information service such as confidentiality or authenticity. A block cipher by itself is only suitable for the secure cryptographic transformation (encryption or decryption) of one fixed-length group of bits called ablock. A mode of operation describes how to repeatedly apply a cipher's single-block operation to securely transform amounts of data larger than a block." [5]

Network Access Control - "technology enables IT to define and control how devices and users gain access to network resources."[4]

Partially Ordered Set - "In mathematics, especially order theory, a partially ordered set (or poset) formalizes and generalizes the intuitive concept of an ordering, sequencing, or arrangement of the elements of a set." [6]

Security Class - Disjoint groups that users are classified into and "employee the relation of partial ordering, that is, a security class at higher level can derive from his own cryptographic key the keys of other security class below him".[7]

Symmetric encryption scheme - “a class of algorithms for cryptography that use the same cryptographic keys for both encryption of plaintext and decryption of ciphertext.”[8]

Time-Bound Hierarchical Key Management Scheme - Hierarchical Key Management Scheme in which the cryptographic key of a class are different for each time period”.[9]

ABSTRACT

Xie, Di. M.S., Purdue University, December 2014. User-Differentiated Hierarchical Key Management for the Bring-Your-Own-Device Environments. Major Professor: Baijian "Justin" Yang.

To ensure confidentiality, the sensitive electronic data held within a corporation is always carefully encrypted and stored in a manner so that it is inaccessible to those parties who are not involved. During this process, the specific manners of how to keep, distribute, use, and update keys which are used to encrypt the sensitive data become an important thing to be considered. Through use of hierarchical key management, a technique that provides access controls in multi-user systems where a portion of sensitive resources shall only be made available to authorized users or security ordinances, required information is distributed on a need-to-know basis. As a result of this hierarchical key management, time-bound hierarchical key management further adds time controls to the information access process. There is no existing hierarchical key management scheme or time-bound hierarchical key management scheme which is able to differentiate users with the same authority. When changes are required for any user, all other users who have the same access authorities will be similarly affected, and this deficiency then further deteriorates due to a recent trend which has been called Bring-Your-Own-Device. This thesis proposes the construction of a new time-bound hierarchical key management scheme called the User-Differentiated Two-Layer Encryption-Based Scheme (UDTLEBC), one

which is designed to differentiate between users. With this differentiation, whenever any changes are required for one user during the processes of key management, no additional users will be affected during these changes and these changes can be done without interactions with the users. This new scheme is both proven to be secure as a time-bound hierarchical key management scheme and efficient for use in a BYOD environment.

CHAPTER 1. INTRODUCTION

1.1 Introduction

To ensure confidentiality, the sensitive electronic data that is maintained for corporations is always carefully encrypted and kept out of bounds to parties who are not involved. During this process, the specific steps of how to keep, distribute, use, and update keys, which are all used to encrypt the sensitive data, become an important thing to be considered. This total process is one of key management.

While users in corporations are naturally, or not artificially, organized into a hierarchy, the scheme which is used for key management by these users is called the hierarchical key management scheme [3]. The hierarchical key management scheme can also be viewed as a technique that provides access controls in multi-user systems 0 where a portion of sensitive resources is only available to authorized users or where security ordinances require that “information is distributed on a need-to-know basis” [11]. To fulfill such need-to-know requirements, users are categorized into different disjoint groups. Each specific disjoint group represents a special type of users based on their access authorities. These disjoint groups are called security classes [7]. The data in maintained for the government is always categorized into four classes: “unclassified”, “confidential”, “secret”, and “top-secret”. Following categorization, only limited people are assigned the authority to access data which is categorized as either “secret” or “top-

secret" [9]. In another words, the senior officers have greater access authorities than do those new staff members, who typically can only access the "unclassified" data.

The deficiency of the hierarchical key management scheme is that researchers fail to consider those cases where users are provided with access for only a short time period. Such constraints are common in real world practices. A company might frequently provide a number of short-time jobs or internships to college students. When absorbing these college students into the corporate structure, the company should ensure that these college students do and can only have access to sensitive data during their specific time of contract.

This demand for time constraints has given birth to the time-bound hierarchical key management scheme. With the time-bound hierarchical key management scheme, access is restricted both by the security class of the data and the purported "valid" time assigned to said data. The concept of the time-bound hierarchical key management scheme was first proposed by Tzeng in 2002 [9] and further developed thereafter [12][13][14][15][16].

1.2 Problem Statement

No existing hierarchical key management scheme or time-bound hierarchical key management scheme can be employed to effectively differentiate between users who are at the same level of authority. If varied users all have an identical access authorization, they are then treated equally by their ability to share matching secret keys and private data. A more detailed introduction of this topic shall be introduced in sections 2.1 and 2.2. This level of similar or equal treatment will often cause problems when an organization is handling personnel changes. A senior manager might leave a company in pursuit of other

business interests. The resignation may be quite unexpected, and the former employee typically maintains privileges to access the sensitive data because of the fact that the secret key and private data information which has been assigned to her/him have not yet expired. In response to the resignation, the information technology (IT) department of the company, which is the central authority of access control, is required to immediately withdraw the senior manager's accessibility in order to avoid a potential data leakage by changing the keys. Since there is no differentiation among users, everyone who has been assigned the same access authority as the person leaving the organization will be similarly rejected to access at this time. The IT department is required to update and reassign new secret keys and data to restore these users' access authorities.

With an efficient time-bound hierarchical key management scheme, this process of regeneration and reassignment does not lead to great inconvenience because personal changes are not part of the daily routine. This problem has been found to be seriously deteriorating as a result of a recently growing trend called Bring-Your-Own-Device (BYOD).

Corporate organizations adopt BYOD to encourage and allow their employees who are using personal devices to have access to privileged company data, information, applications, and systems [2]. Currently, the security of BYOD is ensured by two major techniques, named the (1) Network Access Control (NAC) and (2) Mobile Device Management (MDM) [4]. NAC is a technique used to secure the corporate network through user and device identification [4]. Mobile Device Management (MDM) is an emerging technique that aims at securing the mobile device by means of a secure container system and enforced security policies [4]. A secure container is an isolated,

partitioned, and secure environment applied to a device which may be used to run corporate applications and to store related sensitive corporate data. Through the usage of such containers the company works to build a more enhanced and secure corporate infrastructure for specific devices and better isolate the corporate construct from interference from personal usage [17].

A sketch is shown in Figure 1.1. The security of BYOD can be divided into three layers [4]. The first layer is the security of mobile device network access control. This layer is used to provide for safe networking based on the access policies, which will categorize the users into specific groups, such as guest, limited, and full. Limited users will have greater access authorities than do guest users, and full users shall have access to the entire body of the system. The second layer is that of the security held by mobile device management, the construct which provides the integrated mobile device configuration and operation policies. The third layer is that of the security held by the mobile device app stores. These app stores shall aim to provide safe and certified apps in order to ensure that users avoid downloading malware or trojans. These three layers work together to provide strong protection for the personal devices.

With recent developments in both hardware and software, these two elements are becoming increasingly capable of protecting devices from potential attacks. The measured success has been proven by passing the tests of the U.S. federal government. The federal government has opened the department’s door to the next generation of BlackBerry and Android devices on May 2 [18] and Apple’s iOS 6 devices on May 17, 2014 [19].

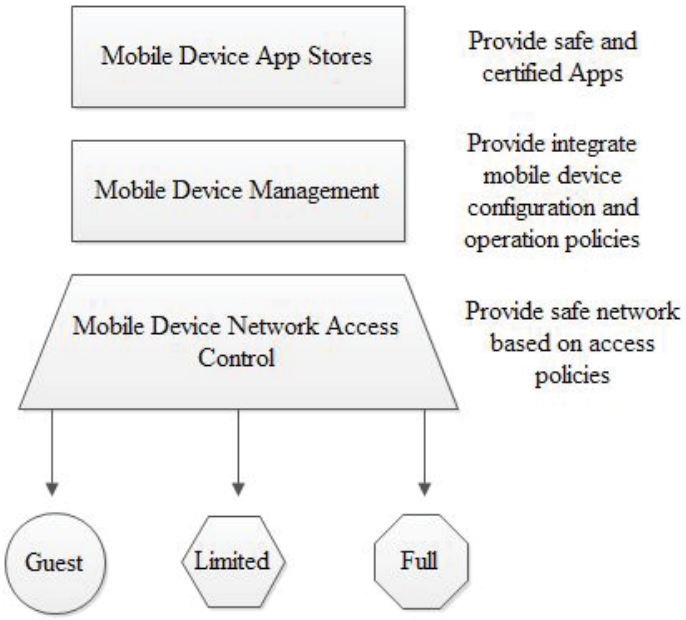

Figure 1.1 BYOD Security Structure

The core concept of BYOD produces a fact that companies no longer have the proprietorship of a number of devices connected into their intranet. This change results in significant influences to the application of the hierarchical key management scheme.

One consequence is that key updates and reassignments will be more frequent. Personal devices are carried everywhere by their owners, so these devices have an increased chance to be lost or stolen than do those devices which are managed by companies under strong security policies. A recent study has discovered that “more than 7,000 devices were lost in seven airports over a twelve month period” and a growing number of thefts occurred between 2007 to 2011 [20]. When a particular device is lost, the older key which was stored on that device should then be abandoned and a relevant new key must be updated and reassigned. We find that problems arise which are similar to those when an employee should choose to leave the company; it is not only one key

that must be reassigned, but the keys for all users who have access to the same body of information. This is an unfortunate result of the current BYOD security systems.

Another consequence is that the actual processes of key updates and reassignments have become formidable in BYOD environment. Since companies no longer have the proprietorship of some devices, companies cannot approach updates and reassignments without seeking the direct permission of their employees. These companies have to define an acceptance baseline of what security and supportability features a BYOD device should support with their employees. The access control is now an awkward combination of the authority an employee has to access corporate data and the authority that company has to manipulate said employee's device. New contracts between companies and employees have to be carefully designed in order to avoid any potential infringement related to privacy and personal property [21]. Extra costs may arise in the design, management, and disposal of these contracts.

Faced with the frequency and formidability of these consequences, this author desires that a new time-bound hierarchical key management scheme with a capability of user differentiation could be built, one which permits that when some changes for a user are required, no additional users shall be affected. It is the research goal that all of these changes can be implemented without any direct interactions with users or their devices, thereby permitting less efforts and costs to avoid any potential privacy infringements.

1.3 Research Question

The basic research question of this thesis is: Is it possible to build a truly provable secure and efficient time-bound hierarchical key management scheme that is capable of differentiating users in the BYOD environment?

1.4 Significance

This thesis proposes a new time-bound hierarchical key management scheme known as the User-Differentiated Two-Layer Encryption-Based Construction (UDTLEBC) by modifying an existing time-bound hierarchical key management scheme, the Two-Layer Encryption-Based Construction (TLEBC). UDTLEBC is constructed with three objectives.

First, the new time-bound hierarchical key management scheme has the capability of user differentiation. When rejecting or accepting one user's access to a security class in a specific time period, other users in the same security class during the same time period will not be affected. On this basis, all changes can be implemented without interactions with users, so the problems of privacy and cost can be precluded.

Second, the new time-bound hierarchical key management scheme is assumed provably secure, i.e., the adoption of this scheme shall not sway its original capacity for security.

At last, he new time-bound hierarchical key management scheme is efficient. That is to say, the performance of the scheme will be evaluated in order to show that the new capability does not generate any significant decline in performance.

1.5 Delimitations

This research focuses only on building a new hierarchical key management scheme with a capability of user differentiation and an ability to establish its security and performance. The new scheme is considered relevant to key distribution and associated access control, so this scheme shall only be used to ensure the confidentiality of data. The BYOD environment will not be discussed in further detail in this thesis, because the process of creating authentic real practices within a BYOD environment actually involve even more complex problems, in which the privacy, efficiency, and cost concerns are merely a few elements of the whole [21]. BYOD will be treated as a truly broad background, and be dealt with in terms of only a few basic concepts in the realm of this study. The security and performance of this scheme will be evaluated from a hierarchical key management scheme perspective rather than by discussing the safety and performance within a real BYOD case in which the environment is complex and where the scheme used may be uncertain. For convenience, the symmetric encryption scheme AES is to be used in all of the cases where a symmetric scheme is required, so that the performance shall necessarily be affected if a different scheme should be used.

1.6 Limitations

This thesis works on building a time-bound hierarchical key management scheme for access controls within a BYOD environment. Only confidentiality and accessibility are considered in this thesis. Security concerns, such as integrity and availability, shall not be considered. The performance of the device is tested by using one emulator and one device, so that the actual performance obtained may be limited by the capability of the

emulator and the device. Detailed information of the emulator and the device shall be provided in this paper to permit others to predict the possibilities of achieving different (better or worse) results through the use of different equipment.

CHAPTER 2. LITERATURE REVIEW

This chapter first provides detailed definitions for both the hierarchical key management scheme and the time-bound hierarchical key management scheme. Then, a historical retrospect is approached to review the evolution of a hierarchical key management scheme and the emergence of the time-bound hierarchical key management scheme. Based on the evolutionary history, this thesis then illustrates that no existing schemes provides the capability of user differentiation, which is believed to be required within a BYOD environment. Further analyzing the capabilities and performance required in a BYOD environment, this thesis explains why TLEBC is believed to be more compatible to the BYOD environment. At last, the TLEBC is briefly introduced.

2.1 Hierarchical Key Management Scheme

This section briefly explains the concept of a hierarchical key management scheme, which is summarized by the author (Relevant works were introduced in section 2.3). In a multi-user system, assume that a set of users are divided into a set of disjoint classes which are called security classes. If we assume that users in one security class have greater access authority than do users in another security class, then a binary relation symbol $\leq$ is used to denote the relationship between these two classes. An example is shown below to introduce such binary relations.

Figure 2.1 A Binary Relation between Two Security Classes

As shown in Figure 2.1, C_1 and C_2 are two security classes and C_1 has more access authority than does C_2. A relationship, $C_2 \leq C_1$, represents a binary relationship of how users in class C_1 can access data in class C_2, but not vice versa. Obviously, these users can access data within their own security class, so the relationships $C_1 \leq C_1$ and $C_2 \leq C_2$ are valid and such a binary relationship is valid for any security class C_i.

If we put security classes in one set, then a hierarchy can be built according to the concept of a partially ordered set (poset). Within the order theory, a partially ordered set formalizes elements of a set by binary relationships which present the ordering of elements. A binary relation indicates that one element can denote another element; for example, security class C_1 can indicate security class C_2 within the previous example. The term "partially" is used to denote that not every pair of elements within the set has a binary relationship.

Such a poset hierarchy can be shown by a graph G^*, in which security classes and binary relations are shown. Each path in the graph G^* presents a binary relation between a pair of security classes and a reflexive of a security class. The graph G^* can be reduced to a graph G, where G denotes the minimal presentation of the graph G^*. If we use V to denote a set of security classes and E^* or E to denote a set of binary relations, Figure 2.2 provides a direct view of G^* and G by using an example of four disjoint security

classes C_1, C_2, C_3, C_4 with the following binary relationships: $C_1 \leq C_1$, $C_2 \leq C_1$, $C_2 \leq C_2$, $C_3 \leq C_1$, $C_3 \leq C_3$, $C_4 \leq C_1$, $C_4 \leq C_2$, and $C_4 \leq C_4$.

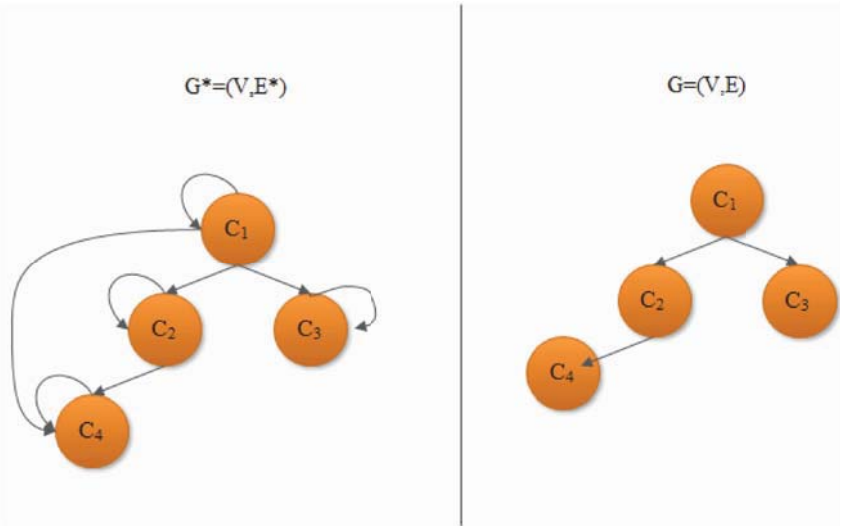

Figure 2.2 Partially Ordered Hierarchy Graphs G* and G

Under the poset hierarchy, users in one security class are not allowed to access data that belongs to another security class unless they already have the direct authority, i.e., a user in security class C_i can access data in security class C_j if and only if there is a binary relationship defined as $C_j \leq C_i$. To satisfy this requirement, data in different security classes are encrypted by different secret keys in order to protect them. The secret key of each security class is only assigned to those users belonging to that security class and to users who are in other security classes but also have access to the data. By using the same example presented in Figure 2.2, if k_i is used to denote the secret key of security class C_i, then users are assigned secret keys by the following table:

Table 2.1 Key Assignments for Users in Different Security Classes

Users	Key Assignments
Users in security class C_1	k_1, k_2, k_3, k_4
Users in security class C_2	k_2, k_4
Users in security class C_3	k_3
Users in security class C_4	k_4

The “higher” a security class within the hierarchy, the more keys which the users in that security class are required to maintain. If users in the security class are “high” in the hierarchy, they have will then have to maintain more keys for accessing data protected the varied security classes which are “low” in the hierarchy. Since the number of keys will expand as the number of security classes and binary relationships increases, storage problems may emerge with hierarchical escalation because those users within the “high” levels of security will need to maintain a large number of keys.

Accordingly, a hierarchical key management scheme is a cryptographic technique designed to build relationships between keys. Such a process is called a key derivation. The term “key derivation” means that a secret key can be used to compute other secret keys. More precisely, a secret key of a particular security class can be used to compute another secret key of another security class if, and only if, there is a binary relationship within the poset hierarchy. This process of key derivation cannot be reversed. The advantage of this technique is that users only have to store one secret key regardless whether they are in a security class that is deemed “high” or “low” in the hierarchy and

regardless of the size of the hierarchy. When they need to access data in other security classes, they can use the secret key assigned to their own security class with the purpose to compute to secret keys which they require. By using the example as is found in Figure 2.2, the following table illustrates these result after using a hierarchical key management scheme:

Table 2.2 Key Assignments and Derivations after Using Hierarchical Key Management Scheme

Users	Key Assignments	Key Derivations
Users in security Class C_1	k_1	$k_1 \rightarrow k_2$ $k_1 \rightarrow k_3$ $k_1 \rightarrow k_4$
Users in security Class C_2	k_2	$k_2 \rightarrow k_4$
Users in security Class C_3	k_3	None
Users in security Class C_4	k_4	None

From Table 2.2, it could be seen that users merely have to store one secret key which belongs to the whole of their security classes, so that the total of the required storage is strongly reduced by using the hierarchical key management scheme.

To enable such a key derivation, secret keys usually have to be generated by a specific generation algorithm, so that the secret keys can be generated with some existent and predetermined binary relationships amongst them. Different hierarchical key management schemes have different generation algorithms which are designed to realize the key derivation intended. The generation algorithm also generates both the private and

public data, which are then used to help with the key derivation. Private data may be sent secretly to the individual users because of the fact that the leakage of such private data might contribute to varied attacks. On the other hand, public data is typically made available to the public because an adversary cannot obtain any particular advantages through gaining this knowledge. With this information in hand, Table 2.2 can then be extended as follows:

Table 2.3 Key Generation, Assignments, and Key Derivation through the Use of a Hierarchical Key Management Scheme

Users	Key Generation	Key Assignments	Key derivation
Users in security Class C_1	k_1, k_2, k_3, k_4 $priv_1, priv_2$ pub	$k_1, priv_1$	$k_1, priv_1, pub \rightarrow k_2$ $k_1, priv_1, pub \rightarrow k_3$ $k_1, priv_1, pub \rightarrow k_4$
Users in security Class C_2		$k_2, priv_2$	$k_2, priv_2, pub \rightarrow k_4$
Users in security Class C_3		k_3	None
Users in security Class C_4		k_4	None

In such a case, then $\lambda = \{t_4, t_5, t_6\}$. The private data $s_{2,\lambda}$ is assigned to the user in order to ensure that the user can only generate the decryption key $k_{j,t}$, whereby j denotes a security class C_j such that $C_j \leq C_2$ and t denotes the time periods $t_4, t_5,$ and t_6. Thus, as compared to a hierarchical key management scheme, the time-bound

hierarchical key management scheme further divides users in each security class into more groups corresponding to different combinations of short time periods.

2.2 Time-Bound Hierarchical Key Management Scheme

This section briefly explains the concept of a time-bound hierarchical key management scheme, which is summarized by the author (Relevant works were introduced in section 2.3). A time-bound hierarchical key management scheme is a special application of the hierarchical key management scheme. Within a hierarchical key management scheme, time is not considered a factor and once a user is categorized into a security class, the user would belong to that security class forever. To delete a user from a security class, the central authority needs to change the secret key of the security class and to reassign a new secret key to all of the users affected. But within a time-bound hierarchal key management schemes, a user will merely belong to a security class for a specified time period. When that time period expires, the user will be eliminated from the security class and shall lose the access authority which he/she once had. This process can be done automatically without regeneration, reassignment or any other participation from the central authority.

To effectuate such a capability, a central authority will first decide the length of the time period, i.e. the long time period. This time period usually extends over a period of key updates. In some real world cases, such a time period could be a season, half a year, or even a year. This long time period will be further divided into a sequence of short time periods. The length of the short time period is also decided by the central authority. An illustrative diagram is shown in Figure 2.3.

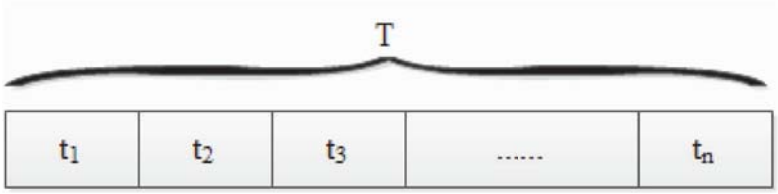

Figure 2.3 Time Sequences

The capital T is used to denote the long time period, and $t_i, i = 1,2,3,...,n$ are used to denote the short time periods within the long time period *T*. Assume that the long time period is the month of January and short time period is merely one day. There are 31 days in January, so there will be 31 short time periods within the long time period. t_1 denotes the first day of the January, t_2 denotes the second day of January, ..., and t_{31} denotes the last day of the January.

On this basis, each security class no longer uses only one key to encrypt data but a set of keys corresponding to the short time periods. If each key is denoted by $k_{i,t}$ where i denotes the security class the key assigned to and t denotes the short time period, key $k_{1,1}$ is used by security class C_1 to protect its data in short time period t_1, key $k_{1,2}$ is used by security class C_1 to protect its data in time-period t_2, and so on. A time-bound hierarchical key management scheme would generate a set of keys for each security class for each short time period. Each security class will be assigned *n* keys, where *n* is the number of short time periods within a long time period. Each user will be assigned a specific piece of private data according to that security class in which the user exists and the short time period the user is allowed to access the data. We denote such specific private data by $s_{v,\lambda}$, where v stands for the security class the user is in and λ stands for

the combination of short time periods in which the user is allowed to access the data. By using the same poset hierarchy shown in Figure 2.2 and same time period shown in Figure 2.3, assume that a user in security class C_2 is allowed to access the data on days 4, 5, and 6; then process of key assignment and derivation are shown in the following table:

Table 2.4 Key Assignment and Derivation of Time-bound Hierarchical Key Management Scheme for a Specific User

User Definition	Key Assignment	Key Derivation
A user belongs to security class C_2 and has access authority on day 4 and 5	$k_2, s_{2,\lambda}$, where λ stands for the combination of $t_4, t_5,$ and t_6	$k_2, s_{2,\lambda}, pub \rightarrow k_{2,4}, k_{2,5}, k_{2,6}, k_{4,4}, k_{4,5}, k_{4,6}$

In such a case, then $\lambda = \{t_4, t_5, t_6\}$. The private data $s_{2,\lambda}$ is assigned to the user in order to ensure that the user can only generate the decryption key $k_{j,t}$, whereby j denotes a security class C_j such that $C_j \leq C_2$ and t denotes the time periods $t_4, t_5,$ and t_6. Thus, as compared to a hierarchical key management scheme, the time-bound hierarchical key management scheme further divides users in each security class into more groups corresponding to different combinations of short time periods.

2.3 Evolutionary History of Hierarchical Key Management Schemes and Time-Bound Hierarchical Key Management Schemes

This section briefly reviews the contributions of other researchers. The first poset hierarchical key management scheme was proposed in 1983 by Aki and Taylor [3]. In

their scheme, users are assigned both a prime and a secret key. Some public data is then made available to the users in order to allow them to implement the key derivation. The security of the scheme is based on a discrete logarithm problem, which is generally believed to be difficult. As the first hierarchical key management scheme, it has been found to have several deficiencies. One major problem is that this scheme requires a significant amount of storage space. In 1985, Mackinnon et al. [22] proposed an improved algorithm that could be used with the Aki-Taylor scheme [3]. This algorithm reduces the amount of public data required by the original scheme, and the required amount of public data was further reduced in 1990 by Harn and Lin [23]. In the original Akl-Taylor scheme [3], the keys are generated in a single linear fashion, proceeding from top to bottom. Harn and Lin [23] then changed this key generation process, so that it was reversed from top-down to bottom-up, a method which requires less storage of the public data.

Besides working to reduce the required amount of storage, many researchers have also tried to make these schemes more “flexible”. This schematic flexibility can be comprehended in two ways. First, the scheme is seen to be suitable for all kinds of situations. Chick and Tavares [24] proposed a special usage of the scheme to enable access control of a set of services, where these services are not necessarily organized within a poset hierarchy. Ohta et al. [25] proposed a scheme especially designed for membership authentication. In their scheme, users are allowed to prove their membership without revealing any additional information (for example, the user’s actual name). Lin et al. [26] proposed a scheme that could be utilized not only in a poset hierarchy but also within more complicated circumstances. A second type of improved flexibility involves

the fact that the scheme, itself, is more flexible to change. Researchers have repeatedly tried to make these changes convenient and fast by reducing the impact of said changes. In 1993, Chang and Buehrer [10] improved the scheme proposed by Mackinnon et al. [22] by importing a one-way trap door function into the scheme. It was discovered that the step of adding a new security class into the poset hierarchy would not necessarily affect most of the previously existing security classes. Hwang and Yang proposed another scheme [27] in 2003, a scheme which provided two contributions. First, it was found that fewer keys are required in their proposed scheme as compared to those existent in previous schemes. Second, they discovered that the process of adding or deleting a security class would affect only a minimal amount of security classes found within the hierarchy.

Other researchers are also providing new schemes which are not based on the research branches that were derived from the Akl-Taylor scheme [3]. Sandu [28] had proposed a hierarchical key management scheme in 1988, using a totally different algorithm. This scheme generates keys through use of a one-way function and the assignment of IDs to each security class. The schematic security depends on the fact that the process of a one-way function cannot be reversed. A large problem with this system remains in the fact that the key derivation of the scheme cannot be approached directly. By using the scheme, a secret key assigned to a security class can only be used to derive secret keys of security classes which are immediate predecessors. To compute a secret key of a security class which is lower in the hierarchy, the intermediate secret keys in the chain must also be generated, thereby leading to many unnecessary computations.

The same problem is also evident in Chang et al.'s scheme [29], proposed in 1992, and Liaw et al.'s scheme [7], proposed in 1993. Both of these research parties relied upon Newton's interpolation method and one-way functions to construct their schemes. Key derivations of their schemes are also indirect. This deficiency was overcome by Hwang [30] in 1999. Hwang subtly used the property of a discrete logarithm and proposed a scheme with similar steps while computing the secret keys directly. Ray et al. [11] provided another solution to this problem which was also found to solve this deficiency.

Zhong [31] proposed a scheme in 2002 through use of the Hash functions. This scheme typically requires less storage than do most of previous schemes. Shen and Chen [32] also proposed a new scheme in 2002 by combining the advantages of Akl-Taylor scheme [3] and Sandu's scheme [28], through their use of the Hash functions. This scheme was then improved by Das et al. [33] in 2005, through a reduction in both its computational overhead and security weaknesses. Tzeng et al. [34] continued with improvements in 2010 in order to reduce the computational time required for key generation and derivation. Sun and Liu [35] proposed another new scheme in 2004 and worked to prove that their scheme required less communication, computation, and storage.

Additional schemes have also been developed which focus more specifically on satisfying particular objectives. Tsai and Chang [36] had proposed a scheme which was based on such factors as: polynomial interpolation, the Chinese remainder, and the Rabin public key system. This scheme was specially designed to provide its users with the capability to freely change their secret keys and their positions within the hierarchy. The scheme was further improved by Kuo et al. [37] by work which enabled an increased

flexibility to change both security classes and keys. The scheme of Kuo [37] was further improved by Hwang [38] in 1999 and Chen and Chung [39] in 2002. First, Hwang was able to reduce the computations required for key generation and derivation. Then, Chen and Chung improved the scheme with regards to both the computation time and storage size.

Birget et al. [40] and Zhang and Wang [41] both proposed schemes that were not only related to the hierarchy of a set of users but also to the hierarchy of a set of data. These researchers worked to develop and/or generate an integral poset hierarchy in which the node in the hierarchy could be either a security class or a special category of data. Lin [42] proposed a scheme that can be efficiently implemented with low cost chips. Chien and Jan [43] presented a scheme similar to that of Lin's [42] but then further reduced the computational load and implementation costs. Ferrara and Masucci [44] designed their scheme from an information-theoretic approach in order to ensure that their scheme was unconditionally secure.

Tzeng [9] proposed the first time-bound hierarchical key management scheme in 2002. He used modular exponentiation, the Lucas function, and the one-way Hash function to enable the capability of time restrictions. The scheme was not only too complicated, but also vulnerable to collusion attacks [45]. Another time-bound hierarchical key management scheme [15] was proposed later by Chen, but quickly proved to be insecure against collusion attacks [46]. Bertino et al. [13] proposed another time-bound hierarchical key management scheme in 2008 through use of a tamper-resistant device, but this was again found to be vulnerable to collusion attack by Sun et al. [46] in 2009. Wang and Laih [16] then worked to propose an efficient time-bounded

hierarchical key management scheme in 2006 beginning with the concepts in the Aki-Taylor scheme [3]. Since they did not adequately formalize the definition of “security” in their research, a question remained as to whether their scheme was provably secure. In 2012, a provably-secure time-bounded hierarchical key management scheme was proposed by Ateniese et al. [12]. Ateniese et al. provided detailed steps for the security proof of their scheme. In the same year, Chen et al. [14] proposed another scheme which some believe to be efficient for defeating collision attacks without the necessity of a tamper-resistant device. With the information that has been gathered and the lessons learned regarding other time-bound hierarchical schemes, the security of Chen et al.’s scheme [14] has been comprehended as having not been breached as of yet.

2.4 Capabilities and Performances of Hierarchical Key Management Schemes and Time-Bound Hierarchical Key Management Schemes

From the evolutional history of both the hierarchical key management scheme and the time-bound hierarchical key management scheme, it is easy to observe that previous researchers had tried to improve the hierarchical key management scheme or time-bound hierarchical key management scheme by offering new schemes with either additional capabilities or a better performance rating. Among these previous efforts, the capabilities and levels of performance which these previous researchers sought to pursue may be summarized (by this Author) as follows:

1. Capabilities:

a) Capability of key derivation: the secret key assigned to a security class which can be used to compute secret keys of another security class if a binary relationship exists.

b) Capability of direct key derivation: if the secret key assigned to a particular security class is used to compute a secret key assigned to another security class, such key derivation can be done directly without generating any other secret keys.

c) Capability of access control (security): by using the scheme, users should be rejected from accessing data in those security classes in which they are not allowed access according to either hierarchy or that point in time. Since the data is protected by the secret keys, it requires that the process of key derivation cannot be reversed.

d) Applicability of the scheme: which establishes whether the scheme can be used under a more complicated policy protocol rather than a poset hierarchy.

2. Performance:

a) Complexity of key updates: involves the repercussion of key updates, including the number of steps, time, and how many security classes would be affected by the key updates.

b) Complexity of hierarchical change: involves the repercussion of adding or deleting a class or adding or deleting a binary relation between two security classes, including the number of steps, time, and how many security classes would be affected by the hierarchical change.

c) Computation overhead: involves the computation amount or time needed for key derivation.

d) Private storage requirement: involves the amount of storage required on the end devices.

e) Public storage requirement: involves the amount of storage required on the public servers.

The term “capabilities” works to describe those elements for which a hierarchical key management scheme is required to be capable. There are no particular orders of importance among the varying capabilities. Capabilities are known as the clear requirements that a scheme should satisfy. Different schemes will have various performance abilities when these schemes are compared with each other. According to the varying situations, some performance requirements will then become viewed as more important than others. If a scheme is implemented into an electronic library system, the importance of the complexity of hierarchical change would be minimal because it is rare to add or delete a new security class or a binary relation between two security classes in such system.

No existing scheme provides for the capability of user differentiation. The development of a new scheme is needed in order to overcome the challenges existent within a BYOD environment. The next section shall provide an analysis to determine which performance of the scheme shall be found to be most important in a BYOD environment and to verify why TLEBC is currently believed to be the most compatible scheme in a BYOD environment.

2.5 Requirements of Time-Bound Hierarchical Key Management Schemes in a Bring-Your-Own-Device Environment

Two major changes can be observed in a BYOD environment. First, the devices are no longer owned by the corporate companies. In the sense of a time-bound hierarchical key management scheme, this change indicates that the central authority no longer has full control over the end devices. Central authority has to cooperate with users in order to proceed with necessary changes, such as key updates or hierarchical changes, so a decline in the level of efficiency may be expected because permission from users must be sought and gained. The complexity of key updates and hierarchical changes frequently become more important when compared to the original working mode process.

Second, these devices are brought by individual users in a BYOD environment but not through unified purchase process (when all devices were under corporate ownership), which indicates that the quality of these devices cannot be guaranteed. To implement a time-bound hierarchical key management scheme, the central authority has to ensure that the scheme can operate efficiently even with the poorest devices. This requirement indicates that the computation overhead and private storage requirements become more important as compared to when the corporation owned all devices. A poorly designed device might not be able both to efficiently support a large computation overhead and meet private storage requirements.

The TLEBC is a time-bound hierarchical key management scheme proposed by Ateniese et al. [12]. It is believed to be the most compatible to the requirements of a time-bound hierarchical key management scheme within an BYOD environment by the author of this thesis because it performs superbly when one considers the complexity of

key updates, complexity of hierarchical changes, computation overhead, and private storage requirements. Through use of the TLEBC, whenever there is a need to change a key or do any modification to the hierarchy, in most cases these processes can be accomplished by changing the public data stored on the corporate server, which is owned by the central authority. It is only when a new security class has been added into the system, that the secret keys or private data must be updated. There is no need to change any secret keys or private data on the end devices. These changes of keys and hierarchy no longer require the corporation to gain permission from individual users. The scheme requires only one piece of private data to be stored on the end device. Thus, the private storage requirement is very small. The key derivation can be done in a single step of decryption through use of a symmetric cryptographic algorithm, so the computational overhead will also remain insignificant if an efficient symmetric cryptographic algorithm is used.

Basing on the excellent performances of the TLEBC with regards to the complexity of key updates, complexity of hierarchical changes, computation overhead, and private storage requirements, this scheme is believed to be the one which is the most compatible to the BYOD environment, for it merely lacks the capability of user differentiation.

2.6 Two-Level Encryption-Based Construction

This section briefly introduces the concept of the time-bound hierarchical key management scheme TLEBC. Here we shall recall the concept of the time-bound hierarchical key management scheme [12]. As was shown in section 2.2, assume that for each security class C_u and each short time period t_j, a secret key k_{u,t_j} is generated as an

encryption key in order to protect the data in said security class over a short time period. Each user would belong to a security class and have the access to data only in several short time periods. If λ_i is used to indicate a possible combination of such a short time period, then a set P is established, such that $\lambda_i \in \mathrm{P}$ is used to indicate such combinations λ_i [12]. If the security class C_v is used to indicate that designated security class within which the user works, then a user can be identified by both the security class C_v and the combination λ_i. A private value s_{v,λ_i} could be used to indicate the access authority by which that user is categorized in security class C_v within a set of short time periods λ_i [12].

TLEBC is built on a graph which presents the binary relationship between s_{v,λ_i} and k_{u,t_j} [12]. s_{v,λ_i} is listed in the superstratum and k_{u,t_j} is listed in the substratum. For any short time period t_j and combination of short time periods λ_i, if $t_j \in \lambda_i$ holds, then a path from s_{v,λ_i} to k_{u,t_j} will be constructed. Assume that there are three security classes R, M, and N, and R is authorized to access data within both M and N. Also assume that only two short time periods, t_1 and t_2, are considered; then there exists three possible combinations: λ_i such that $\lambda_1 = \{t_1\}$, $\lambda_2 = \{t_2\}$, and $\lambda_3 = \{t_1, t_2\}$. Figure 2.4 shows the resulting graph of the construction of the example.

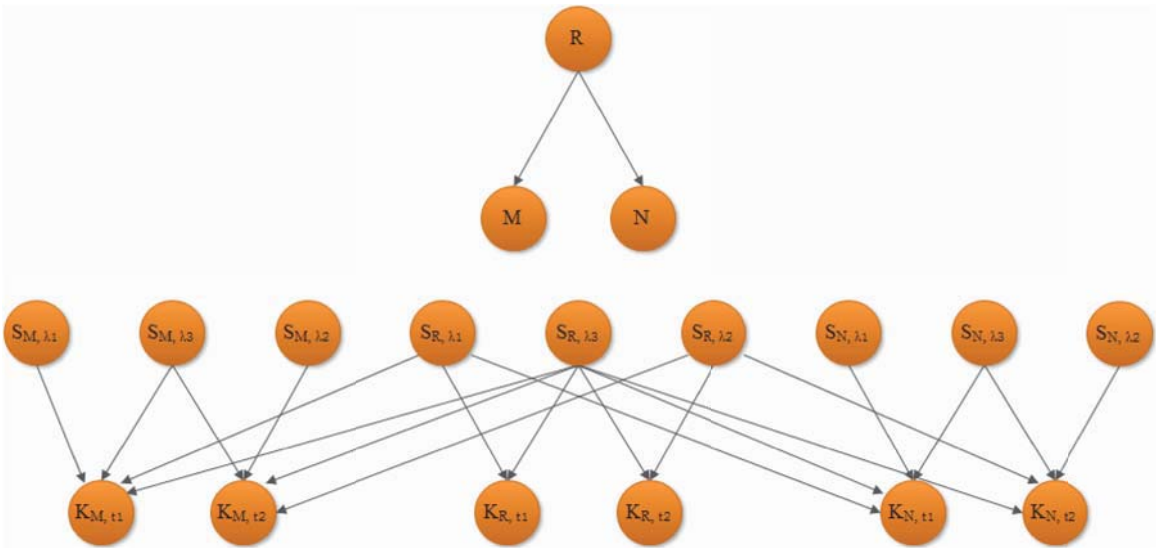

Figure 2.4 Example of a Transformation Graph for TLEBC

The TLEBC will assign one piece of s_{v,λ_i} to each user to indicate their access authorities and publish public data $p_{(s_{v,\lambda_i})(k_{u,t_j})}$ that could be used to compute encryption keys. If a user has access in all the security classes R, M, and N, and in both of the time periods t_1 and t_2, then the secret value S_{R,λ_3} will be assigned to the user to indicate that the user can have access to all of the security classes, since $M \le R$ and $N \le R$, and to all of the time periods, since $\lambda_3 = \{t_1, t_2\}$. To generate a specific key, for example K_{M,t_1}, in order to access data belonged to security class M within the short time period t_1, then the user will have the capacity to obtain public data $p_{(s_{R,\lambda_3})(k_{M,t_1})}$ and use both s_{v,λ_i} and $p_{(s_{R,\lambda_3})(k_{M,t_1})}$ to generate the specific key K_{M,t_1}.

CHAPTER 3. NEW SCHEME - UDTLEBC

This chapter presents how the new scheme – that of the User-Differentiable Two-Layer Encryption-Based Construction (UDTLEBC) – is built based on the basic concept of the TLEBC. The UDTELBC follows the definition of time-bound hierarchical key management (Gen, Der) and requires a symmetric encryption scheme $\Pi(\mathrm{K}, \mathrm{E}, D)$ as a necessary building part (See Appendix A, and Definitions 1 and 2 for detailed definitions of the time-bound hierarchical key management scheme and symmetric encryption scheme). UDTLEBC had the capability of user differentiation. This capability is also illustrated in this chapter.

3.1 User-Differentiated Two-Layer Encryption-Based Construction

Consider a minimal presentation graph $G = (V, E)$ of a poset hierarchy where V indicates the nodes in the poset hierarchy and E indicates the links from nodes to a set of various time combinations P, so that $\lambda_i \in \mathrm{P}$ where λ_i is a possible combination of short time periods t_i (introduced in section 2.6). A transformation graph $G_{\mathrm{PT}} = (V_{\mathrm{PT}}, E_{\mathrm{PT}})$, which presents the binary relationships existent between s_{v,λ_i} and k_{u,t_j}, can be built by transferring the minimal presentation graph $G = (V, E)$ with the following criteria introduced in Ateniese et al.'s work [12]:

1. For each class $v \in V$ and each combination of short time periods $\lambda \in \mathrm{P}$, a class v_λ is put in a set called V_P.

2. For each class $u \in V$ and each short time period $t \in T$, a class u_t is put in a set called V_T.

3. Let $V_\mathrm{PT} = V_\mathrm{P} \cup V_\mathrm{T}$.

4. For each class $v \in V$, each combination of short time periods $\lambda \in \mathrm{P}$, and each short time period $t \in T$, if $t \in \lambda$, an edge between v_λ and v_t is put in E_PT.

5. For each pair of classes $v \in V$ and $u \in V$, each time sequence $\lambda \in \mathrm{P}$, and each short time period $t \in T$, if there is a path between v and u in graph G and $t \in \lambda$, an edge between v_λ and u_t is put in E_PT.

6. Construct $G_\mathrm{PT} = (V_\mathrm{PT}, E_\mathrm{PT})$

This transformation graph $G_\mathrm{PT} = (V_\mathrm{PT}, E_\mathrm{PT})$ presents the mappings from s_{v,λ_i} to k_{u,t_j}. Since each of these users can be categorized into a specific class $v \in U_\mathrm{P}$ by assigning a private data $s_{v,\lambda}$, thus, the mappings can be found between users and the private data $s_{v,\lambda}$ may be assigned to them too. Let ID denote each user and a set V_I denote the users so that $ID \in V_\mathrm{IP}$, and let F_IP denote the mappings so that $f(ID) = s_{v,\lambda}$. From the view of the transformation graph, another part of graph $G_\mathrm{IP} = (V_\mathrm{IP}, F_\mathrm{IP})$ could be added on the basis of $G_\mathrm{PT} = (V_\mathrm{PT}, E_\mathrm{PT})$ and be used to enable the capability of user differentiation.

Assume that the hierarchy and time periods used in the example are those from section 2.6 and the five users and five mappings, employed to offer the visual assistance

provided with in Figure 3.1, are $f(ID_1) = s_{M,\lambda_1}$, $f(ID_2) = s_{R,\lambda_1}$, $f(ID_3) = s_{R,\lambda_3}$, $f(ID_4) = s_{R,\lambda_1}$, and $f(ID_5) = s_{N,\lambda_3}$. Figure 3.1 then shows the combination of $G_{\mathrm{IP}} = (V_{\mathrm{IP}}, F_{\mathrm{IP}})$ and $G_{\mathrm{PT}} = (V_{\mathrm{PT}}, E_{\mathrm{PT}})$ in UDTLEBC.

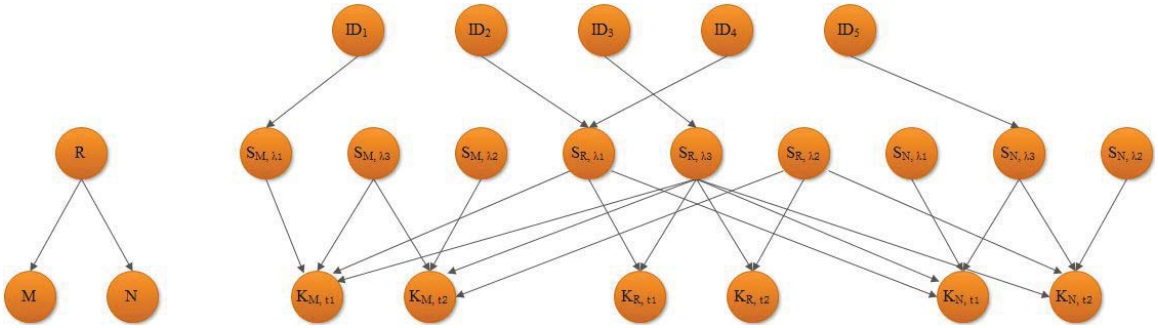

Figure 3.1 Example of a Transformation Graph for UDTLEBC

$G_{\mathrm{IP}} = (V_{\mathrm{IP}}, F_{\mathrm{IP}})$ presents the mappings from ID to s_{v,λ_i} . Thus, by assigning ID to users, a user can first generate the secret value s_{v,λ_i} , which indicates its access authority, and then computes the key k_{u,t_j} by using the secret value s_{v,λ_i} generated. One the basis of the construction of TLEBC [12], we modified the original algorithms and generated new algorithms for UDTLEBC as following:

By the transformation graphs $G_{\mathrm{IP}} = (V_{\mathrm{IP}}, F_{\mathrm{IP}})$ and $G_{\mathrm{PT}} = (V_{\mathrm{PT}}, E_{\mathrm{PT}})$, the graph of a poset hierarchy $G = (V, E)$, time periods that are $t \in T$, combinations of time periods that are $\lambda_i \in \mathrm{P}$, and a symmetric encryption scheme that is $\Pi = (\mathrm{K}, \mathrm{E}, D)$, the Gen and Der algorithms of UDTLEBC is built as follows:

Algorithm. $Gen\left(1^{\tau}, G, P\right)$

1. For each user, let $ID \leftarrow \mathrm{K}\left(1^{\tau}\right)$

2. For each class v_{λ} in V_{P}, let $s_{v,\lambda} \leftarrow \mathrm{K}\left(1^{\tau}\right)$;

3. For each class u_t in V_{T}, randomly choose a secret value $k_{u,t} \in \{0,1\}^{\tau}$;

4. Let I and k be the sequences of private information ID and $k_{u,t}$;

5. For any pair of classes $\left(ID, v_{\lambda}\right) \in V_{\mathrm{I}} \times V_{\mathrm{P}}$ such that $\left(ID, v_{\lambda}\right) \in F_{\mathrm{IP}}$, padding or splitting $s_{v,\lambda}$ to the length of λ and compute the public information $p_{(ID)(s_{v,\lambda})} = E_{ID}\left(S_{v,\lambda}\right)$;

6. For any pair of classes $\left(v_{\lambda}, u_t\right) \in V_{\mathrm{P}} \times V_{\mathrm{T}}$ such that $\left(v_{\lambda}, u_t\right) \in E_{\mathrm{PT}}$, compute the public information $p_{(s_{v,\lambda})(k_{u,t})} = E_{s_{v,\lambda}}\left(k_{v,t}\right)$;

7. Let pub be the sequence of public information computed in the previous step; and

8. Output $\left(ID, k, pub\right)$.

Algorithm $Der\left(1^{\tau}, G, P, u, v, \lambda, ID_i, t, pub\right)$

1. Extract the public value $p_{(ID)(s_{v,\lambda})}$ from pub;

2. Calculate the value $s_{v,\lambda} = D_{ID}\left(p_{(ID)(s_{v,\lambda})}\right)$;

3. Extract the public value $p_{(s_{v,\lambda})(k_{u,t})}$ from pub; and

4. Output the key $k_{v,t} = D_{s_{u,\lambda}}\left(p_{(s_{v,\lambda})(k_{u,t})}\right)$.

3.2 Capabilities of User Differentiation Through Use of UDTLEBC

Since each user is assigned a specific ID, each user can be differentiated from others because these users are no long sharing the identical private keys or values. From Figure 3.1 Example of a Transformation Graph for UDTLEBC we can now see that in order to remove any access authority or change any encryption key used, only the middle and bottom layer is required to be changed. Thus, all changes can be accomplished without modification of those IDs which have already been assigned and are in use, thereby permitting modifications to be done at the central authority side without unnecessary interference with the users. Now, we shall discuss each type of change in greater detail, in part by providing specific examples. The identical hierarchy and time periods that were employed in section 2.6 shall still be the standard for the following cases. The specific examples are as follows:

3.2.1 Add a new security class

Adding a new security class can be simply done by adding new secret values, keys, and mappings to the transformation graph. Assume that a new security class O is added into the hierarchy under security class M. Then, then the following modifications are needed:

1. For security class O, generate $s_{O,\lambda_i} \leftarrow \mathrm{K}\left(1^{\tau}\right)$ for λ_1, λ_2, and λ_3;

2. For security class O, select $k_{O,t_1} \in \{0,1\}^{\tau}$ and $k_{O,t_2} \in \{0,1\}^{\tau}$; and

3. Compute and publish $p_{(s_{R,\lambda_i})(k_{O,t_i})} = E_{s_{R,\lambda_i}}\left(k_{O,t_i}\right)$, $p_{(s_{M,\lambda_i})(k_{O,t_i})} = E_{s_{M,\lambda_i}}\left(k_{O,t_i}\right)$,

$p_{(s_{O,\lambda_i})(k_{O,t_i})} = E_{s_{O,\lambda_i}}\left(k_{O,t_i}\right)$, and for all λ_i and all k_{O,t_i}.

3.2.2 Delete an existing security class

Deleting an existing security class can be simply done by deleting all secret values, keys, and public data associated with the security class. Assume that the security class M is deleted from the hierarchy. Then, the following modifications are needed.

1. Remove s_{M,λ_i} for λ_1, λ_2, and λ_3;

2. Remove k_{M,t_1} and k_{M,t_2}; and

3. Remove $p_{(s_{M,\lambda_i})(k_{u,t_i})}$ for all s_{M,λ_i} and all k_{M,t_i}.

3.2.3 Update one key

Updating one key can be simply done by updating the key and relevant public data. If one assumes that k_{M,t_1} needs to be updated, then the following modifications are needed:

1. Update $K_{M.t_1}$: and

2. Update public data $p_{(s_{v,\lambda_i})(k_{M,t1})}$ for all existing mappings from s_{v,λ_i}.

3.2.4 Add a new user into the system

Adding a new user into the system can be done simply by generating and assigning a new ID to the new user and publishing public data according to that user's access authority. Assume a user is added to the security class and has access authorities R in short time periods t_1 and t_2. The following modifications are then needed to add this new user into the system:

1. Generate a new $ID \leftarrow \mathrm{K}(1^{\tau})$ for the new user;

2. Assign ID to the user; and

3. Generate and publish $p_{(ID)(s_{R,\lambda_3})}$.

3.2.5 Delete an existing user from the system

To delete a user from the system, all of the data that the user is able to obtain needs to be updated. Assume that a user who is in security class R and has access authorities in short time periods t_1 and t_2 is removed from the security class R and no longer has these access authorities. We will show that no other users will be affected in the process of doing such changes through the use of UDTLEBC. We now define the removed user by ID_0 and the other users who are in the security class R and have access authorities in short time periods t_1 and t_2 by ID_i, so that $i = 1,2,3,\ldots$. We will show that the removed user ID_0 would have no effect on each user ID_i and that the amendments can be done on the central authority side without participation from affected users.

Since user ID_0 previously had full accessibility to the entire system, the following modifications are needed to remove user ID_0 from the system:

1. For security class R, regenerate $s_{M,\lambda_3} \leftarrow \mathrm{K}(1^{\tau})$;

2. For security class R, M and N, reselect $k_{R,t_1} \in \{0,1\}^{\tau}$, $k_{R,t_2} \in \{0,1\}^{\tau}$, $k_{M,t_1} \in \{0,1\}^{\tau}$, $k_{M,t_2} \in \{0,1\}^{\tau}$, $k_{N,t_1} \in \{0,1\}^{\tau}$, and $k_{N,t_2} \in \{0,1\}^{\tau}$;

3. Update $p_{(ID_i)(s_{R,\lambda_3})} = \Sigma_{ID_i}(S_{R,\lambda_3})$ for each user ID_i; and

4. Update $p_{(s_{R,\lambda_3})(k_{u,t})} = \Sigma_{s_{R,\lambda_3}}\left(k_{v,t}\right)$ for each new key.

Thus, no matter what the change is, the ID assigned to users shall not be required to be updated and all changes can be accomplished from the side of central authority. Interactions with users are only needed in those cases when new users are added into the system. The users can obtain updated keys by using both the ID they once kept and all of the updated public data, allowing for ease of access without confusion.

CHAPTER 4. UDTLEBC: SECURITY PROOF

This chapter is designed to offer evidence regarding the security of the UDTLEBC. The chapter is organized as follows: First, this chapter presents definitions of security with respect to time-bound hierarchical keys and symmetric encryption schemes. Then, based on these definitions, the security level of TLEBC is introduced. After that, by comparing the differences which arise between UDTLEBC and TLEBC, it will be shown that UDTLEBC is as secure as TLEBC if the symmetric encryption scheme employed is as secure with respect to plaintext indistinguishability.

4.1 Definitions of Security and the Security of TLEBC

This section introduces related definitions of security that are used in [12] to prove the security of TLEBC, which might also be used to prove the security of UDTLEBC.

To ensure the security of access control, users should not be able to access a class during a specific time period if they are not authorized. A_v is used to denote the set that $\{u \in V : u \le v\}$, for any $v \in V$, where V is the set of security classes. That is to say, for every class $u \in V$ and time period $t \in T$, the key $k_{u,t}$ should not be achieved by users in class such that $u \notin A_v$ and users in class v' such that $u \in A_{v'}$ but authorized during a combination of time periods λ such that $t \notin \lambda$. The key $k_{u,t}$ should also be protected

against the coalition of these unauthorized users. The set $\{(u,\lambda) \in V \times P : u \notin A_v \text{ or } t \notin \lambda\}$ is used to present these unauthorized users, denoted by $F_{u,t}$.

When talking about the security of the time-bound hierarchical key management scheme, two different security goals are being considered at this point: security with respect to key indistinguishability and security against key recovery. These two security goals describe the security of the time-bound hierarchical key management scheme. The key indistinguishability postulates that an adversary can learn no information about a key of which the adversary does not have access. The key recovery claims that an adversary is not able to compute any key for which the adversary does not have access. The time-bound hierarchical key management scheme achieves these two security goals with respect to both static and adaptive adversaries, where static adversaries randomly choose a security class to attack and adaptive adversaries choose a specific security class to attack based primarily on that information the adversary actively sought or "jockeyed" for before the attack.

More precisely, regarding the behavior of these varied adversaries, a static adversary is allowed to access private information $s_{v,\lambda}$ such that $(v,\lambda) \in F_{u,t}$ and public information. An algorithm $Corrupt_{u,t}$ is used to denote the process that extracting secret values $s_{v,\lambda}$ has been associated with pairs $(v,\lambda) \in F_{u,t}$. The $corr$ is used to denote the output of $Corrupt_{u,t}(s)$, based on the private information s generated by Gen.

On the other hand, an adaptive adversary is allowed to access any number of users' private information of it's choice in advance and then chooses a security class u which it desires to attack and the time period t in which it wants the attack to be approached.

There are two other notions associated to the definition of security. The first is the idea of the probabilistic algorithm and experimentation. If $A(.,.,...)$ is a probabilistic algorithm, then $a \leftarrow A(x, y,...)$ denotes a running algorithm A with inputs x, y,... where the probability of the outcome a is like the flipping of the coins of A. If X is a set, then $x \leftarrow X$ is a denotation that selects an element uniformly from set X and assigns the value it to x. In another case, "$\leftarrow$" denotes a simple assignment. The second is the concept of negligibility. A function f is negligible if for every polynomial $p(.)$ there exists an N such that for the integer $n > N$ it holds that $f(n) < \frac{1}{p(n)}$.

Basing on the concepts illustrated above, the security level of the time-bound hierarchical key management scheme can be categorized into four types (See Appendix A, Definitions 3, 4, 5, and 6 for more detailed definitions of this security)[12]:

1. Security of key indistinguishability with respect to static adversaries (IND-ST),
2. Security of key indistinguishability with respect to adaptive adversaries (IND-AD),
3. Security against key recovery with respect to static adversaries (REC-ST), and
4. Security against key recovery with respect to adaptive adversaries (REC-AD).

Among these four definitions of security, some notions of security then apply to others. Four theorems are proven in [12] with the purpose to illustrate the relationships between these four notions of security, and they are organized in Figure 4.1. Relationships between the four notions are proven by four theorems. (See Appendix B for detailed introductions to the four theorems). It could be observed that adaptive adversaries do not have more advantages than do their static adversaries, so if a key

management scheme is seen to be secure with respect to its static adversaries, it shall also be secure with regards to its adaptive adversaries. It could also be observed that the security of key indistinguishability implies security against key recovery, but not vice versa. According to these two results, the security of IND-ST implies all of the other three types of securities, so that any hierarchical scheme which has the security of the IND-ST may be said to be secure in the sense of all four types of security.

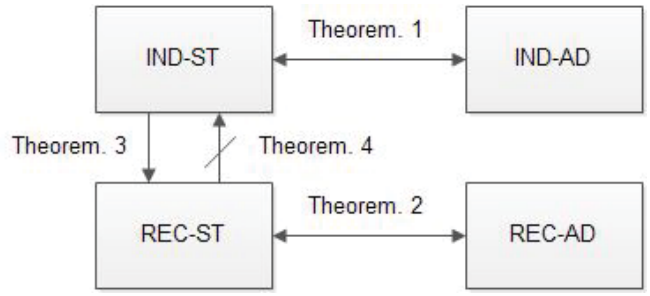

Figure 4.1 Relationships between Security Notions

The security of a symmetric encryption scheme used in UDTLEBC is also important because it is a core element to the construction of the UDTLEBC. The security for both TLEBC and UDTLEBC depends on the plaintext indistinguishability of the symmetric encryption scheme used in them. The plaintext distinguishability means that an adversary cannot differentiate between two plaintexts after they are encrypted. In other words, the encryption shall release no information regarding the plaintext that has been encrypted. The definition of the security of a symmetric encryption scheme is given by Ateniese et al.:

Definition 7 [IND-P1-C0]. Let $\Pi = (\mathrm{K}, \mathrm{E}, D)$ be a symmetric encryption scheme and let τ be a security parameter. Let $A = (A_1, A_2)$ be an adversary that has access to the

encryption oracle only during the first stage of the attack and never has access to the decryption oracle. Consider the following two experiments:

Experiment $Exp_{\Pi,A}^{IND-P1-C0-1}(1^\tau)$	Experiment $Exp_{\Pi,A}^{IND-P1-C0-0}(1^\tau)$
$key \leftarrow \mathrm{K}(1^\tau)$	$key \leftarrow \mathrm{K}(1^\tau)$
$(x_0, x_1, state) \leftarrow A_1^{\mathrm{E}_{key}(.)}(1^\tau)$	$(x_0, x_1, state) \leftarrow A_1^{\mathrm{E}_{key}(.)}(1^\tau)$
$y \leftarrow \mathrm{E}_{key}(x_1)$	$y \leftarrow \mathrm{E}_{key}(x_0)$
$d \leftarrow A_2(1^\tau, y, state)$	$d \leftarrow A_2(1^\tau, y, state)$
return d	return d

The advantage of A is defined as:

$$Adv_{\Pi,A}^{IND-P1-C0}(1^\tau) = \left|\Pr\left[Exp_{\Pi,A}^{IND-P1-C0-1}(1^\tau) = 1\right] - \Pr\left[Exp_{\Pi,A}^{IND-P1-C0-0}(1^\tau) = 1\right]\right|$$

The scheme is said to be secure in the sense of IND-P1-C0 (plaintext indistinguishability against an adversary whose time complexity is the polynomial in τ) if the advantage function $Adv_{\Pi,A}^{IND-P1-C0}(1^\tau)$ is negligible.

The $state$ in the experiment is some state information that could help the attack in the second stage. x_0 and x_1 are plaintext selected by the adversary and sent to the central authority. One of these is then encrypted by the central authority, and the ciphertext y relayed back to the adversary. The adversary will return 0 if the adversary posits that ciphertext y is encrypted by x_0 and 1 if the adversary posits that ciphertext y is encrypted by x_1. [12]

4.2 Security of UDTLEBC

This section illustrates how UDTLEBC is secure in the sense of IND-ST if the symmetric encryption scheme used is secure in the sense of IND-P1-C0. In order to

provide evidence for this statement, we will first prove that UDTLEBC is as secure as TLEBC if the symmetric encryption scheme is secure in the sense of IND-P1-C0.

Theorem 5. If the symmetric encryption scheme $\Pi = (K, E, D)$ is secure in the sense of IND-P1-C0, then the UDTLEBC is as secure as TLEBC.

Proof.

By comparing the construction of TLEBC and UDTLEBC, it could be observed that UDTLEBC has an extra element for the mapping from user identity ID to secret values $s_{v,\lambda}$. In the algorithm $Gen(1^{\tau}, G, P)$, when compared to TLEBC, UDTLEBC generates an extra part of public information $p_{(ID)(s_{v,\lambda})} = E_{ID}(S_{v,\lambda})$, while the other parts are identical as those found in TLEBC. In the algorithm $Der(1^{\tau}, G, P, u, v, \lambda, s_{v,\lambda}, t, pub)$, UDTLEBC has taken extra steps to compute the value $s_{v,\lambda}$ by $s_{v,\lambda} = D_{ID}(p_{(ID)(s_{v,\lambda})})$, while $s_{v,\lambda}$ is stored directly in TLEBC case. After obtaining the value $s_{v,\lambda}$, the concluding steps of UDTLEBC are identical to those taken with TLEBC. The only difference between TLEBC and UDTLEBC is found in the generation and usage of the public information $p_{(ID)(s_{v,\lambda})} = E_{ID}(S_{v,\lambda})$. The public information $p_{(ID)(s_{v,\lambda})}$ is used to compute the value $s_{v,\lambda}$ in UDTLEBC. If the computation of $s_{v,\lambda}$ in UDTLEBC does not have any effects on the usage of the $s_{v,\lambda}$ in the further elements of the UDTLEBC, which are identical to TLEBC, then UDTLEBC must be regarded as a system which is as secure as TLEBC. In other words, if the value $s_{v,\lambda}$ in UDTLEBC, which is associated with the generation and usage of $p_{(ID)(s_{v,\lambda})}$ to generate $s_{v,\lambda}$, is not distinguishable from the value $s_{v,\lambda}$ in TLEBC, then

UDTLEBC must be as secure as TLEBC. Since the value $s_{v,\lambda}$ in TLEBC and UDTLEBC is generated in an identical manner, then the previous statement implies that UDTLEC must be found to be as secure as TLEBC if $p_{(ID)(s_{v,\lambda})}$ similarly provides no information with regards to $s_{v,\lambda}$.

If $p_{(ID)(s_{v,\lambda})}$ were to provide any information regarding $s_{v,\lambda}$, then the value $s_{v,\lambda}$ could not be considered indistinguishable. Assume there is an adversary within two stages: $A=(A_1, A_2)$. Let the ID used to encrypt the value $s_{v,\lambda}$ be hidden from the adversary. The adversary is allowed to access to the encryption oracle $E_{ID}(.)$ during the first stage. In this stage, the adversary could generate some state information $state$ that might be helpful and could then be allowed to choose two values, $s_{v,\lambda}$ and $s'_{v,\lambda}$. This would require that the adversary could not encrypt $s_{v,\lambda}$ or $s'_{v,\lambda}$ and store it in state information $state$. Then, either one of the $s_{v,\lambda}$ and $s'_{v,\lambda}$ will be chosen randomly and encrypted by $E_{ID}(.)$ and a ciphertext y shall be generated as challenge. In the second stage, the adversary will lose its access to the encryption oracle. The ciphertext y is sent back to the adversary and the adversary has to decide whether the ciphertext y is encrypted by $s_{v,\lambda}$ or $s'_{v,\lambda}$. If the adversary believes that the ciphertext y is encrypted by $s_{v,\lambda}$, the adversary should reply 1. If the adversary believes that the ciphertext y is encrypted by $s_{v,\lambda}$, the adversary should reply 0. If we continue these experiment many times, the number of responses from the first should differ from the number of responses

from the second. Let $\Pi = (K, E, D)$ be a symmetric scheme and τ be the security parameter which is used in UDTLEBC, considering the following experiment:

Experiment $Exp_{\Pi,A}^{IND-UDT-1}(1^{\tau})$	Experiment $Exp_{\Pi,A}^{IND-UDT-0}(1^{\tau})$
$ID \leftarrow \mathrm{K}(1^{\tau})$	$ID \leftarrow \mathrm{K}(1^{\tau})$
$(s_{v,\lambda}, s'_{v,\lambda}, state) \leftarrow A_1^{\mathrm{E}_{ID}(.)}(1^{\tau})$	$(s_{v,\lambda}, s'_{v,\lambda}, state) \leftarrow A_1^{\mathrm{E}_{ID}(.)}(1^{\tau})$
$p_{(ID)(S_{v,\lambda})} \leftarrow \mathrm{E}_{ID}(s_{v,\lambda})$	$p_{(ID)(S'_{v,\lambda})} \leftarrow \mathrm{E}_{ID}(s'_{v,\lambda})$
$d \leftarrow A_2(1^{\tau}, p_{(ID)(S_{v,\lambda})}, state)$	$d \leftarrow A_2(1^{\tau}, p_{(ID)(S'_{v,\lambda})}, state)$
return d	return d

The advantage of A is defined as:

$$Adv_{\Pi,A}^{IND-UDT}(1^{\tau}) = \left|\Pr\left[Exp_{\Pi,A}^{IND-UDT-1}(1^{\lambda}) = 1\right] - \Pr\left[Exp_{\Pi,A}^{IND-UDT-0}(1^{\tau}) = 1\right]\right|$$

If the advantage of A that $Adv_{\Pi,A}^{IND-P1-C0}(1^{\tau})$ is negligible, then the value $s_{v,\lambda}$ should be indistinguishable. This implies that the public information $p_{(ID)(s_{v,\lambda})}$ shall also provide no information regarding the value $s_{v,\lambda}$. So, any usage of value $s_{v,\lambda}$ will not be affected with the generation and usage of $p_{(ID)(s_{v,\lambda})}$ to compute $s_{v,\lambda}$. Since $s_{v,\lambda}$ is generated in identical way in TLEBC and UDTLEBC, and there is no way to distinguish a value $s_{v,\lambda}$ from another value $s_{v,\lambda}$ associated with public information $p_{(ID)(s_{v,\lambda})}$. The value $s_{v,\lambda}$ in UDTLEBC is not distinguishable from the value $s_{v,\lambda}$ in TLEBC. In other words, the computation of $s_{v,\lambda}$ in UDTLEBC does not have any effects on the usage of the $s_{v,\lambda}$ in additional elements of UDTLEBC, which are identical to those found in TLEBC, so that the system of UDTLEBC must be as secure as that operating within TLEBC.

Comparing our experiment to the experiment within Definition 7, it could be observed that we have only changed the key to ID and the plaintext to value $s_{v,\lambda}$. The process of our experiment is indistinguishable from the process of a plaintext experiment. As long as the security of the symmetric encryption scheme with respect to plaintext indistinguishability holds, the advantage of the adversary in our experiment will necessarily be negligible. The theorem should be regarded as proven.

The security of TLEBC is proven by Ateniese et al. with the following theorem:

Theorem 6. If the encryption scheme $\Pi = (\mathrm{K}, D, \mathrm{E})$ is secure in the sense of IND-P1-C0, then the TLEBC is secure in the sense of IND-ST.[12]

According to Theorem 6, since the TLEBC is secure in the sense of IND-ST if the symmetric encryption scheme is secure in the sense of IND-P1-C0, then the UDTLEBC will also be secure in the sense of IND-ST if the symmetric encryption scheme is secure in the sense of IND-P1-C0 because UDTLEBC is as secure as TLEBC when the symmetric encryption scheme is secure in the sense of IND-P1-C0. Thus, Theorem 7 is obvious.

Theorem 7. If the encryption scheme $\Pi = (\mathrm{K}, D, \mathrm{E})$ is secure in the sense of IND-P1-C0, then the TLEBC is secure in the sense of IND-ST.

Since security in the sense of IND-ST implies all other three types of security, UDTLEBC is secure in the sense of all four types of security which we have defined in section 4.1.

CHAPTER 5. PERFORMANCE TEST METHODOLOGY

5.1 Overview

In this chapter methods for testing the performance of UDTLEBC was demonstrated. Since the UDTLEBC was built on the basis of TLEBC, the performance of UDTLEBC was also compared with the performance of TLEBC in order to evaluate how the proposed modifications would affect the performance. The all the results obtained were discussed to judge whether the performance of UDTLEBC is acceptable.

5.2 Methodology

As was introduced in second portion of section 2.4, elements of performance were observed with regard to five aspects: (1) complexity of key updates, (2) complexity of hierarchical changes, (3) computation overhead or time requirements, (4) private storage requirements, and (5) public storage requirements.

The private and public storage requirements were computed by the size of storage in Ateniese et al.'s work [12], so at this point we used the same methods to measure the comparison. Two elements may lead to failure in the storage requirements. The first includes the security classes designed by the central authority. The second includes those time periods decided by the central authority. To test these theories, we used V to

indicate the set of security classes and T to indicate the set of short time periods. The size of the storage were shown by giving an expression of the mathematical formula.

Previous works had described the complexity of key updates and the complexity of hierarchical change by providing examples as to how many steps and storage changes are required. Specific to our scheme, all of the key updates and hierarchical changes are about update storage and can be accomplished by the central authority. The complexity of key updates and complexity of hierarchical change are directly doomed by the public storage which is required to be changed. Because of this specialty, we used the size of storage that was required to be updated and which was associated with the key updates and hierarchical change to evaluate the complexity of the key updates and hierarchical changes. The size of storage that should be updated shall similarly be designated by a mathematical formula composited with a V and a T.

Experiments should test the computation overhead of UDTLEBC. Experiments are proceeded with two main objectives. The first objective is that we want to see whether the time cost of key derivation of UDTLEBC exceeded the simple human reaction time. We would like to see whether the processes of key derivation could be noticed by users and affected the user experiences. In this case we considered the simple human reaction time only, in which only one stimulus and one response existed [47]. The mean simple visual reaction time had been believed to be 180-200 msec by many researchers [48], so we were going to observe that whether the time cost of key derivation of UDTLEBC will bypass 180 msec. If not, the conclusion that UDTLEBC is efficient in the aspects of key derivation could be confirmed because the time cost of the process is not noticeable by users. The second objective is that we want to see how our modification, that building

UDTLEBC by adding new capabilities and steps to TLEBC, affected the computation overhead. The computation overhead is the computation amount or time required for key derivation. The following hypothesis is being considered:

H_0: The computation overhead of the UDTLEBC is twice of the computation overhead of TLEBC.

We had doubled the time costs in the TLEBC case and used non-parametric statistic to determine whether the distribution of time in the UDTLEBC case was significantly different to the distribution of time doubled in the TLEBC case, which provided answers to our hypothesis.

To allow for the accuracy of the experiments with the UDTLEBC, a demo was built to simulate the processes of key derivation from the UDTLEBC, and the time of such processes was recorded in order to observe the necessary computation time. The processes of key derivation for the UDTLEBC were introduced in section 3.1.

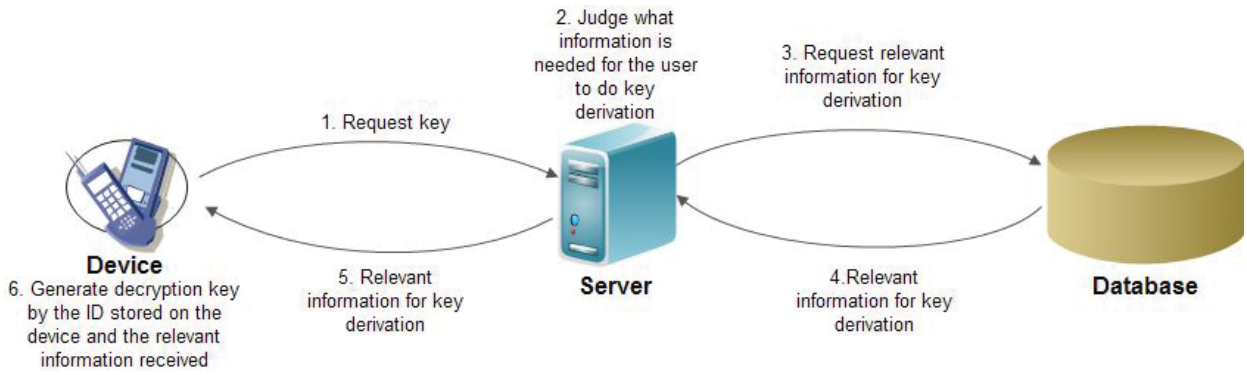

Figure 5.1 UDTLEBC Key Derivation Processes

Figure 5.1 illustrates the brief processes of key derivation for the UDTLEBC. When a device or user requests a key from the server, the server will judge what information is

required in order to compute which key that the device or user actually requires, to obtain the information from the database, and to then send such information back to the device or user. At that point, the device or user shall be able to compute the key it requires by using the ID and information it has received. The device or user has the advantages of the ID, which is a secret and private value distributed to said device or user when it is added into the system. (Results according to this process are shown from Figure 6.1 to Figure 6.12.)

To compare the performance of the UDTLEBC with the TLEBC, a demo of the TLEBC was also built to simulate the processes of key derivation from the TLEBC. The processes of key derivation from the TLEBC were introduced in section 2.6. Instead of an ID, a secret value was stored on the personal device.

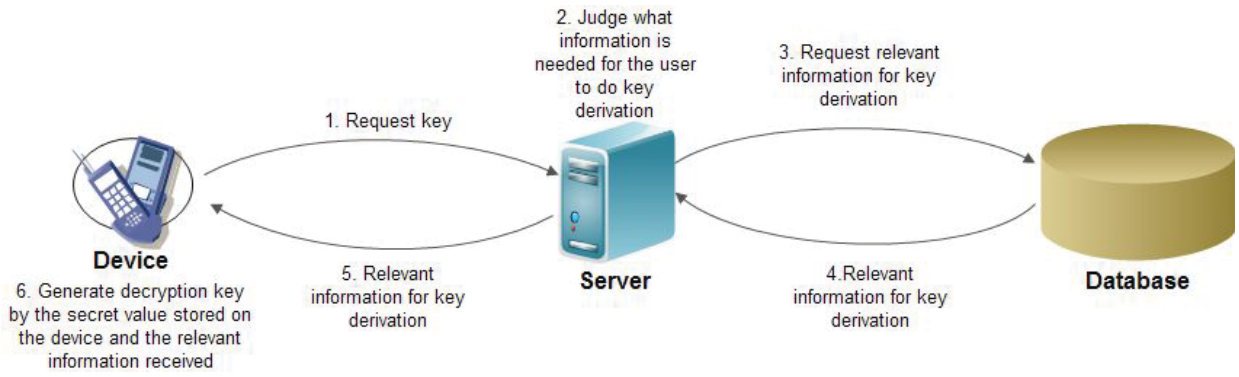

Figure 5.2 TLEBC Key Derivation Processes

Figure 5.2 illustrates the brief processes of key derivation from the TLEBC. Similar to the processes of key derivation with a UDTLEBC, when a device or user requests a key from the server, the server will judge what information is required, obtain such information from the database, and send it back to the device or user. In this time, instead of employing an ID as was done with the UDTLEBC, a secret value shall be stored on the

device and will be used to compute the key requested. This value is also distributed to the device or user while the device or user is added into the system. This value is not private but is shared with all users with the same access authorities. (Results according to this process are shown from Figure 6.1 to Figure 6.12.)

As both the symmetric encryption scheme and the device used might be found to affect the performance of the key derivation in this experiment, the following variables had been considered: the chosen symmetric encryption scheme, the operation mode of encryption, the chosen key size, and the testing environment or the configuration of the device.

Two testing environments were used. The performance were test on both the real device and the emulator. This scheme only used AES as the symmetric encryption scheme since the AES was viewed as the most popular, easily obtained, and provably-secure symmetric encryption scheme that existed. AES has three available key lengths. Since we were going to perform these experiments on Android operation systems, a key length of 128 bits had only been considered because many Android operation systems would not support key lengths of 196 or 256 bits. Similarly, we had only tested the CBC, CFB, and OFB modes of encryption, because the CTR mode was not supported by many of the Android operation systems. We also abandoned the ECB mode due to security concerns. These experiments were going to test the time required to run the processes of key derivation for both the UDTLEBC and TLEBC. The processes of key derivation were simulated 100 times with the purpose to avoid the influence of outliers.

Box-and-whisker plot diagrams were used for the results. Thus, the distribution of the computation time (overhead) could be discovered directly and one could judge whether the performance was acceptable.

We also tested the operation time of the entire processes of data transfer by using UDTLEBC. This is designed to evaluate whether we can expand the usage of UDTLEBC not only for key management but also for data transfer protection. In this case, a piece of data was encrypted, transferred, decrypted, and, finally, displayed. Since the processes of key derivation were also involved in the processes of data transfer, the same variables and methods which were used for key derivation should also be used here. In this case we only performed the simulation on the Samsung GT-i9108 device.

During experimentation, five lengths of data were considered: 1Kbit, 10Kbit, 100Kbit, and 1Mbit, and 10Mbit. Data lengths longer than 10Mbit were not tested because these are too long to be transferred in one communication. The time required for transfer of these data lengths were recorded in order to evaluate how the UDTLEBC performs if it was used to transfer data. Working with control groups, the times required to transfer data of the same length directly was also tested. With this method, we determined how much extra time was needed to provide protection to data transfer through use of the UDTLEBC.

Figure 5.3 illustrates the processes of the data transfer which we have simulated. When a device or user requests data from the server, the server will judge the specific encryption key that should be used to encrypt the data and determine what information is needed for the device or user to compute the encryption key by key derivation. Then, the server will obtain that data, encryption key, and information required in order to compute

the encryption key from the database. After encrypting the data with the encryption key, the server will send the encrypted data and relevant information for key derivation to the device or user. After receiving the encrypted data and the information of key derivation, the device or user will first proceed with the key derivation to obtain the key and will then proceed with decryption through use of the computed key to obtain the data required. (Process results are illustrated in Figure 6.3, Figure 6.4, Figure 6.5, Figure 6.6, and Figure 6.7).

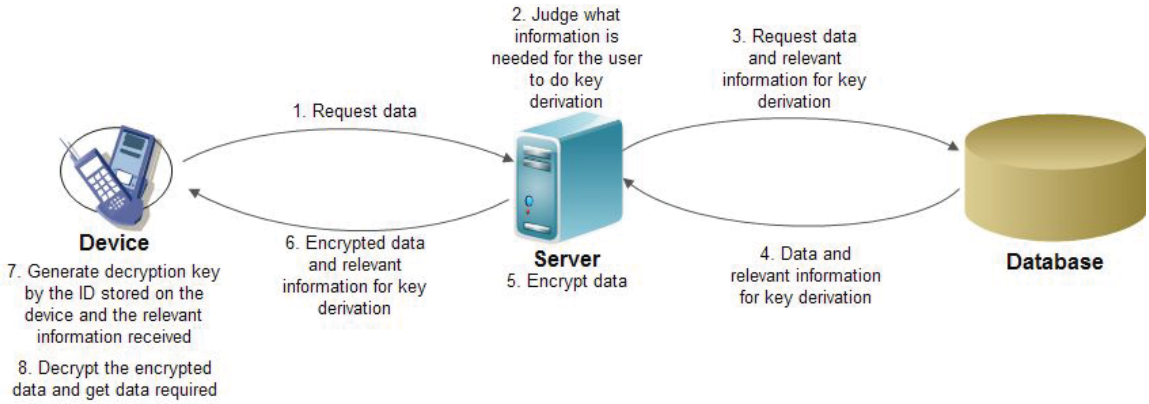

Figure 5.3 UDTLEBC Data Transfer Processes

5.3 Independent Variables and Constants

The independent variables were of different configurations during the experiments. We expected to observe how the performances are going to be affected. Though many factors were considered during the experiments, according to the capability of both the Android and the device or emulator used, those factors which varied during the experiments were merely partial, while the rest of the factors had remained as constants. For the experiments of key derivation, varied configurations included: the different time-bound hierarchical key management scheme, the different operation mode of encryption,

and the testing environment or the configuration of the device. The encryption scheme AES and key length of 128bit were both treated as constants by this author as they had not varied during these experiments. For the experiments of data transfer, varied configurations included: the different operation mode of encryption and the different data length transferred. The device Samsung GT-i9108, the AES encryption scheme, and the key length of 128 bit were treated as constants for these did not vary during the experiments.

5.4 Dependent Variables

The dependent variables were the computation times observed in varied configurations of the experiment. These computation times demonstrated directly how UDTLEBC performs in each varied configuration for either the key derivation or data transfer.

5.5 Internal Validity

Unmanipulated factors, such as noise, might override the influence of the manipulated variables. It was still apparent that one may consider how UDTLEBC performs by observing the computation time with these factors. With regard to unmanipulated factors, such as noise, the manipulated variables which had a significant influence on performance would still exist, and such influences could be considered more trustworthy because these influences existed even in situations involving the unmanipulated factors, such as noise.

5.6 External Validity

All of the results had been obtained by using a convenient emulator and a convenient device, so that additional threats to external validity should be considered.

Detailed information regarding the emulator and the device had been mentioned, so if one uses different configurations one can predict the results by comparing the capability of the emulator or device one is using. The emulator and the device used in this thesis was set to be at a level far lower when compared to the average level of devices used in 2014, so better results could be expected if different emulators or devices were used. So, if the performance levels established in this thesis were found to be acceptable, it was expected that said levels should be acceptable in most cases.

CHAPTER 6. RESULTS

6.1 Performance of Complexities and Storage Requirements

The Table 6.1 provides a summary of the performance of UDTLEBC on complexities and storage requirements. The results and expression of mathematical formula of TLEBC is obtained from Ateniese et al.'s paper [12]. Specifically, V is used to indicate the set of security classes, N is used to indicate the set of users, and T is used to indicate the set of short time periods.

From the construction of UDTLEBC, it is obvious that only one piece of private storage is required for this system. The private storage requirements of UDTLEBC are identical to those of TLEBC. When compared with TLEBC, UDTLEBC only demands extra public information in order to successfully realize the mapping from users to secret values. Each user has one mapping to one specific secret value, so $|N|$ extra public storage is necessary. As compared to $|V^2| \bullet |T|^3$, the $|N|$ extra public storage is actually rather small. If one imagines a small system with just 10 security classes, 10 short time periods, and 10 users, the percentage of $|N|$ extra public storage may be seen as merely:

$$\frac{|N|}{|V|^2 \bullet |T|^3 + |N|} = \frac{10}{10^2 \bullet 10^3 + 10} \approx 0.01\%$$

Table 6.1 Performance Comparison Table of TLEBC and UDTLEBC

Scheme	Complexity of key updates	Complexity of hierarchical change		Private storage requirement	Public storage requirement
		Change one security class	Change one binary relation		
TLEBC[12]	At most $\|V\| \bullet \|T\|$ public storage	At most affects $\|T\|$ keys, $\left\|T^3\right\|$ public storage, and $O(1)$ private storage	At most affects $\left\|T^3\right\|$ public storage	O(1)	At most $\left\|V^2\right\| \bullet \|T\|^3$
UDTLEBC	Is the same as TLEBC	Is the same as TLEBC (Needs no interactions with users)	Is the same as TLEBC	Is the same as TLEBC	At most $\left\|V^2\right\| \bullet \|T\|^3 + \|N\|$

Even though UDTLEBC requires the $|N|$ extra public storage, this does not appear to greatly affect its performance in terms of public storage requirements when compared to those of the TLEBC. By using the formula of public storage, one could also observe that neither UDTLEBC nor TLEBC performs very well from this perspective. For instance, according to the example shown above, a small system with merely 10 security classes, 10 short time periods, and 10 users would require, at most, $|V^2| \bullet |T|^3 + |N| = 10010$ public records to be stored. This is a deficiency for both UDTLEBC and TLEBC. This deficiency requires improvements in future work. As the author of this thesis has noted, within a BYOD environment, the performance of public storage is not interpreted as an important factor, although it still may have significant implications for the future usage of UDTLEBC.

The complexities of key updates and hierarchical change both depend upon the existent relationships between secret values and keys. According to the construction of UDTLEBC, it becomes obvious that the extra layer of users simply adds to the number of relationships between users and secret values and does not affect the actual relationships between secret values and keys. So the complexities of key updates and hierarchical change in UDTLEBC may be viewed as quite similar to the complexities of key updates and hierarchical change in TLEBC. UDTLEBC has one distinct advantage when it is compared to TLEBC in that it does not require actual interactions with the users while performing all of these changes. Changes to the size of $|V| \bullet |T|$ due to the complexity of key updates and to $|T|^3$ as a result of the complexity of hierarchical change seem to be great, but they are only those changes which occur in the worst cases. That situation in

which $|V| \bullet |T|$ would arise assumes that a root key has changed and $|T|^3$ assumes that the changes are being made to top security classes within the partial order hierarchy. Once we add the advantage where all of the changes can be made on server's side, without any unnecessary interactions with the users, so that all changes can be achieved within predefined policies, it then becomes fair to expect that the key updates and hierarchical changes can be accomplished efficiently within our real world parameters.

6.2 Performance of Computation Overhead

The time distributions of TLEBC and UDTLEBC were recorded and compared to see the computation overhead of the UDTLEBC. We considered 3 encryption modes including CBC, CFB and OFB. Two devices were considered. One device was the SAMSUNG GT-i9108 with the operating system Android 2.3.5, I9108ZMKI5 and CPU Samsung S5PV310, Cortex-A9, and a RAM of 1GB. Another testing environment was simulated by the Java SDK. The Java SDK simulated a platform of a 3.2" QVGA ADP2 with an operating system for an Android 2.2, the CPU ARM (armeabi), and a 512 MB RAM. In total the time distribution of TLEBC and UDTLEBC were considered in 6 cases.

Figure 6.1 shows the time distribution of key derivation of TLEBC and UDTLEBC in the case of using encryption scheme AES, key length of 128 bits, device GT-i9108, and CBC encryption mode. One can observe that the average time of key derivation of UDTLEBC is 30-40 msec, which is far from the simple human visual reaction time 180 msec. It can be confirmed that the computation overhead of UDTLEBC in this case is efficient because the time cost of the processes could not be noticed by users. The result of the non-parametric test is shown in Figure 6.2. Our Null hypothesis was retained.

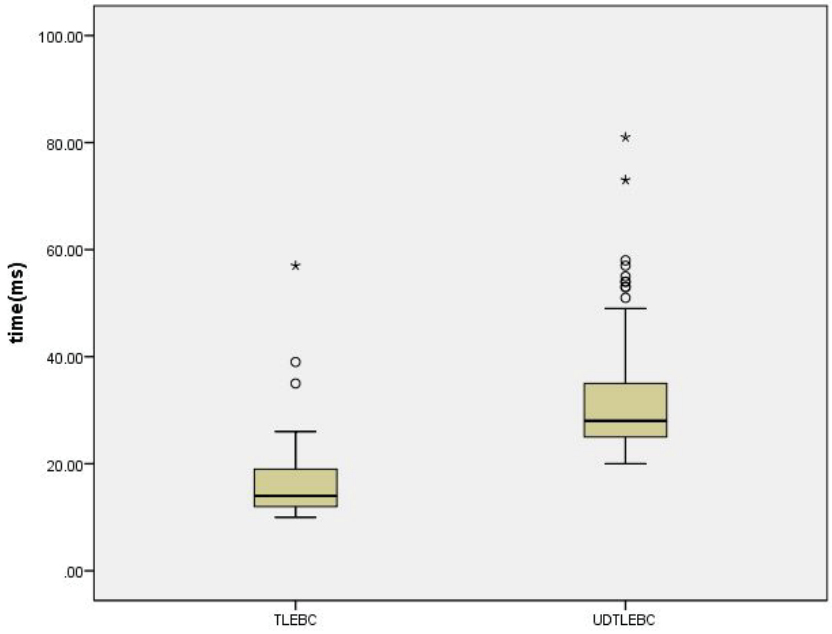

Figure 6.1 Time Distribution of key derivation of TLEBC and UDTLEBC in the Case of Device GT-i9108 and CBC Encryption Mode

Hypothesis Test Summary

	Null Hypothesis	Test	Sig.	Decision
1	The distribution of Time is the same across categories of Scheme.	Independent-Samples Mann-Whitney U Test	.430	Retain the null hypothesis.

Asymptotic significances are displayed. The significance level is .05.

Figure 6.2 Non-parametric Test for Distribution of Time in the Case of Device GT-i9108 and CBC Encryption Mode

Figure 6.3 shows the time distribution of key derivation of TLEBC and UDTLEBC in the case of using encryption scheme AES, key length of 128 bits, device GT-i9108, and CFB encryption mode. One can observe that the average time of key derivation of UDTLEBC is also 30-40 msec, which is far from the simple human visual reaction time

180 msec. It can be confirmed that the computation overhead of UDTLEBC in this case is efficient because the time cost of the processes could not be noticed by users. The result of the non-parametric test is shown in Figure 6.4. Our Null hypothesis was rejected.

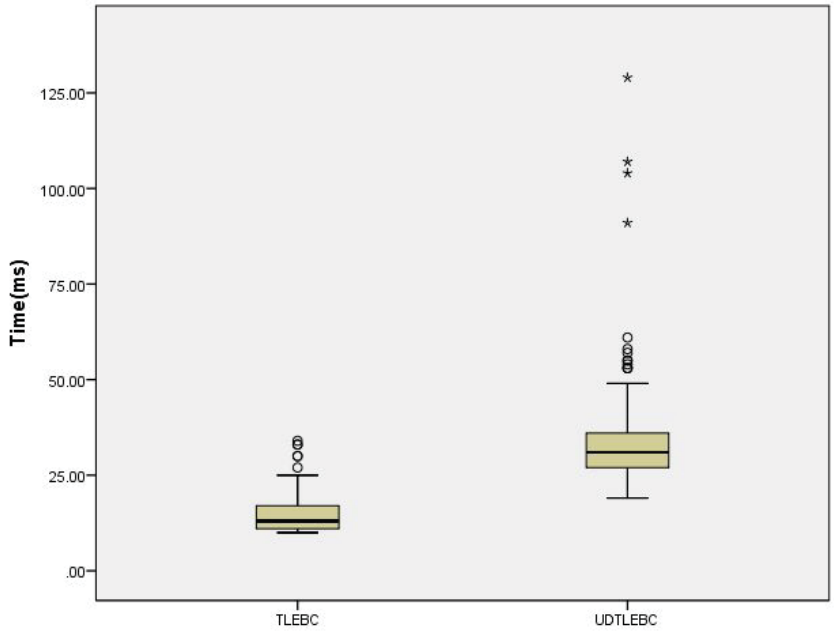

Figure 6.3 Time Distribution of key derivation of TLEBC and UDTLEBC in the Case of Device GT-i9108 and CFB Encryption Mode

Hypothesis Test Summary

	Null Hypothesis	Test	Sig.	Decision
1	The distribution of Time is the same across categories of Scheme.	Independent-Samples Mann-Whitney U Test	.000	Reject the null hypothesis.

Asymptotic significances are displayed. The significance level is .05.

Figure 6.4 Non-parametric Test for Distribution of Time in the Case of Device GT-i9108 and CFB Encryption Mode

Figure 6.5 shows the time distribution of key derivation of TLEBC and UDTLEBC in the case of using encryption scheme AES, key length of 128 bits, device GT-i9108,

and OFB encryption mode. One can observe that the average time of key derivation of UDTLEBC is also 30-40 msec, which is far from the simple human visual reaction time 180 msec. It can be confirmed that the computation overhead of UDTLEBC in this case is efficient because the time cost of the processes could not be noticed by users. The result of the non-parametric test is shown in Figure 6.6. Our Null hypothesis was retained.

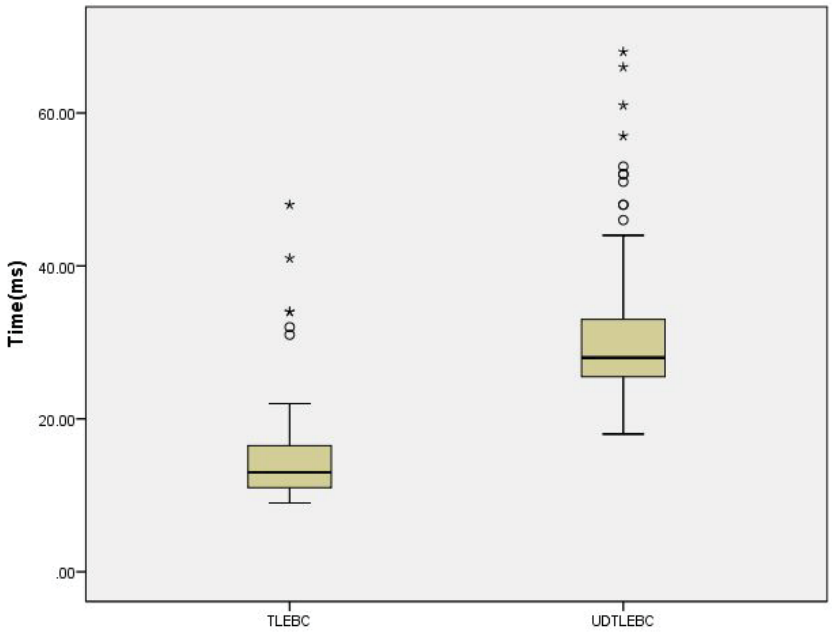

Figure 6.5 Time Distribution of key derivation of TLEBC and UDTLEBC in the Case of Device GT-i9108 and OFB Encryption Mode

Hypothesis Test Summary

	Null Hypothesis	Test	Sig.	Decision
1	The distribution of time is the same across categories of scheme.	Independent-Samples Mann-Whitney U Test	.061	Retain the null hypothesis.

Asymptotic significances are displayed. The significance level is .05.

Figure 6.6 Non-parametric Test for Distribution of Time in the Case of Device GT-i9108 and OFB Encryption Mode

Figure 6.7 shows the time distribution of key derivation of TLEBC and UDTLEBC in the case of using encryption scheme AES, key length of 128 bits, emulator Java SDK, and CBC encryption mode. One can observe that the average time of key derivation of UDTLEBC is 200-350 msec, which exceed the simple human visual reaction time 180 msec. It can be confirmed that the computation overhead of UDTLEBC in this case is not efficient because the time cost of the processes could be noticed by users. The result of the non-parametric test is shown in Figure 6.8. Our Null hypothesis was retained.

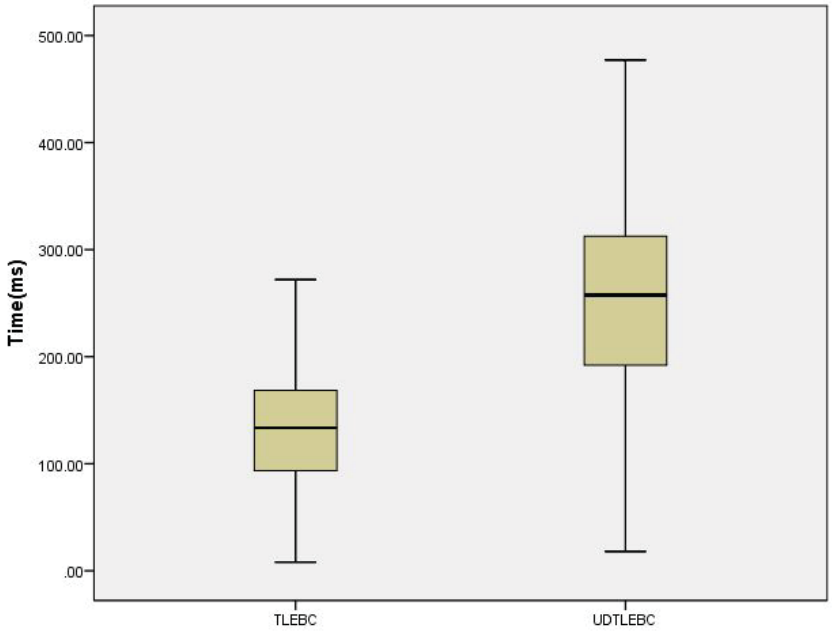

Figure 6.7 Time Distribution of key derivation of TLEBC and UDTLEBC in the Case of Java SDK and CBC Encryption Mode

Hypothesis Test Summary

	Null Hypothesis	Test	Sig.	Decision
1	The distribution of time is the same across categories of Scheme.	Independent-Samples Mann-Whitney U Test	.313	Retain the null hypothesis.

Asymptotic significances are displayed. The significance level is .05.

Figure 6.8 Non-parametric Test for Distribution of Time in the Case of Java SDK and CBC Encryption Mode

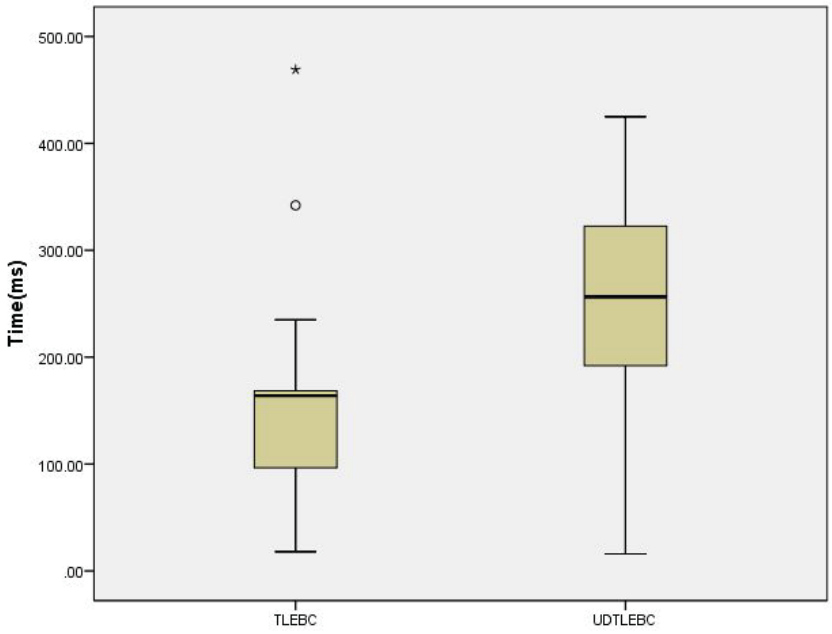

Figure 6.9 Time Distribution of key derivation of TLEBC and UDTLEBC in the Case of Java SDK and CFB Encryption Mode

Figure 6.9 shows the time distribution of key derivation of TLEBC and UDTLEBC in the case of using encryption scheme AES, key length of 128 bits, emulator Java SDK, and CFB encryption mode. One can observe that the average time of key derivation of UDTLEBC is also 200-350 msec, which exceed the simple human visual reaction time

180 msec. It can be confirmed that the computation overhead of UDTLEBC in this case is not efficient because the time cost of the processes could be noticed by users. The result of the non-parametric test is shown in Figure 6.10. Our Null hypothesis was rejected.

Hypothesis Test Summary

	Null Hypothesis	Test	Sig.	Decision
1	The distribution of time is the same across categories of scheme.	Independent-Samples Mann-Whitney U Test	.001	Reject the null hypothesis.

Asymptotic significances are displayed. The significance level is .05.

Figure 6.10 Non-parametric Test for Distribution of Time in the Case of Java SDK and CFB Encryption Mode

Figure 6.11 shows the time distribution of key derivation of TLEBC and UDTLEBC in the case of using encryption scheme AES, key length of 128 bits, emulator Java SDK, and OFB encryption mode. One can observe that the average time of key derivation of UDTLEBC is 200-350 msec, which exceed the simple human visual reaction time 180 msec. It can be confirmed that the computation overhead of UDTLEBC in this case is not efficient because the time cost of the processes could be noticed by users. The result of the non-parametric test is shown in Figure 6.12. Our Null hypothesis was retained.

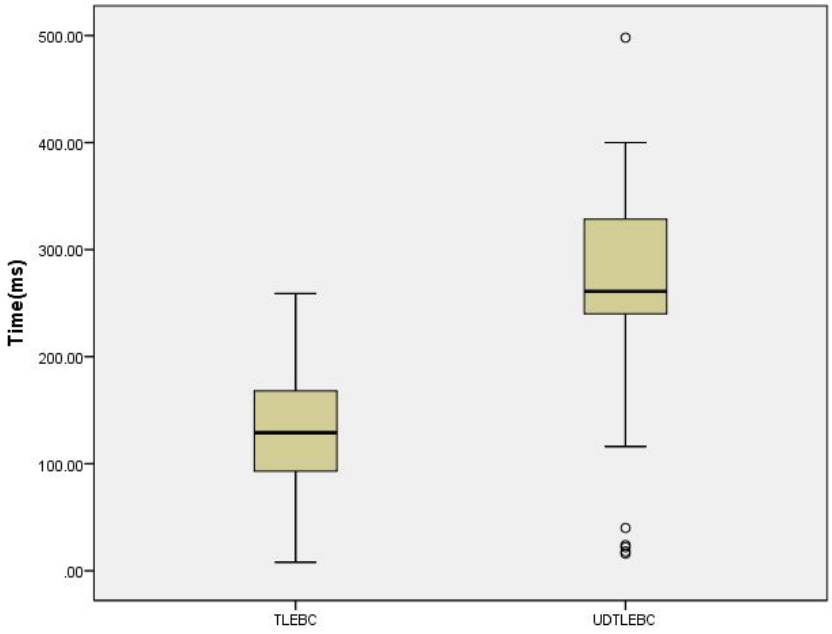

Figure 6.11 Time Distribution of key derivation of TLEBC and UDTLEBC in the Case of Java SDK and OFB Encryption Mode

Hypothesis Test Summary

	Null Hypothesis	Test	Sig.	Decision
1	The distribution of time is the same across categories of scheme.	Independent-Samples Mann-Whitney U Test	.446	Retain the null hypothesis.

Asymptotic significances are displayed. The significance level is .05.

Figure 6.12 Non-parametric Test for Distribution of Time in the Case of Java SDK and OFB Encryption Mode

It could be observed that the Java SDK emulator performs much less effectively than does the Samsung GT-i9108 device. While the performance of UDTLEBC is efficient in the case of Samsung GT-i9108, the performance of UDTLEB is not efficient in the case of using the emulator Java SDK. It could also be observed that the standard

deviation (SD) of the computation overhead for the Java SDK emulator is quite large. In the case of the OFB mode, the deviation is even greater than half of the mean. So one could conclude that the results generated through the use of the Java SDK are not trustworthy. From the other side, the Samsung GT-i9108 is a “real” device while the Java SDK is considered an “emulator”, so the results gathered by the GT-i9108 are more likely to reflect the real performance of the computation overhead. From this perspective, we shall abandon the results gathered from the Java SDK and select the results obtained from the GT-i9108 device. In addition, the Samsung GT-i9108 is a device that came on the market at end of 2011, so it may already be viewed by many as an “old and used” device. For devices which have entered the market after 2011, a better performance ratio could be expected. With this fact in mind, confidence may also be generated that the performance of UDTLEBC will be even more acceptable on phones that have appeared on the market after 2011. Thus, the performance of UDTLEBC in the aspect of computation overhead is believed efficient.

With the results gathered in the non-parametric test, we have failed to reject our hypothesis in the CBC and OFB encryption mode cases, and we have succeeded to reject our hypothesis in the CFB encryption mode case. Thus, our results support our hypothesis that the computation overhead of the UDTLEBC is twice the computation overhead of TLEBC in both the CBC and OFB encryption mode cases, while they do not similarly support the hypothesis in the OFB encryption mode case.

6.3 Performance of Data Transfer

The results of performance of the entire processes of data transfer are shown from Figure 6.13 to Figure 6.17. Table 6.2 illustrates the varied median times of data transfer performance.

Table 6.2 Median Time of Data Transfer Performance Test

	No Encryption	CBC Mode	CFB Mode	OFB Mode
1Kbit	14 ms	45.5 ms	43.5 ms	45 ms
10Kbit	16 ms	47 ms	49 ms	52 ms
100Kbit	50 ms	105 ms	102 ms	104 ms
1Mbit	322 ms	456 ms	432.5 ms	444 ms
10Mbit	7.23 s	10.81 s	10.84 s	10. 30 s

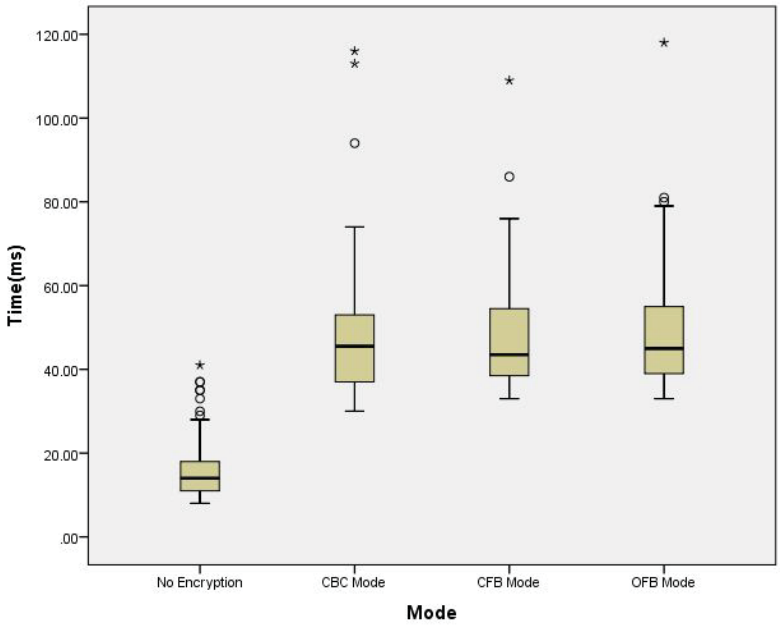

Figure 6.13 Data Transfer Performance for Data Length of 1Kbit

Figure 6.13 shows the data transfer performance for a data length of 1Kbit on the Samsung GT-i9108 device. These results have been achieved by following the processes in Figure 5.3. The time to transfer the data directly is around 14 ms. To transfer data by using UDTLEBC for protection, the median time is 45.5 ms for the CBC mode, 43.5 ms for the CFB mode, and 45 ms for the OFB Mode.

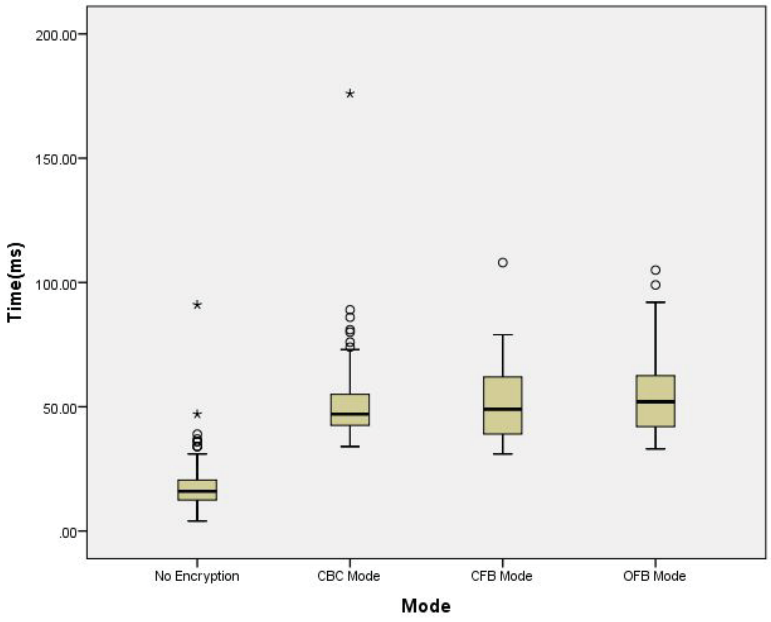

Figure 6.14 Data Transfer Performance for Data Lengths of 10Kbit

Figure 6.14 shows the data transfer performance for a data length of 10Kbit on the Samsung GT-i9108 device. The results are achieved by following the processes illustrated in Figure 5.3. The calculated time to directly transfer the data is approximately 16 ms. In order to transfer data by using UDTLEBC for protection, the median time increases to 47 ms increases to with the CBC mode, 49 ms with the CFB mode, and 52 ms with the OFB Mode. When one compares the results of this figure with the results of Figure 6.13, one should notice that although the data length has been extended by 10

times, the time of data transfer has only slightly increased, with the multiple nowhere near that of 10. One possible explanation is that most of the time involved is the actual costs of the embellishment of the communication, so that a mere increase in the data length itself should not affect the actual time of data transfer.

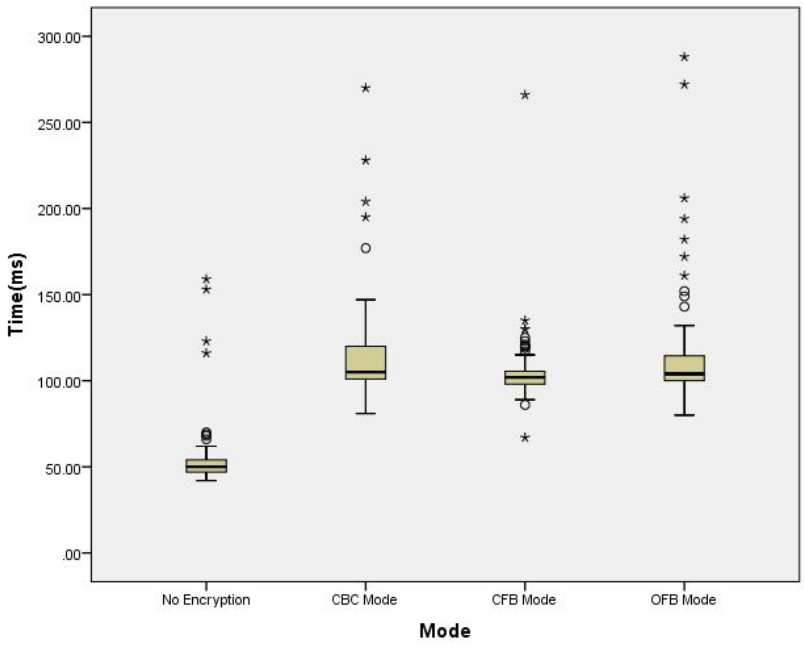

Figure 6.15 Data Transfer Performance for Data Lengths of 100Kbit

Figure 6.15 shows the data transfer performance for data lengths of 100Kbit on the Samsung GT-i9108 device. These results are achieved by following the processes shown in Figure 5.3. In this case, the time to transfer the data directly is around 50 ms. In order to directly transfer the gathered data by using UDTLEBC for protection, the median time is 105 ms for the CBC mode, 102 ms for the CFB mode, and 104 ms for the OFB Mode. When one compares the results gathered from this case to the results from the previous two cases, one could observe that the time required to transfer data increases a great deal.

With this information in hand, we can conclude that this data length is long enough to affect the time of data transfer.

It is interesting to mention that the ratio of transferring data by using UDTLEBC as a protection measure as opposed to transferring data directly is decreased with the increase of data length. This trend can be better observed by viewing Table 6.2. For data lengths of between 1Kbit and 10Kbit, it could be observed that the time to transfer data while using UDTLEBC as a protection becomes tripled in relationship to the time required to transfer the data directly. But for data lengths of 100Kbit, the time to transfer data through the use of UDTLEBC as a protection is only twice as the length of time required for direct data transfer.

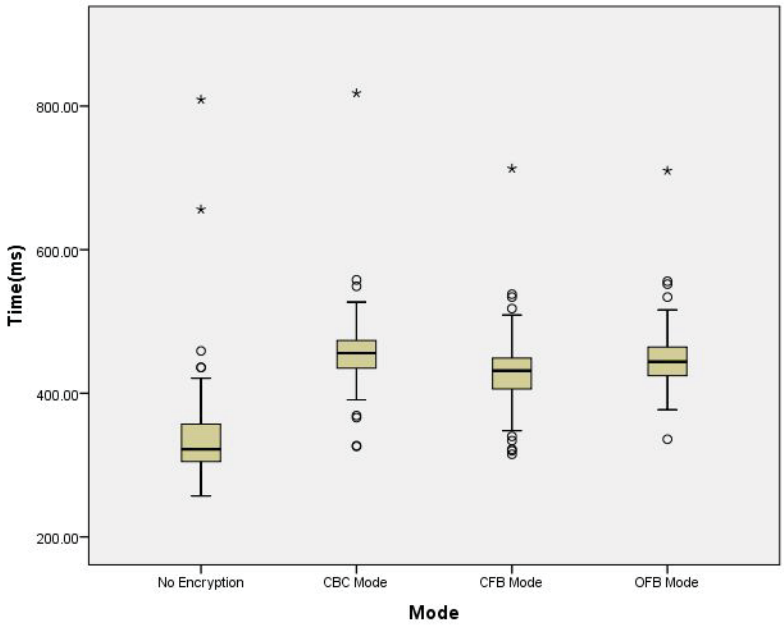

Figure 6.16 Data Transfer Performance for Data Lengths of 1Mbit

Figure 6.16 illustrates the data transfer performance for a data length of 1Mbit on the Samsung GT-i9108 device. These results are achieved by following the processes

shown in Figure 5.3. With this case, the time to transfer the data directly is approximately 322 ms. When one shifts to transferring data by using UDTLEBC for protection, the median time increases to 456 ms for the CBC mode, 432.5 ms for the CFB mode, and 444 ms for the OFB Mode. As compared to the previous three figures, we can now see that the increase of data length has caused an even greater impact on time costs. More importantly, the ratio of transferring data through the use of UDTLEBC as a protection for direct data transfer has further decreased.

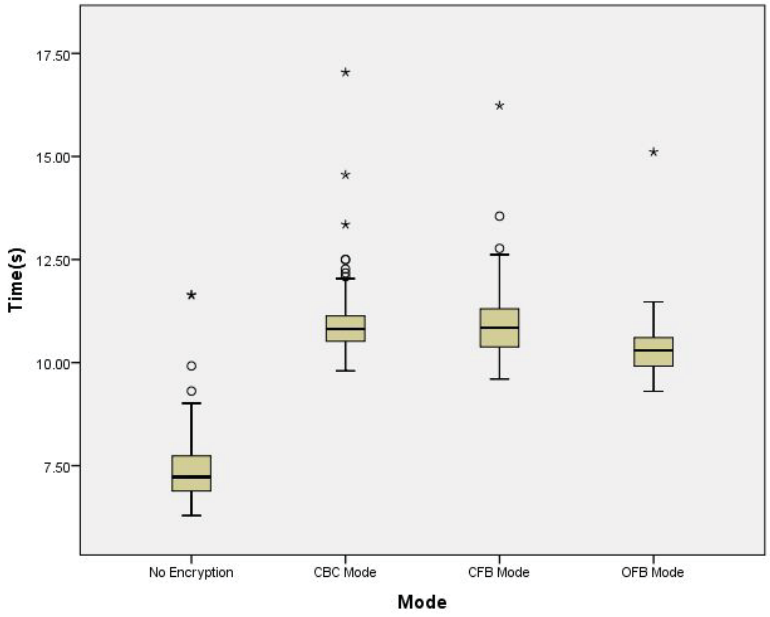

Figure 6.17 Data Transfer Performance for Data Lengths of 10Mbit

Figure 6.17 illustrates the data transfer performance for a data length of 10Mbit on the Samsung GT-i9108 device. These results have been achieved by following the processes shown in Figure 5.3. The time to transfer the data directly is approximately 7.23 seconds. In order to transfer data by using UDTLEBC as a protection device, the median time further increases 10.81 seconds for the CBC mode, 10.84 seconds for the

CFB mode, and 10.30 seconds for the OFB Mode. In this case it could be observed that the time cost is much increased when compared to the previous results. The time required to transfer a data length of 10Mbit is over 20 times more than that required for a data length of 1Mbit, so additional factors (beyond data length) should be considered. One possible answer is the flow control of the device. In communication, flow control is used to manage the rate of transmission between two nodes. When a package is lost, the rate of transmission may be poorly controlled, thereby requiring extra time to transfer the data. Another possible answer is the computation capabilities of the device. To manage a data length of 10Mbit in one communication requires that a device has extra memory, which may add to the burdens on that device or may be something that the device does not have available.

One can observe the differences existent between the direct transfer of data and the choice to transfer data by using UDTLEBC as protection device becomes gradually reduced with the gradual increase of the data length. While transferring a data length of 1Kbit, the time costs of transferring this data through the use of UDTLEBC triples the cost of transferring the data directly.By increasing this data length to 10Mbit, the ratio then decreases to less than 1.5.

One reasonable answer to this factor is that the ratio of the time spent on the encryption to that of the time spent on the data transfer has decreased. More precisely, the time costs for transferring the data directly can be divided into two categories, i.e. (1) the time to establish the communication and (2) the time to transfer the data. The time required to establish the communication is similar in all instances, but the time necessary to transfer the data increases with the increase of the data length. On the other hand, the

time costs of transferring the data by UDTLEBC as a protection device can be divided into three categories, i.e. (1) the time to establish the communication, (2) the time to transfer the data, and (3) the time to encrypt and decrypt the data. The time to establish the communication is also a stable element, but the times required to transfer, encrypt, and decrypt the data will be increased with the increase of data length. If we set the data length to be x, the average time to establish a communication to be c, the time cost to transfer data to be $f(x)$, and the time to encrypt and decrypt data to be $g(x)$, the ratio r of transferring data by using UDTLEBC as protection to transferring data directly will be:

$$r = \frac{c + f(x) + g(x)}{c + f(x)}$$

Since r is decreased with the increase of x, it means that $f(x)$ increases faster than $g(x)$, and it means that with the increase of the data length, on the ratio perspective, the influence of using UDTLEBC as a protection will be necessarily decreased.

In our experiments we have tested data length up to 10Mbit, which is a little higher than that required to transfer a file the size of 1Mbyte. Thus, in order to transfer any files which are less than 1Mbyte, we may expect a reduction in at least 50% of the extra time if UDTLEBC is used to protect the data transfer. At last, with good network conditions or through the use of a more powerful device, for any file of more than 1Mbyte, this ratio is expected to decrease further.

CHAPTER 7. CONCLUSION

Results had proven that a provably-secure and efficient new time-bound hierarchical key management scheme with the capability of user differentiation could be built to circumvent the problems of privacy and costs that the current time-bound hierarchical key management schemes faced in the BYOD environment. The proof had been accomplished by constructing the new scheme: the User-Differentiated Two-Layer Encryption-Based Scheme (UDTLEBC) and by proving its capability for user differentiation and security of key indistinguishability and against key recovery with respect to both static and adaptive adversaries. With its capability of user differentiation, it had been shown that when changes were required to be done for one user, no other users would be affected and all processes would be completed on the server's side without interactions with users so that a personal devices used in a BYOD environment would no longer cause potential privacy and cost issues.

The efficiency of the UDTLEBC had been evaluated by comparing its performance with that of a prototype, the time-bound hierarchical key management scheme Two-Layer Encryption-Based Scheme (TLEBC). It had been shown that, in the perspectives of the complexity of key updates and the complexity of the hierarchical change, the UDTLEBC performed better than does the TLEBC because it did not require interactions with users to proceed with the changes, and its performance in these two perspectives were believed

acceptable. In the private storage requirement, UDTLEBC is noted to perform as well as TLEBC and required only one piece of private storage. In the perspectives of public storage requirement, UDTLEBC also performed as well as TLEBC, but neither UDTLEBC nor TLEBC performed particularly well in this perspective because they both required a great deal of public storage. The public storage was not believed to be providing a great influence in a BYOD environment by the author of this thesis, so this deficiency was considered bearable. Improvements were required in future work. In the perspective of computation overhead, UDTLEBC required as twice as much time on average as did the TLEBC to do the key derivation, but it was still very efficient in terms of usage because it only required an additional 20-30 milliseconds to proceed with the key derivation. According to the results gathered in this thesis, the UDTLEBC was believed to be both capable and efficient for usage within a BYOD environment.

This thesis had further tested whether we could expand the usage of UDTLEBC into data transfer protection. These results indicated that for data lengths less than 10Mbits, transferring data by using UDTLEBC as protection device required at least 1.5 to 3 times the time required to transfer the data directly. Under good network conditions or through use of a more powerful device, for any file larger than 1Mbyte, this ratio was expected to decrease.

REFERENCES

REFERENCES

[1] *Pub, N. F. 197: Advanced Encryption Standard (AES) (2001),* Federal Information Processing Standards Publication 197, US Department of Commerce/NIST, November 26, 2001. Available from the NIST website.

[2] J. Loucks, R. Medcalf, L. Buckalew, and F. Faria. (2013) *The Financial Impact of BYOD. A Model of BYOD's Benefits to Global Companies* [online]. Available: http://www.webtorials.com/content/2013/06/the-financial-impact-of-byod-a-model-of-byods-benefits-to-global-companies.html

[3] S. G. Akl and P. D. Taylor, "Cryptographic solution to a problem of access control in a hierarchy," ACM Transactions on Computer Systems (TOCS), vol. 1, pp. 239-248, 1983.

[4] Network World. (2011). *BYOD Deluge: Network Access Control and Mobile Device Management Tools Work Together To Create A Blueprint For BYOD Success* [online]. Available: http://infosightsol.com/wordpress/wp-content/uploads/2013/03/Managing_the_BYOD_Deluge_with_a_BYOD_Blueprint.pdf

[5] Wikipedia. *Block cipher mode of operation* [online]. Available: http://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

[6] Wikipedia. *Partially ordered set* [online]. Available: http://en.wikipedia.org/wiki/Partially_ordered_set#cite_note-1

[7] H. Liaw, S. Wang, and C. Lei, "A dynamic cryptographic key assignment scheme in a tree structure," Computers & Mathematics with Applications, vol. 25, pp. 109-114, 1993.

[8] Wikipedia. *Symmetric-key algorithm* [online]. Available: http://en.wikipedia.org/wiki/Symmetric_encryption

[9] W.-G. Tzeng, "A time-bound cryptographic key assignment scheme for access control in a hierarchy," Knowledge and Data Engineering, IEEE Transactions on, vol. 14, pp. 182-188, 2002.

[10] C. Chang and D. Buehrer, "Access control in a hierarchy using a one-way trap door function," Computers & Mathematics with Applications, vol. 26, pp. 71-76, 1993

[11] I. Ray, I. Ray, and N. Narasimhamurthi, "A cryptographic solution to implement access control in a hierarchy and more," in Proceedings of the seventh ACM symposium on Access control models and technologies, 2002, pp. 65-73.

[12] G. Ateniese, A. De Santis, A. L. Ferrara, and B. Masucci, "Provably-secure time-bound hierarchical key assignment schemes," Journal of cryptology, vol. 25, pp. 243-270, 2012.

[13] E. Bertino, N. Shang, and S. S. Wagstaff, "An efficient time-bound hierarchical key management scheme for secure broadcasting," Dependable and Secure Computing, IEEE Transactions on, vol. 5, pp. 65-70, 2008.

[14] C.-M. Chen, T.-Y. Wu, B.-Z. He, and H.-M. Sun, "An efficient time-bound hierarchical key management scheme without tamper-resistant devices," in Computing, Measurement, Control and Sensor Network (CMCSN), 2012 International Conference on, 2012, pp. 285-288.

[15] H.-Y. Chen, "Efficient time-bound hierarchical key assignment scheme," Knowledge and Data Engineering, IEEE Transactions on, vol. 16, pp. 1301-1304, 2004.

[16] S.-Y. Wang and C.-S. Laih, "Merging: an efficient solution for a time-bound hierarchical key assignment scheme," Dependable and Secure Computing, IEEE Transactions on, vol. 3, pp. 91-100, 2006.

[17] C. Rose, "BYOD: An Examination Of Bring Your Own Device In Business," Review of Business Information Systems (RBIS), vol. 17, pp. 65-70, 2013.

[18] M. B. Walker. (2013, May 08). *DoD okays Blackberry 10, Samsung Knox for Android* [online]. Available: http://www.fiercemobilegovernment.com/story/dod-okays-blackberry-10-samsung-knox-Android/2013-05-08

[19] M. B. Walker. (2013, May 21). *DoD clears iOS 6 for use* [online]. Available: http://www.fiercemobilegovernment.com/story/dod-clears-ios-6-use/2013-05-21

[20] Avotus. The Cost of Lost Smartphones and Stolen Tablets: Beware of BYOD[online]. Available: http://www.avotus.com/blog-bring-your-own-device-byod.asp

[21] Advanced Network System. (2012). *BYOD Best Practice, Requirements and Challenges*[online]. Available: http://www.getadvanced.net/byod-more-info

[22] S. J. MacKinnon, P. D. Taylor, H. Meijer, and S. G. Akl, "An Optimal Algorithm for Assigning Cryptographic," IEEE Transactions on computers, vol. 100, 1985.

[23] L. Harn and H.-Y. Lin, "A cryptographic key generation scheme for multilevel data security," Computers & Security, vol. 9, pp. 539-546, 1990.

[24] G. C. Chick and S. E. Tavares, "Flexible access control with master keys," in Advances in Cryptology—CRYPTO'89 Proceedings, 1990, pp. 316-322.

[25] K. Ohta, T. Okamoto, and K. Koyama, "Membership authentication for hierarchical multigroups using the extended fiat-shamir scheme," in Advances in Cryptology—EUROCRYPT'90, 1991, pp. 446-457.

[26] I.-C. Lin, M.-S. Hwang, and C.-C. Chang, "A new key assignment scheme for enforcing complicated access control policies in hierarchy," Future Generation Computer Systems, vol. 19, pp. 457-462, 2003.

[27] M.-S. Hwang and W.-P. Yang, "Controlling access in large partially ordered hierarchies using cryptographic keys," Journal of Systems and Software, vol. 67, pp. 99-107, 2003.

[28] R. S. Sandhu, "Cryptographic implementation of a tree hierarchy for access control," Information Processing Letters, vol. 27, pp. 95-98, 1988.

[29] C.-C. Chang, R.-J. Hwang, and T.-C. Wu, "Cryptographic key assignment scheme for access control in a hierarchy," Information Systems, vol. 17, pp. 243-247, 5// 1992.

[30] M.-S. Hwang, "A new dynamic key generation scheme for access control in a hierarchy," Nordic Journal of Computing, vol. 6, pp. 363-371, 1999.

[31] S. Zhong, "A practical key management scheme for access control in a user hierarchy," Computers & Security, vol. 21, pp. 750-759, 2002.

[32] V. R. Shen and T.-S. Chen, "A novel key management scheme based on discrete logarithms and polynomial interpolations," Computers & Security, vol. 21, pp. 164-171, 2002.

[33] M. L. Das, A. Saxena, V. P. Gulati, and D. B. Phatak, "Hierarchical key management scheme using polynomial interpolation," ACM SIGOPS Operating Systems Review, vol. 39, pp. 40-47, 2005.

[34] S.-F. Tzeng, C.-C. Lee, and T.-C. Lin, "A Novel Key Management Scheme for Dynamic Access Control in a Hierarchy," IJ Network Security, vol. 12, pp. 178-180, 2011.

[35] Y. Sun and K. R. Liu, "Scalable hierarchical access control in secure group communications," in INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies, 2004, pp. 1296-1306.

[36] T. Hui-Min and C. Chin-Chen, "A cryptographic implementation for dynamic access control in a user hierarchy," Computers & Security, vol. 14, pp. 159-166, 1995.

[37] F. Kuo, V. R. Shen, T.-S. Chen, and F. Lai, "Cryptographic key assignment scheme for dynamic access control in a user hierarchy," IEE Proceedings-Computers and Digital Techniques, vol. 146, pp. 235-240, 1999.

[38] M.-S. Hwang, "An improvement of novel cryptographic key assignment scheme for dynamic access control in a hierarchy," IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol. 82, pp. 548-550, 1999.

[39] T.-S. Chen and Y.-F. Chung, "Hierarchical access control based on Chinese remainder theorem and symmetric algorithm," Computers & Security, vol. 21, pp. 565-570, 2002.

[40] J.-C. Birget, X. Zou, G. Noubir, and B. Ramamurthy, "Hierarchy-based access control in distributed environments," in Communications, 2001. ICC 2001. IEEE International Conference on, 2001, pp. 229-233.

[41] Q. Zhang and Y. Wang, "A centralized key management scheme for hierarchical access control," in Global Telecommunications Conference, 2004. GLOBECOM'04. IEEE, 2004, pp. 2067-2071.

[42] C.-H. Lin, "Hierarchical key assignment without public-key cryptography," Computers & Security, vol. 20, pp. 612-619, 2001.

[43] H.-Y. Chien and J.-K. Jan, "New hierarchical assignment without public key cryptography," Computers & Security, vol. 22, pp. 523-526, 2003.

[44] A. L. Ferrara and B. Masucci, "An information-theoretic approach to the access control problem," in Theoretical Computer Science, ed: Springer, 2003, pp. 342-354.

[45] X. Yi, "Security of Chien's efficient time-bound hierarchical key assignment scheme," IEEE Transactions on Knowledge and Data Engineering, vol. 17, pp. 1298-1299, 2005.

[46] X. Yi, "Security of Chien's efficient time-bound hierarchical key assignment scheme," IEEE Transactions on Knowledge and Data Engineering, vol. 17, pp. 1298-1299, 2005.

[47] R. D. Luce, Response Times: Their Role in Inferring Elementary Mental Organization3: Oxford University Press, 1986.

[48] Robert J. Kosinski (Sep 2013). *A Literature Review on Reaction Time.* Available: http://biae.clemson.edu/bpc/bp/lab/110/reaction.htm

[49] H.-M. Sun, K.-H. Wang, and C.-M. Chen, "On the security of an efficient time-bound hierarchical key management scheme," Dependable and Secure Computing, IEEE Transactions on, vol. 6, pp. 159-160, 2009.

[50] Aberdeen Group. (2011). *Prepare Your WLAN for the BYOD Invasion* [online]. Available: http://research.aberdeen.com/internetcontent/somoclo/0161-7262-AI-WLAN-BYOD-AB-08.pdf

[51] R. Ballagas, M. Rohs, J. G. Sheridan, and J. Borchers, "Byod: Bring your own device," in Proceedings of the Workshop on Ubiquitous Display Environments, Ubicomp, 2004.

[52] C.-C. Chang, I.-C. Lin, H.-M. Tsai, and H.-H. Wang, "A key assignment scheme for controlling access in partially ordered user hierarchies," in Advanced Information Networking and Applications, 2004. AINA 2004. 18th International Conference on, 2004, pp. 376-379.

[53] J.-S. Chou, C.-H. Lin, and T.-Y. Lee, "A novel hierarchical key management scheme based on quadratic residues," in Parallel and Distributed Processing and Applications, ed: Springer, 2005, pp. 858-865.

[54] Cisco. (2012). Cisco Study: *IT saying Yes to BYOD* [online]. Available: http://newsroom.cisco.com/release/854754/Cisco-Study-IT-Saying-Yes-To-BYOD

[55] Cloud Security Alliance. (2013) *How secure is mobile device management anyway* [online]? Available: https://blog.cloudsecurityalliance.org/2013/04/25/how-secure-is-mobile-device-management-anyway/

[56] J. Crampton, K. Martin, and P. Wild, "On key assignment for hierarchical access control," in Computer Security Foundations Workshop, 2006. 19th IEEE, 2006, pp. 14 pp.-111.

[57] A. De Santis, A. L. Ferrara, and B. Masucci, "Cryptographic key assignment schemes for any access control policy," Information Processing Letters, vol. 92, pp. 199-205, 2004.

[58] M. He, P. Fan, F. Kaderali, and D. Yuan, "Access key distribution scheme for level-based hierarchy," in Parallel and Distributed Computing, Applications and Technologies, 2003. PDCAT'2003. Proceedings of the Fourth International Conference on, 2003, pp. 942-945.

[59] H.-F. Huang and C.-C. Chang, "A new cryptographic key assignment scheme with time-constraint access control in a hierarchy," Computer Standards & Interfaces, vol. 26, pp. 159-166, 2004.

[60] M. Iron, "Building “Bring-Your-Own-Device”(BYOD) Strategies," BYOD Strategies, pp. 1-8, 2011.

[61] A. Joch. (2012). *BYOD: A Cost Saver or a Curse?*[online] Available: http://www.business2community.com/tech-gadgets/byod-a-cost-saver-or-a-curse-0166377

[62] T. Kaneshige. (2012). *BYOD - Five hidden costs to a bring-your-own-device progamme* [online]. Available: http://www.computerworlduk.com/in-depth/mobilewireless/3349518/byod--five-hidden-costs-to-a-bring-your-own-device-progamme/

[63] S. M. Kerner (2013). *Cisco Reduces Support costs with BYOD* [online]. Available: http://www.enterprisenetworkingplanet.com/netsysm/cisco-saves-support-costs-with-byod.html

[64] J. Lee, Y. Lee, and S.-C. Kim, "A White-List Based Security Architecture (WLSA) for the Safe Mobile Office in the BYOD Era," in Grid and Pervasive Computing, ed: Springer, 2013, pp. 860-865.

[65] D. McCafferty (2013). *With BYOD, Enterprise Matches Device to Employee*[online]. Available: http://www.cioinsight.com/it-news-trends/slideshows/with-byod-enterprise-matches-device-to-employee-10/

[66] R. S. Sandhu, "On some cryptographic solutions for access control in a tree hierarchy," in Proceedings of the 1987 Fall Joint Computer Conference on Exploring technology: today and tomorrow, 1987, pp. 405-410.

[67] U.S. Department of Defense. (2013). *Officials Approve Guide for Government-issued iOS 6 Devices*[online]. Available: http://www.defense.gov/news/newsarticle.aspx?id=120073

[68] J. Wu and R. Wei, "An access control scheme for partially ordered set hierarchy with provable security," in Selected Areas in Cryptography, 2006, pp. 221-232.

[69] A. Zych, J. Doumen, P. Hartel, and W. Jonker, "A Diffie-Hellman based key management scheme for hierarchical access control," 2005.

APPENDICES

Appendix A Definitions

All below definitions are intercepted from the work of Ateniese et al.[11]:

Definition 1. A time-bound hierarchical key management scheme for Γ is a pair of algorithms (Gen, Der) satisfying the following conditions:

1. The information generation algorithm Gen is probabilistic polynomial time. It takes as inputs the security parameter 1^r, a graph $G = (V, E)$ in Γ, and the interval-set P over a sequence of distinct time periods T, and produces as outputs:

a) A private information $s_{u,\lambda}$, for any class $u \in V$ and any time sequence $\lambda \in \mathrm{P}$;

b) A key $k_{u,t}$, for any class $u \in V$ and any time period $t \in T$;

c) A public information pub.

We denote by (s, k, pub) the output of the algorithm Gen where s and k denote the sequence of private information and of keys, respectively.

2. The key derivation algorithm Der is deterministic polynomial time. It takes as inputs the security parameter 1^r, a graph $G = (V, E)$ in Γ, and the interval-set P over a sequence of distinct time periods T, two classes u and v such that $v \in A_u$, a time sequence $\lambda \in \mathrm{P}$, a private information $s_{u,\lambda}$ assigned to class u for the time sequence λ, a time

period $t \in \lambda$, and the public information pub, and produces as output the key $k_{v,t}$ assigned to the class v at time period t. We require that for each class $u \in V$, each class $v \in A_u$, each time sequence $\lambda \in \mathrm{P}$, each time period $t \in \lambda$, each private information $s_{u,\lambda}$, each key $k_{v,t}$, each public information pub which can be computed by Gen on inputs 1^τ, G, and P, it holds that

$$Der\left(1^\tau, G, \mathrm{P}, u, v, \lambda, s_{u,\lambda}, t, pub\right) = k_{v,t}$$

Definition 2. A symmetric encryption scheme is a triple $\Pi(\mathrm{K}, \mathrm{E}, D)$ of algorithms satisfying the following conditions:

1. The key-generation algorithm K is probabilistic polynomial time. It takes as input the security parameter 1^λ and produces as output a string key.

2. The encryption algorithm E is probabilistic polynomial time. It takes as inputs 1^λ, a string key produced by $\mathrm{K}\left(1^\lambda\right)$, and a message $m \in (0,1)^*$, and produces as output the ciphertext y.

3. The decryption algorithm D is deterministic polynomial time. It takes as inputs 1^λ, a string key produced by $\mathrm{K}\left(1^\lambda\right)$, and a ciphertext y, and produces as output a message m. We require that for any string key which can be output by $\mathrm{K}\left(1^\lambda\right)$, for any message $m \in (0,1)^*$, and for y that can be output by $\mathrm{E}\left(1^\tau, key, m\right)$, we have $D\left(1^\tau, key, y\right) = m$.

Definition 3 [IND-ST]. Let Γ be a family of graphs corresponding to partially ordered hierarchies, let $G=(V,E)\in\Gamma$ be a graph, let T be a sequence of distinct time periods, let P be the interval-set over T, and let (Gen, Der) be a time-bound hierarchical key assignment scheme for Γ. Let $STAT_{u,t}$ be a static adversary which attacks a class $u\in V$ in a time period $t\in T$. Consider the following two experiments:

Experiment $Exp^{IND-1}_{STAT_{u,t}}(1^\tau, G, \mathrm{P})$
$(s, k, pub)\leftarrow Gen(1^\tau, G, \mathrm{P})$
$corr\leftarrow Corrupt_{u,t}(s)$
$d\leftarrow STAT_{u,t}(1^\tau, G, \mathrm{P}, pub, corr, k_{u,t})$
return d

Experiment $Exp^{IND-0}_{STAT_{u,t}}(1^\tau, G, \mathrm{P})$
$(s, k, pub)\leftarrow Gen(1^\tau, G, \mathrm{P})$
$corr\leftarrow Corrupt_{u,t}(s)$
$\rho\leftarrow\{0,1\}^{length(k_{u,t})}$
$d\leftarrow STAT_{u,t}(1^\tau, G, \mathrm{P}, pub, corr, k_{u,t})$
return d

The advantage of $STAT_{u,t}$ is defined as

$$Adv^{IND}_{STAT_{u,t}}(1^\tau, G, \mathrm{P})=\left|\Pr\left[Exp^{IND-1}_{STAT_{u,t}}(1^\tau, G, \mathrm{P})=1\right]-\Pr\left[Exp^{IND-0}_{STAT_{u,t}}(1^\tau, G, \mathrm{P})=1\right]\right.$$

The scheme is said to be secure in the sense of IND-ST (key indistinguishability against a static adversary whose time complexity is polynomial in τ) if the function $Adv^{IND}_{STAT_{u,t}}(1^\tau, G, \mathrm{P})$ is negligible

Definition 4 [IND-AD]. Let Γ be a family of graphs corresponding to partially ordered hierarchies, let $G=(V,E)\in\Gamma$ be a graph, let T be a sequence of distinct time periods, let P be the interval-set over T, and let (Gen, Der) be a time-bound hierarchical key assignment scheme for Γ. Let $ADAPT=(ADAPT_1, ADAPT_2)$ be an adaptive adversary

that is given access to the oracle $\mathrm{O}_s(.)$ during both stages of the attack, where s is the private information computed by Gen. Consider the following two experiments:

Experiment $Exp^{IND-1}_{ADAPT_{u,t}}\left(1^\tau, G, \mathrm{P}\right)$
$(s, k, pub) \leftarrow Gen\left(1^\tau, G, \mathrm{P}\right)$
$(u, t, state) \leftarrow ADAPT_1^{\mathrm{O}_s(.)}\left(1^\tau, G, \mathrm{P}, pub\right)$
$d \leftarrow ADAPT_2^{\mathrm{O}_s(.)}\left(1^\tau, G, \mathrm{P}, pub, u, t, state, k_{u,t}\right)$
return d

Experiment $Exp^{IND-0}_{ADAPT_{u,t}}\left(1^\tau, G, \mathrm{P}\right)$
$(s, k, pub) \leftarrow Gen\left(1^\tau, G, \mathrm{P}\right)$
$(u, t, state) \leftarrow ADAPT_1^{\mathrm{O}_s(.)}\left(1^\tau, G, \mathrm{P}, pub\right)$
$\rho \leftarrow \{0,1\}^{length\left(k_{u,t}\right)}$
$d \leftarrow ADAPT_2^{\mathrm{O}_s(.)}\left(1^\tau, G, \mathrm{P}, pub, u, t, state, \rho\right)$
return d

$O_s(.)$ is a oracle which can provide the adversary knowledge associated with a pair (u,t). It is required that the $ADAPT_1$ can only output pair (u,t) belongs to $F_{u.t}$ and $ADAPT_2$ cannot query a pair (v,λ) such that $u \in A_v$ and $t \in \lambda$. The advantage of the scheme is defined as

$$Adv^{IND}_{ADAPT}\left(1^\tau, G, \mathrm{P}\right) = \left|\Pr\left[Exp^{IND-1}_{ADAPT}\left(1^\tau, G, \mathrm{P}\right) = 1\right] - \Pr\left[Exp^{IND-0}_{ADAPT}\left(1^\tau, G, \mathrm{P}\right) = 1\right]\right|$$

The scheme is said to be secure in the sense of IND-AD (key indistinguishability against an adaptive adversary whose time complexity is polynomial in τ) if the function $Adv^{IND}_{ADAPT}\left(1^\tau, G, \mathrm{P}\right)$ is negligible.

Definition 5 [REC-ST]. Let Γ be a family of graphs corresponding to partially ordered hierarchies, let $G = (V, E) \in \Gamma$ be a graph, let T be a sequence of distinct time periods, let P be the interval-set over T, and let (Gen, Der) be a time-bound hierarchical key assignment scheme for Γ. Let $STAT_{u,t}$ be a static adversary which attacks a class $u \in V$ in a time period $t \in T$. Consider the following two experiments:

Experiment $Exp^{REC}_{STAT_{u,t}}\left(1^{\tau}, G, \mathrm{P}\right)$
$\left(s, k, pub\right) \leftarrow Gen\left(1^{\tau}, G, \mathrm{P}\right)$
$corr \leftarrow Corrupt_{u,t}\left(s\right)$
$k^{'}_{u,t} \leftarrow STAT_{u,t}\left(1^{\tau}, G, \mathrm{P}, pub, corr\right)$
return $k^{'}_{u,t}$

The advantage of $STAT_{u,t}$ is defined as

$$Adv^{REC}_{STAT_{u,t}}\left(1^{\tau}, G, P\right) = \Pr\left[k^{'}_{u,t} = k_{u,t}\right]$$

The scheme is said to be secure in the sense of REC-ST (against key recovery with respect to a static adversary whose time complexity is polynomial in τ) if the function $Adv^{REC}_{STAT_{u,t}}\left(1^{\tau}, G, P\right)$ is negligible.

Definition 6 [REC-AD]. Let Γ be a family of graphs corresponding to partially ordered hierarchies, let $G = \left(V, E\right) \in \Gamma$ be a graph, let T be a sequence of distinct time periods, let P be the interval-set over T, and let $\left(Gen, Der\right)$ be a time-bound hierarchical key assignment scheme for Γ. Let $ADAPT = \left(ADAPT_1, ADAPT_2\right)$ be an adaptive adversary that is given access to the oracle $\mathrm{O}_s\left(.\right)$ during both stages of the attack, where s is the private information computed by Gen. Consider the following two experiments:

Experiment $Exp^{IND-1}_{ADAPT_{u,t}}\left(1^{\tau}, G, \mathrm{P}\right)$
$\left(s, k, pub\right) \leftarrow Gen\left(1^{\tau}, G, \mathrm{P}\right)$
$(u, t, state) \leftarrow ADAPT_1^{\mathrm{O}_s(.)}\left(1^{\tau}, G, \mathrm{P}, pub\right)$
$k^{'}_{u,t} \leftarrow ADAPT_2^{\mathrm{O}_s(.)}\left(1^{\tau}, G, \mathrm{P}, pub, u, t, state\right)$
return $k^{'}_{u,t}$

It is required that the $ADAPT_1$ can only output pair (u,t) belongs to $F_{u.t}$ and $ADAPT_2$ cannot query a pair (v,λ) such that $u \in A_v$ and $t \in \lambda$. The advantage of the scheme is defined as

$$Adv_{ADAPT}^{REC}(1^{\tau}, G, P) = \Pr[k'_{u,t} = k_{u,t}]$$

The scheme is said to be secure in the sense of REC-AD (against key recovery with respect to an adaptive adversary whose time complexity is polynomial in τ) if the function $Adv_{ADAPT}^{REC}(1^{\tau}, G, P)$ is negligible.

Appendix B Theorems

All below theorems are intercepted from the work of Ateniese et al.[11]:

Theorem 1 [IND-ST $\Leftrightarrow$ IND-AD] Let Γ be a family of graphs corresponding to partially ordered hierarchies. A time-bound hierarchical key assignment scheme for Γ is secure in the sense of IND-ST if and only if it is secure in the sense of IND-AD.

Theorem 2 [REC-ST $\Leftrightarrow$ REC-AD] Let Γ be a family of graphs corresponding to partially ordered hierarchies. A time-bound hierarchical key assignment scheme for a family of graphs Γ is secure in the sense of REC-ST if and only if it is secure in the sense of REC-AD.

Theorem 3 [IND-ST $\Rightarrow$ REC-ST] Let Γ be a family of graphs corresponding to partially ordered hierarchies. If a time-bound hierarchical key assignment scheme for Γ is secure in the sense of IND-ST, then it is secure in the sense of REC-ST.

Theorem 4 [REC-ST$\neq$>IND-ST] Let Γ be a family of graphs corresponding to partially ordered hierarchies. If there exists a time-bound hierarchical key assignment scheme for Γ which is secure in the sense of REC-ST, there exists a time-bound hierarchical key assignment scheme for Γ which is secure in the sense of REC-ST but which is not secure in the sense of IND-ST.