Università degli Studi di Salerno
Dipartimento di Informatica

DOTTORATO DI RICERCA IN INFORMATICA
CICLO XIV - NUOVA SERIE

Tesi di Dottorato in Informatica

New Insights on Cryptographic Hierarchical Access Control: Models, Schemes and Analysis

Candidate
Arcangelo Castiglione

Tutor
Prof. Alfredo De Santis

Co-Tutor
Prof. Barbara Masucci

Coordinator
Prof. Gennaro Costagliola

2014/2015

To my family

Acknowledgements

During my PhD I had the great pleasure to work with wonderful and professional people and I wish to express my gratitude to everybody of them, for the fundamental research experience they have granted me.

First and foremost, I would like to thank Prof. Alfredo De Santis, my wise tutor, for his help, his many precious suggestions and his constant encouragement during my PhD. Alfredo has served as a role model for me and I am sure that he will continue to be a source of inspiration. I have learnt many things from Alfredo; both technically and also regarding ones attitude and approach to research. I hope that I will have many opportunities to continue working with Alfredo in the future.

I would like also to express my sincere gratitude to my co-tutor, Prof. Barbara Masucci for her collaboration, her patience and her constructive critical opinions. Again, I would like to thank her for fruitful joint work and many helpful discussions. It was always a pleasure to work with Barbara, since her demand for excellence on the one hand and positive encouragement on the other, served as great motivating factors throughout the ups and downs of the last years. My work with Barbara began in the second year of my studies and her contribution to me on both a personal and research level has been enormous.

I wish also to express my sincere gratitude to Prof. Francesco Palmieri. He has been a true mentor for me, an essential and constant point of reference. I have had the pleasure of working with him and this has been an invaluable experience for me.

Moreover, I would like to thank Prof. Marek R. Ogiela for his hospitality at the Laboratory of Cryptography and Cognitive Informatics, AGH University of Science and Technology in Krakow, Poland.

My deepest thanks goes to my closest collaborator and good friend Raffaele Pizzolante for the many discussions that we had, ranging from "consumer electronics" to technical issues in our research. Furthermore, I would like to thank my long-time friends Ciriaco D'Ambrosio, Pietro Albano and Andrea Bruno for their unconditional friendship and their constant support.

Many thanks to my other co-authors: Prof. Bruno Carpentieri, Prof. Paolo D'Arco, Prof. Xinyi Huang and Prof. Jin Li. I enjoyed working with them all and learnt much from each of them.

Finally, I would like to thank my family, for the support and constant presence.

Abstract

Nowadays the current network-centric world has given rise to several security concerns regarding the access control management, which ensures that only authorized users are given access to certain resources or tasks. In particular, according to their respective roles and responsibilities, users are typically organized into *hierarchies* composed of several disjoint classes (*security classes*). A hierarchy is characterized by the fact that some users may have more access rights than others, according to a top-down inclusion paradigm following specific hierarchical dependencies. A user with access rights for a given class is granted access to objects stored in that class, as well as to all the descendant ones in the hierarchy. The problem of *key management* for such hierarchies consists in assigning a key to each class of the hierarchy, so that the keys for descendant classes can be efficiently obtained from users belonging to classes at a higher level in the hierarchy.

In this thesis we analyze the security of hierarchical key assignment schemes according to different notions: security with respect to *key indistinguishability* and against *key recovery* [4], as well as the two recently proposed notions of security with respect to *strong key indistinguishability* and against *strong key recovery* [42]. More precisely, we first explore the relations between all security notions and, in particular, we prove that security with respect to strong key indistinguishability is *not stronger* than the one with respect to key indistinguishability. Afterwards, we propose a general construction yielding a hierarchical key assignment scheme that ensures security against strong key recovery, given any hierarchical key assignment scheme which guarantees security against key recovery.

Moreover, we define the concept of *hierarchical key assignment schemes supporting dynamic updates*, formalizing the relative security model. In particular, we provide the notions of security with respect to *key indistinguishability* and *key recovery*, by taking into account the dynamic changes to the hierarchy. Furthermore, we show how to construct a hierarchical key assignment scheme supporting dynamic updates, by using as a building block a symmetric encryption scheme. The proposed construction is provably secure with respect to key indistinguishability, provides efficient key derivation and updating procedures, while requiring each user to store only a single private key.

Finally, we propose a novel model that generalizes the conventional hierarchical access control paradigm, by extending it to certain additional sets of qualified users. Afterwards, we propose two constructions for hierarchical key assignment schemes in this new model, which are provably secure with respect to key indistinguishability. In particular, the former construction relies on both symmetric encryption and perfect secret sharing, whereas, the latter is based on public-key threshold broadcast encryption.

Contents

List of Figures

LIST OF FIGURES

Chapter 1

Introduction

"Those who are enamored of practice without theory are like a pilot who goes into a ship without rudder or compass and never has any certainty where he is going. Practice should always be based upon a sound knowledge of theory."

— Leonardo Da Vinci, *1452-1519*

1.1 Introduction

Nowadays the current network-centric world has given rise to several security issues concerning the *access control management*, which ensures that only authorized users are given access to certain resources or tasks. In particular, the Internet has been created to design a resource network where users interact seamlessly and share information without needing to worry about the location of the information or the path over which it is sent. Again, user interactivity and information sharing on the Internet cause issues concerning data privacy and performance in terms of accessibility. Those issues arise from the fact that it is difficult to forecast and hard-code solutions to all the possible scenarios that an application has to face during its execution, therefore, guaranteeing data privacy under changing conditions is challenging. For example, as organizations increase their dependence on network-based information systems for daily business, such

organizations become more and more exposed to security threats, even though they improve their efficiency and productivity. Although several techniques, such as encryption and digital signatures have been proposed in the literature to protect data, a holistic approach for ensuring data security should be taken into account. Such an approach should enforce access control policies based on data contents, user qualifications, and other relevant contextual information. Consequently, when dealing with the security of an information system, a large number of concepts must be taken into account and several questions should be answered: should I use cryptography? how do I generate keys? how do I change keys if I think my system has been breached? do I have to re-encrypt all my data every time I generate a new key for one class of data? do I have to be constantly monitoring the system and intervening to update keys?

The *access control problem* deals with the ability to ensure that only authorized users of a system are given access to some sensitive resources or tasks. In general, users belonging to a given organization are grouped according to their competencies and responsibilities in a certain number of disjoint classes, referred to as *security classes*. The basic assumption is that the keys will be assigned to groups, so that when the security policy provides a group with the access to some data, the key can be used for decrypting the data that can be accessed by the group. Of course, if the security policy changes frequently, group memberships may change, requiring both new keys and re-encrypting the data accessible by the group. Therefore, data and resources that can be accessed by a user are based on its relative membership class. For example, in the healthcare scenario, doctors can access data concerning their patients, such as diagnosis, medication prescriptions, and laboratory tests, whereas, researchers can be limited to only consult anonymous clinical information for studies.

The set of rules which characterizes the information flow between different security classes in the system defines an *access control policy*. More precisely, for any class in the system, the access control policy specifies the set of classes which can be accessed by that class. Such a set is denoted as the *accessible set* of the class. It is important to remark that according to their responsibilities and rights, users can have overlapping permission accesses; indeed, users belonging to different classes, might need to access some common data and resources. In detail,

a permission allows a user to perform several well-defined operations. Permissions are characterized by a hierarchy of access rights, hence, users are assigned roles that define which permissions they can exercise and in what context. A role is a set of operations that a user is allowed to perform on data and resources. Notice that a user can have more than one role. In general, hierarchies can be created to ensure access control according to the inherent structure of an organization. It is easy to note that *hierarchical access control policies* are a natural way of organizing users to reflect reflect their authority, responsibility and competency. In fact, there are several scenarios where supervisors have the full privileges to control the tasks of their subordinates, while the subordinates have no privileges at all to access the supervisors' tasks. Similar situations are very common in many further scenarios, e.g., in the government, military and healthcare. Hierarchical access control policies find a natural way of application also in the business and in many other fields, such as the management of databases containing sensitive information or the protection of industrial secrets.

It is important to point out that sometimes the conventional hierarchical access control may be a limitation, since it could be necessary to provide some particular sets of users, having specific access credentials, with access to the key of a certain security class. This novel access control model finds a natural field of application even when there is the need to manage unusual, exceptional or emergency situations, which in general require special permissions. In particular, consider the case in which the trust is based on a single entity, let it be a person or an organization. Obviously, this may lead to abuses or violations by such entity, as in the Snowden event [69], where a great deal of confidential information held by the U.S. *National Security Agency* (*NSA*) was stolen. However, the NSA itself has defined in the past some strict guidelines for limiting such abuses, namely, the *Orange Book* [63] and *Two-Person Authorization* [17] [39], whose main goal was to prevent a single user from viewing top-secret documents. The concept upon which the guidelines are based is that, in general, somebody is less inclined to do something dishonest if someone else is watching. In addition, the two guidelines clearly state that the information within a system must be organized in a "*compartmental manner*", providing different levels of access and security to each compartment. In this case, a simple protection may be the use of two

or more "locks" to protect a given resource or activity, where each lock needs a different key, owned by a different person. Thus, two or more people are needed in order to grant the access to that resource or activity.

The Snowden event highlights the fact that the collaboration among several users and organizations is preferable for gaining the permission to carry out a given task or to access sensitive information. Such collaboration is needed so as to ensure that the requested permission has been granted through the acceptance and agreement among all the involved entities, thus preventing users from any kind of abuse. In general, the collaboration characterizes any scenario where more than one entity is required to achieve a specific authorization. More precisely, there are many real-world scenarios in which such a collaborative access is necessary, i.e., where a user might have a sort of "pre-authorization" for the access, but he may need to get the approval from someone else. For example, consider the healthcare environment, which typically consists of several professional profiles, such as doctors, nurses, etc.. In this environment, nurses may access a subset of stored patient's clinical data, while a doctor can usually access all the data. However, it is important to emphasize that the doctor and nurse must have the patient's consent to access clinical information. In addition, a nurse should not access all the information concerning a patient, unless she does not gain the permission from both patient and doctor. Moreover, if a doctor wants to access some clinical data without the explicit consent of the patient, he should be granted permission from several entities, e.g., hospital administration, medical committee, government authority, etc..

Again, the access to the workspace of a specific project branch could be granted either directly to the project manager or to a set of project team members. The same arguments apply to distributed cryptographic file systems [22]. A further real field of application lies in the collaborative access to logs concerning accesses and events, where the access can be achieved either by a single entity (e.g., a communications authority) or by more of them, which cooperate with each other. For example, the access might be allowed only if the judicial authority cooperates with a given service provider. Another concrete example arises from the military field, in which a decision can be taken by a single person with a specific rank, by a certain number of his subordinates or more generally, by a

given number of people with certain credentials, which do not have the authority to decide on their own. Furthermore, consider a committee board composed of several members and a general chair. In this context, the chair might be away for personal reasons or could be in a situation which prevents him from making any decisions for a given action. Only one member of the board cannot independently take such a decision on behalf of the chair. However, the board members can collectively take such a decision on behalf of the chair, as long as their number is greater than or equal to a certain threshold.

Finally, advances in wireless communication and electronics have given rise to the need of ensuring access control even in contexts characterized by high dynamicity. Thus, one of our goals has been to show how access control and cryptographic key assignment can be combined to ensure dynamic and fine-grained data security.

1.2 Cryptographic Hierarchical Access Control and Key Assignment Schemes

An *access control policy* defines the set of rules characterizing the information flow between different user classes in the system, and it can be represented by a directed graph $G = (V, E)$, where the vertex set V corresponds to the set of security classes and there is a directed edge $(u, v) \in E$ if and only if class u can access class v. Again, each class has a self-loop. For each $u \in V$, we define the *accessible set* of u as the set of classes that can be accessed by u. We also define the *incoming set* of u as the set of classes that can access class u. Based on certain characteristics, access control policies can be categorized as follows.

- **Poset based access control policy.** Many organizations are characterized by an inherent hierarchical structure and can be represented through a *partially ordered set (poset)*. More formally, a poset based access control policy has the three following properties:

  - *Reflexive*: A class u has access to its own data;

  - *Antisymmetric*: If a class u has access to the data of a class v and the class v can access the data of u, then u and v are equivalent classes;
  - *Transitive*: If a class u can access the data of a class v and the class v can access the data of a class z, then the class u is able to access the data of z.

- **Arbitrary access control policy.** There are many scenarios which cannot be characterized by a strict hierarchy, thus requiring more general access control policies. For example, an access control policy corresponding to these scenarios may violate the anti-symmetric and transitive properties of a partially ordered hierarchy. More formally, an arbitrary access control policy has the two following properties:
  - *Reflexive*: A class u has access to its own data;
  - *Equivalence*: If the sets of classes that u and v can access are the same, and the classes having access to the data of u are also authorized to access the data of v and viceversa, then u and v are equivalent classes.

In the field of cryptography, an access control policy can be implemented by means of a *key assignment scheme*, which is a method to assign an encryption key and some private information to each class. The encryption key will be used by each class to protect its own data, usually through a symmetric cryptosystem, whereas, the private information will be used by each class to compute the keys assigned to its accessible set. This assignment is carried out by a *trusted authority*, referred to as *TA*, which is active only during the distribution phase. Specifically, we consider how encryption keys can be used to reinforce the security of access control schemes.

A basic and trivial cryptographic key assignment scheme requires each class to store the encryption keys assigned to all classes in its accessible set. The main drawback of this solution is that users in high level classes need to handle more information than users in low level classes. It is easy to observe that low space requirements and best performances enable a scheme to be used much more extensively and in wider contexts than costly schemes. Therefore, an important measure concerning the efficiency of a key assignment scheme is the size of the

private information that each user stores to gain data accesses. Again, since the derivation of a key could be performed in real-time, by users which may have constrained hardware and software capabilities, it is important to design schemes where the number of operations required to compute the key of the classes lower down in the hierarchy is as small as possible. Obviously, also the operation complexity should be as efficient as possible.

Furthermore, to improve the deployability of a key assignment scheme, also access control policies which are more general than the hierarchical one should be considered. Finally, to increase the effectiveness of a key assignment scheme, it should be provided with the ability of managing dynamic access control policies. More precisely, in those schemes the topology of the access control policy is allowed to change dynamically, i.e., classes and relations may be added to or deleted from the system. As a consequence of any update, some private information associated to particular classes may need to be updated. However, due to efficiency reasons, it would be preferable that when any change takes place, the number of classes involved in the update is as small as possible.

1.3 State of the Art

Akl and Taylor [2] first addressed the problem of reducing the inherent complexity of the basic trivial key assignment scheme. In particular, they proposed an elegant solution to solve the access control problem in systems organized as a partially ordered hierarchy (poset). In their scheme, each class is assigned a key that can be used later on, along with some public parameters generated by the TA, to compute the key assigned to any class lower down in the hierarchy. Furthermore, the scheme due to Akl and Taylor is secure against *collusion attacks* performed by non-authorized classes. However, as the number of entities in the system grows, so does the size of the keys held by higher level classes and the number of computations required to derive lower level keys. In order to overcome the above defined limitation, Mackinnon et al. [60] proposed an algorithm aimed at determining an optimal assignment of keys by assigning the smallest primes to the longest chains in the hierarchy.

Subsequently, many schemes have been proposed, which either have better

performances or allow inserting and deleting classes in the hierarchy, as for example the ones proposed in [27], [46], [61], [49], [58], [57], [59], [60], [64].

For instance, Sandhu [64] considered a significant type of hierarchy, i.e., *a rooted tree hierarchy*. In his scheme, the key of a security class is generated through the class identity and its parents' secret key, by using a *one-way function*. In 1990, Harn and Lin [46] proposed an approach that, instead of using a top-down strategy as in the Akl-Taylor scheme, it uses a bottom-up key generation approach. By doing this, the space required to store the public parameters for security classes is much smaller than that required by the previous schemes. However, when there are many security classes in the system, a large amount of storage space is still required to store the public parameters. Subsequently, Chang et al. [27] proposed a scheme based on *Newton's interpolation method* and using one-way functions. The main drawback is that all of those schemes can be used to implement only poset-based access control policies.

The problem of designing cryptographic key assignment schemes for access control policies with transitive and anti-symmetrical exceptions was first considered by Yeh et al. [72]. However, Hwang [50] showed that their scheme was insecure against *collusion attacks* carried out by non-authorized classes.

It is important to point out that the above solutions deal with the access control problem when the keys are assigned to users for an indefinite period of time, and it is supposed that the only time in which it is necessary to re-assign keys is when a user joins or leaves the system. However, in practical scenarios, it is likely that users may belong to a class for a limited period of time. In 2002, Tzeng [70] proposed the first time-bound key assignment scheme to deal with this situation. His proposed solution assumes that each class has a different cryptographic key for each time period. The scheme is efficient in terms of space requirements, but the computation of a key requires expensive public-key and *Lucas computations* [56].

Afterwards, Bertino et al. [14] pointed out how key assignment schemes play a fundamental role to enforce secure broadcasting of XML documents. In particular, they showed how to enforce access control on XML documents. This is a very important feature for services where updates are sent to subscribers which have access permissions for different portions of the same document. The

hierarchical structure of an XML document allows to encrypt each different portions of a document just once. The idea is to generate a single encrypted copy of the document, where each different portion is encrypted by using a different cryptographic key. Bertino et al. [14] used the Tzeng's scheme to encrypt parts of an XML document and distribute appropriate decryption keys to authorized users. However, the Tzeng's scheme was shown to be insecure against *collusion attacks* in [73]. Then, a new time-bound hierarchical key assignment scheme was proposed by Huang and Chang [48], but Tang and Mitchell [68] analyzed the security of such a scheme and showed some vulnerabilities, which enable malicious users to breach the privacy of other users. Subsequently, Chien [28] proposed a time-bound key assignment scheme based on *tamper-resistant devices*. Again, Bertino et al. [66] proposed an efficient hierarchical key generation and key diffusion scheme for sensor networks. Finally, De Santis et al. [35] proposed a new key assignment scheme for access control in a *complete tree hierarchy*. However, all those schemes either require high computational load or lack of a formal security proof.

According to the *security reduction paradigm* introduced by Goldwasser and Micali [44], a scheme is *provably-secure* under a complexity assumption if the existence of an adversary A breaking the scheme implies the existence of an adversary B breaking the computational assumption [44]. Atallah et al. [4] first addressed the problem of formalizing security requirements for hierarchical key assignment schemes and proposed two different notions: security against *key recovery* and with respect to *key indistinguishability*. Informally speaking, the former captures the notion that an adversary should not be able to compute a key to which it should not have access, while in the latter, the adversary should not even be able to distinguish between the real key and a random string of the same length. In particular, the model considered in [4] allows an adversary attacking a certain class in the hierarchy to gain access to the private information assigned to all users not allowed to access such a class, as well as all the public information.

Atallah et al. [4] also proposed two provably-secure constructions for hierarchical key assignment schemes: the first one is based on pseudorandom functions and satisfies security against key recovery, whereas, the second one requires the

additional use of a symmetric encryption scheme and guarantees security with respect to key indistinguishability. Different constructions satisfying the above defined notions of security have been proposed in [8, 32, 6, 29, 30, 33, 38, 7, 41]. In particular, De Santis et al. [32, 33] proposed two different constructions satisfying security with respect to key indistinguishability: the first one, which is based on symmetric encryption schemes, is simpler than the one proposed in [4], requires a single computational assumption, and offers more efficient procedures for key derivation and key updates; the second one, which is based on a public-key broadcast encryption scheme, allows to obtain a hierarchical key assignment scheme offering constant private information and public information linear in the number of classes. D'Arco et al. [29, 30] analyzed the Akl-Taylor scheme according to the definitions proposed in [4] and showed how to choose the public parameters in order to get instances of the scheme which are secure against key recovery under the RSA assumption. Moreover, they showed how to turn the Akl-Taylor scheme in a construction offering security with respect to key indistinguishability; however, such a scheme is less efficient than the constructions proposed in [4, 32, 33]. Then, Freire et al. [41] proposed a construction based on factoring, satisfying security with respect to key indistinguishability. Again, Ateniese et al. [8, 7] and De Santis et al. [36] extended the model proposed in [4] to schemes satisfying additional time-dependent constraints and proposed two different constructions offering security with respect to key indistinguishability. Other constructions for time-dependent schemes, offering different trade-offs in terms of amount of public and private information and complexity of key derivation, were shown in [37, 6, 38, 15].

Recently, Freire et al. [42] proposed new security definitions for hierarchical key assignment schemes. Such definitions, called security against *strong key recovery* and security with respect to *strong key indistinguishability*, provide the adversary with additional compromise capability, thus representing a strengthening of the model provided in [4]. As stated by Freire et al., such a new model is able to characterize a variety of scenarios which may arise in real-world situations, since it allows the protection of the key assigned to a certain class u, even when the keys held by classes which are predecessors of u in the hierarchy have been leaked, due to their use, loss or theft. More precisely, Freire et al.

considered an adversary which, given a certain class, is allowed to achieve the private information assigned to all users not allowed to access such class, as well as all the public information and *keys assigned to all the other classes which are predecessors of the attacked (target) class in the hierarchy*. Freire et al. also proposed two hierarchical key assignment schemes which are secure in the sense of strong key indistinguishability. The former construction is based on *pseudorandom functions*, whereas, the latter one is based on *forward-secure pseudorandom generators*. Finally, they showed that the notions of security against key recovery and against strong key recovery *are separated*, i.e., there exist schemes that are secure against key recovery but which are not secure against strong key recovery. On the other hand, they did not clarify the relations between the notions of security with respect to key indistinguishability and with respect to strong key indistinguishability.

1.4 Contributions of This Thesis

In this thesis we provide new insights on cryptographic hierarchical access control and key assignment, in particular by focusing on models, schemes and security notions, as well as on their relative analysis. As stated before, cryptographic hierarchical access control is implemented by means of proper key assignment schemes, which involve secret and public information generated and distributed by the TA to the users, in order to set up the scheme. More precisely, in this thesis we focus on cryptographic hierarchical access control schemes which are based on *dependent key management* approaches, besides discussing the challenges involved for extending them to deal with changing conditions and dynamic environments. In general, we recall that dependent key management schemes organize users into groups and assign each group a unique key. Afterwards, such a key can be used either to decrypt data that the users are authorized to access or to derive the keys of their accessible set. We remark that assigning groups single keys makes security management easier for both the users and security administrator, since it reduces the risks of mismanagement that could lead to security violations. Obviously, it is important to construct schemes whose amount of public and secret information is as small as possible. The efficiency of a hierarchical key assign-

ment scheme is evaluated according to different parameters: *storage requirements*, which correspond to the amount of secret data that needs to be distributed and stored by the users and the amount of data that needs to be made public; the *complexity* of both *key derivation* and *key update* procedures. Indeed, it is preferable that updates to the access hierarchy, performed to reflect changes in access permissions, require only local changes to the public information and do not need any private information to be re-distributed; the *computational assumption* on which the security of the scheme relies. In fact, it is preferable to rely on *standard assumptions*.

In this thesis we analyze the security of hierarchical key assignment schemes according to different notions, that is, security with respect to *key indistinguishability* and against *key recovery*, as well as the two recently proposed notions of security with respect to *strong key indistinguishability* and against *strong key recovery*. We first explore the relations between all security notions and, in particular, we prove that security with respect to strong key indistinguishability is *not stronger* than the one with respect to key indistinguishability, thus answering an important open question in the field of hierarchical key assignment schemes. Afterwards, we propose a general construction yielding a hierarchical key assignment scheme which ensures security against strong key recovery, given any hierarchical key assignment scheme which guarantees security against key recovery.

Moreover, we define the concept of *hierarchical key assignment schemes supporting dynamic updates*, besides formalizing the relative security model. In particular, we provide the notions of security with respect to *key indistinguishability* and *key recovery*, by taking into account the dynamic changes to the hierarchy. Moreover, we show how to construct a hierarchical key assignment scheme supporting dynamic updates, by using as a building block a *symmetric encryption scheme*. The proposed construction is provably secure with respect to key indistinguishability, provides efficient key derivation and updating procedures, while requiring each user to store only a single private key.

Finally, we propose a novel model that generalizes the conventional hierarchical access control paradigm, by extending it to certain additional sets of qualified users. Then, we propose two constructions for hierarchical key assignment

schemes in this new model, which are provably secure with respect to key indistinguishability. In particular, the former construction relies on both *symmetric encryption* and *perfect secret sharing*, whereas, the latter is based *on public-key threshold broadcast encryption*. We remark that in this thesis the security of all the proposed hierarchical key assignment schemes rely on the computational infeasibility of breaking it (*computational security*).

The motivations underlying this thesis are mainly based on two observations. The first observation is that even though the simplicity of using password-based authentication schemes has made them the "de facto" standard to enforce access control, their vulnerability to even more sophisticated attacks has made cryptographic alternatives or support for authentication schemes attractive. Unlike authentication schemes relying on system-specific security policies, *Cryptographic access control (CAC)* schemes have the advantage that their security is not based on the physical security of the system on which the data is stored. Furthermore, since CAC schemes typically use data encryption to enforce access control, unauthorized access is more difficult to gain, because the data remains encrypted regardless of its location, and only a valid key can be used to decrypt it. The second observation is that access control models are prone to failures (security violation or the inability to meet their goals), which arise from the fact that security designers are likely to assume that if security schemes are correctly defined, failure is unlikely. As a consequence, cryptographic hierarchical access control has gained popularity as a solution to design multilevel security models which are more general and able to provide security in different contexts, without requiring significant changes to the fundamental architecture.

1.5 Organization of the Thesis

In this introduction we have provided an overview concerning the scenarios, motivations, state of the art and new insights on cryptographic hierarchical access control and key assignment schemes. In the following chapters we develop the formal frameworks and present our results.

The results presented in this thesis are based on joint works with Alfredo De Santis, Barbara Masucci, Francesco Palmieri, Xinyi Huang and Jin Li. The

organization of the rest of this thesis is as follows.

- **Chapter 2**: In this chapter we analyze and clarify all the relations existing between the security notions for Hierarchical Key Assignment Schemes (HKASs). More precisely, the most important contribution presented in this chapter is the proof of the equivalence between the notions of key-indistinguishability [4] and strong key indistinguishability [42], which answers an important open question in the field of hierarchical key assignment schemes. The results presented in this chapter can be found in [24].

- **Chapter 3**: In this chapter we formally extend the notions of security proposed by Atallah et al. [4] to address the further changes introduced by dynamic and adaptive updates to the access control hierarchy. More precisely, in this chapter we introduce novel security definitions which are able to model the behavior of an adversary that, besides being able to corrupt a certain set of users, it is also able to modify dynamically the access structure it intends to attack, according to the specific scenario. Again, we propose a HKAS, based on symmetric encryption schemes, which is secure with respect to our novel proposed model. The results presented in this chapter can be found in [25].

- **Chapter 4**: In this chapter we extend the classical hierarchical access control model to deal with scenarios which require the shared reconstruction of the secret key. Moreover, we propose two HKASs, which are secure with respect to the new model. More precisely, the former is based on symmetric encryption schemes and perfect secret sharing schemes, whereas, the latter is based on threshold broadcast encryption schemes. The results presented in this chapter can be found in [23] [26].

- **Chapter 5**: Finally, in this chapter we conclude the thesis, by providing discussions and some final remarks.

1.6 Notation

In this thesis we use the standard notation to describe probabilistic algorithm and experiments following [45]. If $A(\cdot, \cdot, \ldots)$ is any probabilistic algorithm then $a \leftarrow A(x, y, \ldots)$ denotes the experiment of running A on inputs $x, y, \ldots$ and letting a be the outcome, the probability being over the coins of A. Similarly, if X is a set, then $x \leftarrow X$ denotes the experiment of selecting an element uniformly from X and assigning x this value. If w is neither an algorithm nor a set, then $x \leftarrow w$ is a simple assignment statement. A function $\epsilon : N \rightarrow R$ is *negligible* if for every constant $c > 0$ there exists an integer n_c such that $\epsilon(n) < n^{-c}$ for all $n \geq n_c$.

Chapter 2

Key Indistinguishability vs. Strong Key Indistinguishability for Hierarchical Key Assignment Schemes

"A proof is whatever convinces me."

— Shimon Even, *1935-2004*

2.1 Introduction

In this chapter we explore the relations between all security notions for hierarchical key assignment schemes, by clarifying implications and separations occurring between such notions. In particular, we show that security with respect to strong key indistinguishability is *not stronger* than the one with respect to key indistinguishability, thus establishing the equivalence between such two security notions. A similar result has been recently shown in the *unconditionally secure setting* [20]. Furthermore, we also show how to construct a hierarchical key assignment scheme which is secure against strong key recovery, starting from any scheme which guarantees security against key recovery.

This chapter is organized as follows: in Section 2.2 we review the definition

of hierarchical key assignment schemes; in Section 2.3 we describe all security definitions for hierarchical key assignment schemes; in Section 2.4 we analyze the relations among these definitions and in particular we show that security with respect to strong key indistinguishability is *not stronger* than the one with respect to key indistinguishability; finally in Section 2.5, we show how to construct a hierarchical key assignment scheme secure against strong key recovery, starting from any hierarchical key assignment scheme which is secure against key recovery.

2.2 Hierarchical Key Assignment Schemes

Consider a set of users divided into a number of disjoint classes, called *security classes*. A security class can represent a person, a department or a user group in an organization. A binary relation $\preceq$ that partially orders the set of classes V is defined in accordance with authority, position or power of each class in V. The poset $(V, \preceq)$ is called a *partially ordered hierarchy*. For any two classes u and v, the notation $u \preceq v$ is used to indicate that the users in v can access u's data. Clearly, since v can access its own data, it holds that $v \preceq v$, for any $v \in V$. We denote the accessible set of a class v by A_v, which corresponds to the set $\{u \in V : u \preceq v\}$, for any $v \in V$. The partially ordered hierarchy $(V, \preceq)$ can be represented by the directed graph $G^* = (V, E^*)$, where each class corresponds to a vertex in the graph and there is an edge from class v to class u if and only if $u \preceq v$. We denote by $G = (V, E)$ the *minimal representation* of the graph G^*, namely, the directed acyclic graph corresponding to the *transitive and reflexive reduction* of the graph $G^* = (V, E^*)$. The graph G has the same transitive and reflexive closure of G^*, i.e., there is a path (of length greater than or equal to zero) from v to u in G if and only if there is the edge (v, u) in E^*. Aho et al. [1] showed that every directed graph has a transitive reduction, which can be computed in polynomial time and is unique for directed acyclic graphs. In the following, we denote by Γ a family of graphs corresponding to partially ordered hierarchies. For example, Γ could be the family of the rooted trees [64], the family of the d-dimensional hierarchies [5], etc..

A hierarchical key assignment scheme for a family Γ of graphs, corresponding to partially ordered hierarchies, is defined as follows in [8, 32, 37, 38, 29, 30, 33, 7].

Definition 2.2.1. *A* hierarchical key assignment scheme *for* Γ *is a pair* (Gen, Der) *of algorithms satisfying the following conditions:*

1. *The* information generation algorithm *Gen is probabilistic polynomial-time. It takes as inputs the security parameter* 1^τ *and a graph* $G = (V, E)$ *in* Γ*, and produces as outputs*

   (a) *a private information* s_u*, for any class* $u \in V$*;*

   (b) *a key* $k_u \in \{0,1\}^\tau$*, for any class* $u \in V$*;*

   (c) *a public information pub.*

   We denote by (s, k, pub) *the output of the algorithm Gen on inputs* 1^τ *and* G*, where* s *and* k *denote the sequences of private information and keys, respectively.*

2. *The* key derivation algorithm *Der is deterministic polynomial-time. It takes as inputs the security parameter* 1^τ*, a graph* $G = (V, E)$ *in* Γ*, two classes* u, v *in* V*, the private information* s_u *assigned to class* u *and the public information pub, and produces as output the key* $k_v \in \{0,1\}^\tau$ *assigned to class* v *if* $v \in A_u$*, or a special rejection symbol* $\perp$ *otherwise.*

   We require that for each class $u \in V$*, each class* $v \in A_u$*, each private information* s_u*, each key* $k_v \in \{0,1\}^\tau$*, each public information pub which can be computed by Gen on inputs* 1^τ *and* G*, it holds that*

$$Der(1^\tau, G, u, v, s_u, pub) = k_v.$$

2.3 Notions of Security

A hierarchical key assignment scheme must be resistant to *collusive attacks*. More precisely, for each class $u \in V$, the key k_u should be protected against a coalition of all users in the set $F_u = \{v \in V : u \notin A_v\}$, corresponding to the ones which are not allowed to compute the key k_u.

Atallah et al. [4] first introduced two different security goals for hierarchical key assignment schemes: security with respect to *key-indistinguishability* and

security against *key recovery*. The former formalizes the requirement that the adversary is not able to learn any information (even a single bit) about a key k_u which it should not have access to, i.e., it is not able to distinguish it from a random string having the same length. On the other hand, the latter corresponds to the weaker requirement that an adversary is not able to compute a key k_u which it should not have access to. The notion of key indistinguishability offers security guarantees that cannot be achieved by schemes whose security relies only upon key recovery. These stronger security guarantees could be necessary. For example, as pointed out in [33], it is straightforward that the key indistinguishability notion is needed when the data associated to a class are protected by means of a symmetric encryption scheme, whose implementation details make the confidentiality of the ciphertext (or of part of it) depending on the secrecy of only a portion of the encryption key.

Recently, Freire et al. [42] proposed a new security definition for hierarchical key assignment schemes. Such a definition, called security with respect to *strong key-indistinguishability*, formalizes the requirement that the adversary is not able to learn any information about a key k_u which it should not have access to, *even if it has the additional capability of gaining access to the keys associated to all other classes which are predecessors of the target class in the hierarchy.* Notice that these encryption keys might leak through usage and their compromise could not directly lead to a compromise of the private information s_u or the encryption key k_u of the target class u. Freire et al. also introduced the definition of security against *strong key recovery*. Such a definition formalizes the requirement that the adversary is not able to compute a key k_u which it should not have access to, even if it has the additional capability of gaining access to encryption keys assigned to all the other classes which are predecessors of the target class in the hierarchy.

In the following, we consider a *static* adversary which, given a class u, is allowed to gain the private information assigned to all users not allowed to access such class, as well as all the relative public information. For the case of *strong key indistinguishability* and *strong key recovery*, such an adversary is also able to access keys assigned to all other classes which are predecessors of the target class in the hierarchy. A different kind of adversary, the *adaptive* one, could

be also considered. In detail, such an adversary is first allowed to access all public information as well as all private information of a number of classes of its choice; afterwards, it chooses the class u it wants to attack. In [8, 7] it has been proven that security with respect to adaptive adversaries is (*polynomially*) equivalent to the one against static ones. In particular, the scenario considered in [8, 7, 9] is more general, since the lifetime of each key is limited to a given period of time. In such a setting, each class is assigned to a different key for each different period of time. These schemes are called *Time-Bound Hierarchical Key Assignment Schemes*. However, the equivalence between adaptive and static adversaries shown in [8, 7] also applies to hierarchical key assignment schemes, since they can be seen as time-bound hierarchical key assignment schemes with a single period of time. Therefore, in this thesis we will only consider static adversaries.

2.3.1 Security with respect to Key Indistinguishability

Consider a *static adversary* $\texttt{STAT}_u$ that wants to attack a class $u \in V$ and which is able to corrupt *all* users in F_u. We define an algorithm $Corrupt_u$, which on input the private information s generated by the algorithm Gen, extracts the secret values s_v associated to all classes $v \in F_u$. We denote by $corr_u$ the sequence output by $Corrupt_u(s)$. Two experiments are considered. In the first one, the adversary is given the key k_u, whereas, in the second one, it is given a random string ρ having the same length as k_u. It is the adversary's job to determine whether the received challenge corresponds to k_u or to a random string. We require that the adversary will succeed with probability only negligibly different from 1/2.

Definition 2.3.1. `[IND-ST]` *Let Γ be a family of graphs corresponding to partially ordered hierarchies, let $G = (V, E)$ be a graph in Γ, let (Gen, Der) be a hierarchical key assignment scheme for Γ and let $\texttt{STAT}_u$ be a static adversary which attacks a class u. Consider the following two experiments:*

Experiment $\mathbf{Exp}^{\mathtt{IND-1}}_{\mathtt{STAT}_u}(1^\tau, G)$
$(s, k, pub) \leftarrow Gen(1^\tau, G)$
$corr_u \leftarrow Corrupt_u(s)$
$d \leftarrow \mathtt{STAT}_u(1^\tau, G, pub, corr_u, k_u)$
return d

Experiment $\mathbf{Exp}^{\mathtt{IND-0}}_{\mathtt{STAT}_u}(1^\tau, G)$
$(s, k, pub) \leftarrow Gen(1^\tau, G)$
$corr_u \leftarrow Corrupt_u(s)$
$\rho \leftarrow \{0,1\}^\tau$
$d \leftarrow \mathtt{STAT}_u(1^\tau, G, pub, corr_u, \rho)$
return d

The advantage of $\mathtt{STAT}_u$ *is defined as*

$$\begin{aligned}\mathbf{Adv}^{\mathtt{IND}}_{\mathtt{STAT}_u}(1^\tau, G) &= |Pr[\mathbf{Exp}^{\mathtt{IND-1}}_{\mathtt{STAT}_u}(1^\tau, G) = 1] \\ &- Pr[\mathbf{Exp}^{\mathtt{IND-0}}_{\mathtt{STAT}_u}(1^\tau, G) = 1]|.\end{aligned}$$

The scheme is said to be secure in the sense of `IND-ST` *if, for each graph* $G = (V, E)$ *in* Γ *and each* $u \in V$*, the function* $\mathbf{Adv}^{\mathtt{IND}}_{\mathtt{STAT}_u}(1^\tau, G)$ *is negligible, for each static adversary* $\mathtt{STAT}_u$ *whose time complexity is polynomial in* τ*.*

2.3.2 Security with respect to Key Recovery

Now consider the case where there is a static adversary $\mathtt{STAT}_u$ which wants to *compute* the key assigned to a class $u \in V$. As done before, we denote by $corr_u$ the sequence output by the algorithm $Corrupt_u$, on input the private information s generated by the algorithm Gen. The adversary, on input all public information generated by the algorithm Gen, as well as the private information $corr_u$, outputs a string k'_u and succeeds if $k'_u = k_u$. We require that the adversary will succeed with probability only negligibly different from $1/2^\tau$.

Definition 2.3.2. `[REC-ST]` *Let* Γ *be a family of graphs corresponding to partially ordered hierarchies, let* $G = (V, E)$ *be a graph in* Γ*, let* (Gen, Der) *be a hierarchical key assignment scheme for* Γ *and let* $\mathtt{STAT}_u$ *be a static adversary which attacks a class* u*. Consider the following experiment:*

$$\begin{array}{l}\text{Experiment } \mathbf{Exp}^{\mathtt{REC}}_{\mathtt{STAT}_u}(1^\tau, G)\\ \quad (s, k, pub) \leftarrow Gen(1^\tau, G)\\ \quad corr_u \leftarrow Corrupt_u(s)\\ \quad k'_u \leftarrow \mathtt{STAT}_u(1^\tau, G, pub, corr_u)\\ \quad \textbf{return } k'_u\end{array}$$

The advantage of $\mathtt{STAT}_u$ *is defined as*

$$\mathbf{Adv}^{\mathtt{REC}}_{\mathtt{STAT}_u}(1^\tau, G) = Pr[\mathbf{Exp}^{\mathtt{REC}}_{\mathtt{STAT}_u}(1^\tau, G) = k_u].$$

The scheme is said to be secure in the sense of `REC-ST` *if, for each graph* $G = (V, E)$ *in* Γ *and each class* $u \in V$*, the function* $\mathbf{Adv}^{\mathtt{REC}}_{\mathtt{STAT}_u}(1^\tau, G)$ *is negligible, for each static adversary* $\mathtt{STAT}_u$ *whose time complexity is polynomial in* τ*.*

2.3.3 Security with respect to Strong Key Indistinguishability

Consider a *static adversary* $\mathtt{STAT}_u$ that wants to attack a class $u \in V$. Such adversary is able to corrupt *all* users in F_u and to gain access to the keys associated to all classes in the set $P_u = \{v \in V \setminus \{u\} : u \in A_v\}$ of the predecessors of class u. As done before, we denote by $corr_u$ the sequence output by the algorithm $Corrupt_u$, on input the private information s generated by the algorithm Gen. Moreover, we define an algorithm $Keys_u$, which on input the encryption keys k generated by the algorithm Gen, extracts keys k_v associated to all classes $v \in P_u$. We denote by $keys_u$ the sequence output by $Keys_u(k)$. Two experiments are considered. In the first one, the adversary is given the key k_u, whereas, in the second one, it is given a random string ρ having the same length as k_u. It is the adversary's job to determine whether the received challenge corresponds to k_u or to a random string. We require that the adversary will succeed with probability only negligibly different from $1/2$.

Definition 2.3.3. `[STRONG-IND-ST]` *Let* Γ *be a family of graphs corresponding to partially ordered hierarchies, let* $G = (V, E)$ *be a graph in* Γ*, let* (Gen, Der) *be a hierarchical key assignment scheme for* Γ *and let* $\mathtt{STAT}_u$ *be a static adversary which attacks a class* u*. Consider the following two experiments:*

Experiment $\mathbf{Exp}^{\texttt{STRONG-IND-1}}_{\texttt{STAT}_u}(1^\tau, G)$
$(s, k, pub) \leftarrow Gen(1^\tau, G)$
$corr_u \leftarrow Corrupt_u(s)$
$keys_u \leftarrow Keys_u(k)$
$d \leftarrow \texttt{STAT}_u(1^\tau, G, pub, corr_u, keys_u, k_u)$
return d

Experiment $\mathbf{Exp}^{\texttt{STRONG-IND-0}}_{\texttt{STAT}_u}(1^\tau, G)$
$(s, k, pub) \leftarrow Gen(1^\tau, G)$
$corr_u \leftarrow Corrupt_u(s)$
$keys_u \leftarrow Keys_u(k)$
$\rho \leftarrow \{0,1\}^\tau$
$d \leftarrow \texttt{STAT}_u(1^\tau, G, pub, corr_u, keys_u, \rho)$
return d

The advantage of $\texttt{STAT}_u$ *is defined as*

$$\begin{aligned}\mathbf{Adv}^{\texttt{STRONG-IND}}_{\texttt{STAT}_u}(1^\tau, G) &= |Pr[\mathbf{Exp}^{\texttt{STRONG-IND-1}}_{\texttt{STAT}_u}(1^\tau, G) = 1] \\ &- Pr[\mathbf{Exp}^{\texttt{STRONG-IND-0}}_{\texttt{STAT}_u}(1^\tau, G) = 1]|.\end{aligned}$$

The scheme is said to be secure in the sense of `STRONG-IND-ST` *if, for each graph* $G = (V, E)$ *in* Γ *and each* $u \in V$*, the function* $\mathbf{Adv}^{\texttt{STRONG-IND}}_{\texttt{STAT}_u}(1^\tau, G)$ *is negligible, for each static adversary* $\texttt{STAT}_u$ *whose time complexity is polynomial in* τ*.*

2.3.4 Security against Strong Key Recovery

Finally, consider the case where there is a static adversary $\texttt{STAT}_u$ that wants to *compute* the key assigned to a class $u \in V$. Such adversary is able to corrupt *all* users in F_u and gain access to the keys associated to all classes in the set P_u of the predecessors of u. As done before, we denote by $corr_u$ the sequence output by the algorithm $Corrupt_u$, on input the private information s generated by the algorithm Gen. Moreover, we denote by $keys_u$ the sequence output by $Keys_u(k)$. The adversary, on input all public information generated by the algorithm Gen, as well as the private information $corr_u$ and the sequence $keys_u$,

outputs a string k'_u and succeeds if $k'_u = k_u$. We require that the adversary will succeed with probability only negligibly different from $1/2^\tau$.

Definition 2.3.4. [STRONG-REC-ST] *Let Γ be a family of graphs corresponding to partially ordered hierarchies, let $G = (V, E)$ be a graph in Γ, let (Gen, Der) be a hierarchical key assignment scheme for Γ and let $\mathtt{STAT}_u$ be a static adversary which attacks a class u. Consider the following experiment:*

Experiment $\mathbf{Exp}^{\mathtt{STRONG-REC}}_{\mathtt{STAT}_u}(1^\tau, G)$
$(s, k, pub) \leftarrow Gen(1^\tau, G)$
$corr_u \leftarrow Corrupt_u(s)$
$keys_u \leftarrow Keys_u(k)$
$k'_u \leftarrow \mathtt{STAT}_u(1^\tau, G, pub, corr_u, keys_u)$
return k'_u

The advantage of $\mathtt{STAT}_u$ is defined as

$$\mathbf{Adv}^{\mathtt{STRONG-REC}}_{\mathtt{STAT}_u}(1^\tau, G) = Pr[\mathbf{Exp}^{\mathtt{STRONG-REC}}_{\mathtt{STAT}_u}(1^\tau, G) = k_u].$$

The scheme is said to be secure in the sense of STRONG-REC-ST *if, for each graph $G = (V, E)$ in Γ and each class $u \in V$, the function $\mathbf{Adv}^{\mathtt{STRONG-REC}}_{\mathtt{STAT}_u}(1^\tau, G)$ is negligible, for each static adversary $\mathtt{STAT}_u$ whose time complexity is polynomial in τ.*

2.4 Implications and Separations

In this section, we analyze the relations between the security definitions described in Section 2.3. In particular, we show implications and separations occurring between such notions. Figure 2.1 summarizes our results.

It is easy to see that any adversary which breaks the security of the key assignment scheme in the sense of STRONG-REC-ST can be easily turned into another adversary which breaks the security of the key assignment scheme in the sense of STRONG-IND-ST. Hence, the next result holds.

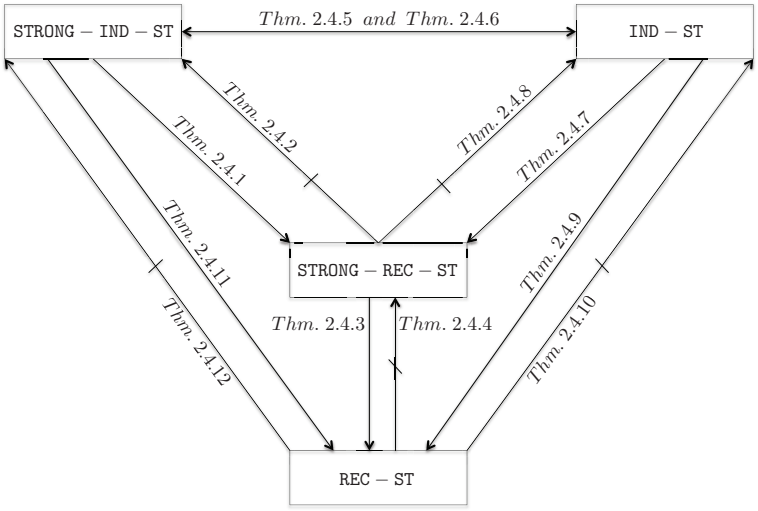


Figure 2.1: Relations between the security notions for hierarchical key assignment schemes.

Theorem 2.4.1. [STRONG-IND-ST⇒STRONG-REC-ST] *Let* Γ *be a family of graphs corresponding to partially ordered hierarchies. If a hierarchical key assignment scheme for* Γ *is secure in the sense of* STRONG-IND-ST*, then it is also secure in the sense of* STRONG-REC-ST*.*

In the following, we show that security against strong key recovery does not necessarily imply security with respect to strong key indistinguishability. Let (Gen, Der) be a hierarchical key assignment scheme which is secure in the sense of STRONG-REC-ST. We construct another scheme (Gen', Der') and we show that it is secure in the sense of STRONG-REC-ST but is not secure in the sense of STRONG-IND-ST. Let $u \in V$ be a class and let k_u be the key assigned by Gen to u. Algorithm Gen' computes the key assigned to class u as $k'_u = 1||k_u$, where the symbol $||$ denotes string concatenation. All other values computed by Gen' are exactly the same as the ones computed by Gen. Algorithm Der' first computes k_u by using Der, then obtains $k'_u = 1||k_u$. Let $\mathtt{STAT_u}$ be a static adversary that simply checks whether the first bit x_0 of the challenge x, corresponding either to the key k'_u or to a random string having the same length as k'_u, is equal to 0. If this happens, then $\mathtt{STAT_u}$ can easily conclude that the challenge x does not correspond to the key k'_u, but is a random string. Since the advantage $\mathbf{Adv}^{\mathtt{STRONG-IND}}_{\mathtt{STAT}_u}$ is non-negligible, it follows that (Gen', Der') is not secure in the sense of STRONG-IND-ST.

On the other hand, (Gen', Der') is secure in the sense of STRONG-REC-ST. Assume by contradiction that (Gen', Der') is not secure in the sense of STRONG-REC-ST. It follows that also (Gen, Der) is not secure in the sense of STRONG-REC-ST, thus leading to a contradiction. For this reason, the next result holds.

Theorem 2.4.2. [STRONG-REC-ST$\not\Rightarrow$STRONG-IND-ST] *Let Γ be a family of graphs corresponding to partially ordered hierarchies. If there exists a hierarchical key assignment scheme for Γ which is secure in the sense of* STRONG-REC-ST*, then there exists a hierarchical key assignment scheme for Γ that is secure in the sense of* STRONG-REC-ST *but which is not secure in the sense of* STRONG-IND-ST*.*

The relations between the definitions of security against strong key recovery and security against key recovery have been established by Freire et al. [42]. In particular, they showed that the two notions of security against key recovery and against strong key recovery *are separated*, i.e., there exist hierarchical key assignment schemes that are secure against key recovery but which are not secure against strong key recovery. An example of such schemes is the one based on pseudorandom functions, proposed by Atallah et al. [4]. Thus, the following theorems hold.

Theorem 2.4.3. [STRONG-REC-ST$\Rightarrow$REC-ST] *Let Γ be a family of graphs corresponding to partially ordered hierarchies. If a hierarchical key assignment scheme for Γ is secure in the sense of* STRONG-REC-ST*, then it is also secure in the sense of* REC-ST*.*

Theorem 2.4.4. [REC-ST$\not\Rightarrow$STRONG-REC-ST] *Let Γ be a family of graphs corresponding to partially ordered hierarchies. If there exists a hierarchical key assignment scheme for Γ which is secure in the sense of* REC-ST*, then there exists a hierarchical key assignment scheme for Γ that is secure in the sense of* REC-ST *but which is not secure in the sense of* STRONG-REC-ST*.*

In detail, the above implication can be inferred from the main idea underlying the separating example proposed in [42]. On the other hand, the relations between the notions of security with respect to key indistinguishability and with respect to strong key indistinguishability are not completely clear. As stated by

the next theorem, it is easy to see that security with respect to strong key indistinguishability implies security with respect to key indistinguishability. However, nothing is known about the other direction.

Theorem 2.4.5. [STRONG-IND-ST⇒IND-ST] *Let* Γ *be a family of graphs corresponding to partially ordered hierarchies. If a hierarchical key assignment scheme for* Γ *is secure in the sense of* STRONG-IND-ST*, then it is also secure in the sense of* IND-ST*.*

In the following, we show that security with respect to strong key indistinguishability is *not stronger* than the one with respect to key indistinguishability, that is to say, STRONG-IND-ST and IND-ST are equivalent.

Theorem 2.4.6. [IND-ST⇒STRONG-IND-ST] *Let* Γ *be a family of graphs corresponding to partially ordered hierarchies. If a hierarchical key assignment scheme for* Γ *is secure in the sense of* IND-ST*, then it is also secure in the sense of* STRONG-IND-ST*.*

Proof. Assume by contradiction that there exists a hierarchical key assignment scheme Σ for a graph $G = (V, E)$ in Γ, which is secure in the sense of IND-ST but that is not secure in the sense of STRONG-IND-ST. Therefore, there exists a class $u \in V$ and a static adversary $\mathtt{STAT}_u$ which is able to distinguish between experiments $\mathbf{Exp}^{\mathtt{STRONG-IND-0}}_{\mathtt{STAT}_u}$ and $\mathbf{Exp}^{\mathtt{STRONG-IND-1}}_{\mathtt{STAT}_u}$ with non-negligible advantage. Recall that the only difference between $\mathbf{Exp}^{\mathtt{STRONG-IND-0}}_{\mathtt{STAT}_u}$ and $\mathbf{Exp}^{\mathtt{STRONG-IND-1}}_{\mathtt{STAT}_u}$ is the last input of $\mathtt{STAT}_u$, which corresponds to a random value chosen in $\{0,1\}^\tau$ in the first experiment and to the real key k_u in the second one.

Let $P_u = \{v \in V \setminus \{u\} : u \in A_v\}$ be the set of predecessors of class u and let $G_u = (P_u, E_u)$ be the subgraph of G induced by P_u. Without loss of generality, let $(u_1, \ldots, u_m)$ be any topological ordering of the vertices in P_u, i.e., any linear ordering of elements in P_u such that for each edge $(u_i, u_j) \in E_u$, u_i precedes u_j in the ordering. It is well know that any *directed acyclic graph (DAG)* has at least one topological ordering. More precisely, a directed graph G has a topological ordering if and only if G is a DAG [54]. Notice that the sequence $keys_u$, which is an input of $\mathtt{STAT}_u$ in both the experiments $\mathbf{Exp}^{\mathtt{STRONG-IND-0}}_{\mathtt{STAT}_u}$ and $\mathbf{Exp}^{\mathtt{STRONG-IND-1}}_{\mathtt{STAT}_u}$, contains exactly the keys $k_{u_1}, \ldots, k_{u_m}$. First of all, it is easy to observe that if

$m = 0$ the sequence $keys_u$ is empty, thus the experiments $\mathbf{Exp}^{\mathtt{STRONG-IND-0}}_{\mathtt{STAT}_u}$ and $\mathbf{Exp}^{\mathtt{STRONG-IND-1}}_{\mathtt{STAT}_u}$ correspond to $\mathbf{Exp}^{\mathtt{IND-0}}_{\mathtt{STAT}_u}$ and $\mathbf{Exp}^{\mathtt{IND-1}}_{\mathtt{STAT}_u}$, respectively. In this case, since $\mathtt{STAT}_u$ is able to distinguish between such experiments with non-negligible advantage, it follows that the scheme Σ is not secure in the sense of IND-ST, which is a contradiction.

In addition, consider the case in which $m > 0$. We will show how to turn the adversary $\mathtt{STAT}_u$ into another polynomial-time adversary $\mathtt{STAT}'_{u_h}$, where $u_h \in P_u$, which breaks the scheme Σ in the sense of IND-ST, thus leading to a contradiction. We construct two sequences $\mathbf{Exp}^{1,1}_u, \ldots, \mathbf{Exp}^{1,m+1}_u$ and $\mathbf{Exp}^{2,1}_u, \ldots, \mathbf{Exp}^{2,m+1}_u$ of $m+1$ experiments each, all defined over the same probability space, where the first experiment of the former sequence, that is $\mathbf{Exp}^{1,1}_u$, is equal to $\mathbf{Exp}^{\mathtt{STRONG-IND-0}}_{\mathtt{STAT}_u}$, whereas, the last experiment of the latter sequence, that is $\mathbf{Exp}^{2,m+1}_u$, is equal to $\mathbf{Exp}^{\mathtt{STRONG-IND-1}}_{\mathtt{STAT}_u}$.

For any $q = 2, \ldots, m+1$, experiment $\mathbf{Exp}^{1,q}_u$ in the first sequence is defined as follows:

Experiment $\mathbf{Exp}^{1,q}_u(1^\tau, G)$
  $(s, k, pub) \leftarrow Gen(1^\tau, G)$
  $corr_u \leftarrow Corrupt_u(s)$
  $keys^q_u \leftarrow Keys^q_u(k)$
  $d \leftarrow \mathtt{STAT}_u(1^\tau, G, pub, corr_u, keys^q_u, \rho)$
  return d

The output of the algorithm $Keys^q_u$ is the sequence $keys^q_u$ where the first $q-1$ values are independently chosen at random in $\{0,1\}^\tau$ and, if $q \leq m$, the other $m - q + 1$ values correspond to the keys assigned to the classes $u_q, \ldots, u_m$.

On the other hand, for any $q = 1, \ldots, m$, experiment $\mathbf{Exp}^{2,q}_u$ in the second sequence is defined as follows:

Experiment $\mathbf{Exp}^{2,q}_u(1^\tau, G)$
  $(s, k, pub) \leftarrow Gen(1^\tau, G)$
  $corr_u \leftarrow Corrupt_u(s)$
  $keys^{m-q+2}_u \leftarrow Keys^{m-q+2}_u(k)$
  $d \leftarrow \mathtt{STAT}_u(1^\tau, G, pub, corr_u, keys^{m-q+2}_u, k_u)$
  return d

where $keys_u^{m-q+2}$ denotes the sequence where the first $m-q+1$ values are independently chosen at random in $\{0,1\}^\tau$ and, if $q \geq 2$, the other $q-1$ values correspond to the keys assigned to the classes $u_{m-q+2}, \ldots, u_m$.
Since $\mathbf{Exp}_u^{1,1}$, which corresponds to $\mathbf{Exp}_{\mathtt{STAT}_u}^{\mathtt{STRONG-IND-0}}$, and $\mathbf{Exp}_u^{2,m+1}$, which corresponds to $\mathbf{Exp}_{\mathtt{STAT}_u}^{\mathtt{STRONG-IND-1}}$, can be distinguished by $\mathtt{STAT}_u$ with non-negligible advantage, then there exists at least a pair of consecutive experiments, in the sequence of $2m+2$ experiments obtained by composition of the two above defined sequences, which are distinguishable by $\mathtt{STAT}_u$ with non-negligible advantage.

We first show that such a pair cannot consist of the two extremal experiments, namely, the last experiment of the first sequence, that is $\mathbf{Exp}_u^{1,m+1}$, and the first experiment of the second sequence, that is $\mathbf{Exp}_u^{2,1}$. Assume by contradiction that $\mathtt{STAT}_u$ is able to distinguish between $\mathbf{Exp}_u^{1,m+1}$ and $\mathbf{Exp}_u^{2,1}$ with non-negligible advantage. Notice that the only difference between such two experiments is the last input of $\mathtt{STAT}_u$, which corresponds to a random value chosen in $\{0,1\}^\tau$ in experiment $\mathbf{Exp}_u^{1,m+1}$, and to the real key k_u in experiment $\mathbf{Exp}_u^{2,1}$. We show how to construct another adversary $\mathtt{STAT'}_u$ which breaks the security of the scheme Σ in the sense of IND-ST, by using the adversary $\mathtt{STAT}_u$. The adversary $\mathtt{STAT'}_u$, on inputs 1^τ, G, the sequence of private information $corr_u$ and a final value α, corresponding either to the key k_u or to a random value chosen in $\{0,1\}^\tau$, constructs the sequence $keys_u^{m+1}$ needed for $\mathtt{STAT}_u$ choosing independently at random m elements in $\{0,1\}^\tau$. Then, $\mathtt{STAT'}_u$ outputs the same output as $\mathtt{STAT}_u(1^\tau, G, pub, corr_u, keys_u^{m+1}, \alpha)$. Clearly, since $\mathtt{STAT}_u$ is able to distinguish between $\mathbf{Exp}_u^{1,m+1}$ and $\mathbf{Exp}_u^{2,1}$ with non-negligible advantage, then $\mathtt{STAT'}_u$ is able to distinguish between $\mathbf{Exp}_{\mathtt{STAT'}_u}^{\mathtt{IND-0}}$ and $\mathbf{Exp}_{\mathtt{STAT'}_u}^{\mathtt{IND-1}}$ with non-negligible advantage, thus breaking the security of the scheme Σ in the sense of IND-ST. Contradiction. Thus, the pair of consecutive experiments which can be distinguished by $\mathtt{STAT}_u$, belongs either to the first sequence or to the second one.

Assume that the pair of distinguishable consecutive experiments belongs to the first sequence and it is composed by $\mathbf{Exp}_u^{1,h}$ and $\mathbf{Exp}_u^{1,h+1}$, for some $h = 1, \ldots, m$. Notice that the views of $\mathtt{STAT}_u$ in such two consecutive experiments differ only for a single value, which corresponds to the key k_{u_h} in $\mathbf{Exp}_u^{1,h}$ and to a random value chosen in $\{0,1\}^\tau$ in $\mathbf{Exp}_u^{1,h+1}$. We show how to construct an adversary $\mathtt{STAT''}_{u_h}$ which breaks the security of the scheme Σ in the sense of IND-ST, by using

the adversary $\mathtt{STAT}_u$. In particular, we show that $\mathtt{STAT''}_{u_h}$ is able to distinguish between experiments $\mathbf{Exp}^{\mathtt{IND-0}}_{\mathtt{STAT''}_{u_h}}$ and $\mathbf{Exp}^{\mathtt{IND-1}}_{\mathtt{STAT''}_{u_h}}$ with non-negligible advantage. The adversary $\mathtt{STAT''}_{u_h}$, on inputs 1^τ, G, the sequence of private information $corr_{u_h}$ and a final value α, corresponding either to the key k_{u_h} or to a random value chosen in $\{0,1\}^\tau$, constructs the inputs for $\mathtt{STAT}_u$ as follows:

- First, $\mathtt{STAT''}_{u_h}$ extracts from $corr_{u_h}$ the sequence $corr_u$. This can be done since $u_h \in P_u$, i.e., u_h is a predecessor of u, hence classes which are corrupted for u are also corrupted for u_h and their private information is in $corr_{u_h}$.

- Then, $\mathtt{STAT''}_{u_h}$ uses $corr_{u_h}$ and α to construct a sequence $keys^\alpha_u$, which corresponds either to $keys^h_u$ or to $keys^{h+1}_u$. In particular, the first $h-1$ elements of $keys^\alpha_u$ are independently chosen at random in $\{0,1\}^\tau$, the h-th element is set equal to α, whereas, the remaining $m-h$ ones, which correspond to the keys of the classes $u_{h+1},\ldots,u_m$, are computed by using the private information of such classes, which are contained in $corr_{u_h}$.

- Moreover, the last input for $\mathtt{STAT}_u$ is set equal to a random value ρ chosen in $\{0,1\}^\tau$.

Finally, $\mathtt{STAT''}_{u_h}$ outputs the same output as $\mathtt{STAT}_u(1^\tau, G, pub, corr_u, keys^\alpha_u, \rho)$. Clearly, since $\mathtt{STAT}_u$ is able to distinguish between $\mathbf{Exp}^{1,h}_u$ and $\mathbf{Exp}^{1,h+1}_u$ with non-negligible advantage, then $\mathtt{STAT''}_{u_h}$ is able to distinguish between $\mathbf{Exp}^{\mathtt{IND-0}}_{\mathtt{STAT''}_{u_h}}$ and $\mathbf{Exp}^{\mathtt{IND-1}}_{\mathtt{STAT''}_{u_h}}$ with non-negligible advantage, thus breaking the security of the scheme Σ in the sense of IND-ST. Contradiction.

Notice that if the pair of distinguishable consecutive experiments belongs to the second sequence, i.e., is composed by $\mathbf{Exp}^{2,h}_u$ and $\mathbf{Exp}^{2,h+1}_u$, for some $h = 1,\ldots,m$, the proof is similar to the previous case. □

From Theorems 2.4.5, 2.4.6, 2.4.1 and 2.4.2 we obtain the following results.

Theorem 2.4.7. [IND-ST⇒STRONG-REC-ST] *Let Γ be a family of graphs corresponding to partially ordered hierarchies. If a hierarchical key assignment scheme for Γ is secure in the sense of* IND-ST*, then it is also secure in the sense of* STRONG-REC-ST*.*

Theorem 2.4.8. [STRONG-REC-ST⇏IND-ST] *Let* Γ *be a family of graphs corresponding to partially ordered hierarchies. If there exists a hierarchical key assignment scheme for* Γ *which is secure in the sense of* STRONG-REC-ST*, then there exists a hierarchical key assignment scheme for* Γ *that is secure in the sense of* STRONG-REC-ST *but which is not secure in the sense of* IND-ST*.*

The next result, which has already been proven in [7], follows from Theorems 2.4.6, 2.4.1, and 2.4.3.

Theorem 2.4.9. [IND-ST⇒REC-ST] *Let* Γ *be a family of graphs corresponding to partially ordered hierarchies. If a hierarchical key assignment scheme for* Γ *is secure in the sense of* IND-ST*, then it is also secure in the sense of* REC-ST*.*

On the other hand, the next result holds [7].

Theorem 2.4.10. [REC-ST⇏IND-ST] *Let* Γ *be a family of graphs corresponding to partially ordered hierarchies. If there exists a hierarchical key assignment scheme for* Γ *which is secure in the sense of* REC-ST*, then there exists a hierarchical key assignment scheme for* Γ *that is secure in the sense of* REC-ST *but which is not secure in the sense of* IND-ST*.*

From Theorems 2.4.5, 2.4.6, and 2.4.9, we obtain the next result.

Theorem 2.4.11. [STRONG-IND-ST⇒REC-ST] *Let* Γ *be a family of graphs corresponding to partially ordered hierarchies. If a hierarchical key assignment scheme for* Γ *is secure in the sense of* STRONG-IND-ST*, then it is also secure in the sense of* REC-ST*.*

Finally, from Theorems 2.4.5, 2.4.6, and 2.4.10, the next result holds.

Theorem 2.4.12. [REC-ST⇏STRONG-IND-ST] *Let* Γ *be a family of graphs corresponding to partially ordered hierarchies. If there exists a hierarchical key assignment scheme for* Γ *which is secure in the sense of* REC-ST*, then there exists a hierarchical key assignment scheme for* Γ *that is secure in the sense of* REC-ST *but which is not secure in the sense of* STRONG-IND-ST*.*

2.5 Towards Security against Strong Key Recovery

As said in the previous section, the two notions of security against key recovery and against strong key recovery *are separated*, i.e., there exist hierarchical key assignment schemes that are secure against key recovery but which are not secure against strong key recovery. In this section, we investigate the possibility of obtaining a scheme which is secure with respect to the stronger notion, starting from any scheme which is secure with respect to the weaker one.

The idea behind our construction is the following. Given a graph $G = (V, E)$ representing a partially ordered hierarchy, we construct another graph G' which represents the same hierarchy, but that has $|V|$ *additional* classes. Then, we use a hierarchical key assignment scheme to assign private information and encryption keys to the classes of G'. This assignment can be easily turned into an assignment for the original graph G. Indeed, the private information for each class in G is set equal to that assigned to the same class in G', whereas, the encryption keys for classes in G are those assigned to the additional classes in G'. We will show how the resulting hierarchical key assignment scheme for G satisfies security against strong key recovery, provided that the underlying scheme for G' satisfies security against key recovery.

Formally, let Γ be a family of graphs corresponding to partially ordered hierarchies. For each graph $G = (V, E)$ in Γ we define a graph transformation, whose output, denoted by $G' = (V', E')$, is called the *extended graph for* G. We denote by Γ' the family of extended graphs for elements in Γ. The transformation works as follows:

- For each $u \in V$, we place two classes u and u_0 in V';
- For each class $u \in V$, we place the edge (u, u_0) in E';
- For each $(u, v) \in E$, we place the edge (u, v) in E'.

Figure 2.2 shows an example of the extended graph for $G = (V, E)$, where $V = \{a, b, c, d\}$ and $E = \{(a, b), (a, c), (b, d), (c, d)\}$.

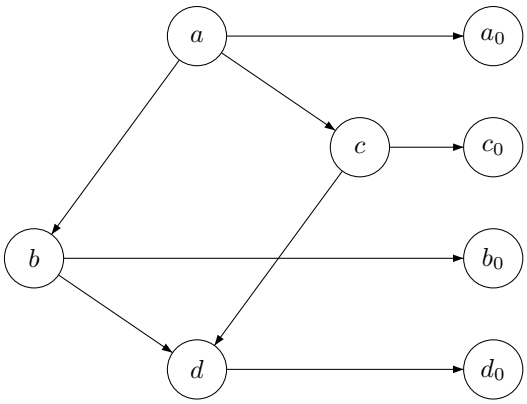


Figure 2.2: The graph $G' = (V', E')$, where $V = \{a, b, c, d\}$ and $E = \{(a, b), (a, c), (b, d), (c, d)\}$.

Let Γ' be the family of extended graphs for elements in Γ and let (Gen', Der') be a hierarchical key assignment scheme for Γ'. The proposed key assignment scheme for Γ works as follows.

Algorithm $Gen(1^\tau, G)$

1. Construct the extended graph $G' = (V', E')$ for $G = (V, E)$;
2. Let (s', k', pub') be the output of Gen' on inputs $(1^\tau, G')$;
3. For each class $u \in V$, let $s_u = s'_u$;
4. For each class $u \in V$, let $k_u = k'_{u_0}$;
5. Let s and k be the sequences of private information and keys, respectively, computed in the previous steps;
6. Output (s, k, pub).

Algorithm $Der(1^\tau, G, u, v, s_u, pub)$

1. Let k'_{v_0} be the output of Der' on inputs $(1^\tau, G', u, v_0, s'_u, pub')$;
2. Output $k_v = k'_{v_0}$.

The next theorem states that if (Gen', Der') is secure against key recovery, then (Gen, Der) is secure against strong key recovery.

Theorem 2.5.1. *If* (Gen', Der') *is secure in the sense of* REC-ST*, then* (Gen, Der) *is secure in the sense of* STRONG-REC-ST*.*

Proof. Assume by contradiction that the scheme (Gen, Der) is not secure in the sense of STRONG-REC-ST. Therefore, there exists a graph $G = (V, E)$ in Γ and a class $u \in V$ for which there exists a polynomial time adversary $\mathtt{STAT}_u$ whose advantage $\mathbf{Adv}^{\mathtt{STRONG-REC}}_{\mathtt{STAT}_u}(1^\tau, G)$ is non-negligible. We show how to construct a polynomial-time adversary which, by using $\mathtt{STAT}_u$, is able to break the security of the scheme (Gen', Der') in the sense of REC-ST. Such an adversary, which we denote by $\mathtt{STAT'}_{u_0}$, on inputs 1^τ, an extended graph G', the public information pub', and the sequence $corr'_{u_0}$ of private information held by corrupted users, constructs the inputs for $\mathtt{STAT}_u$ as follows:

- First, $\mathtt{STAT'}_{u_0}$ constructs the graph G from G', so that G' is the extended graph for G. This operation simply involves the cancellation of all the classes $v_0 \in V'$.
- Then, the adversary sets the public information pub to be equal to pub'.
- Afterwards, the adversary extracts the sequence $corr_u$ from $corr'_{u_0}$. Indeed, $corr'_{u_0}$ contains the private information s'_v for each class $v \in F_u$.
- Moreover, the adversary constructs the sequence $keys_u$ as follows: first, it extracts from the sequence $corr'_{u_0}$ the private information s'_{v_0} for each $v \neq u$. Such values are then used to compute the sequence of keys k'_{v_0} for each $v \neq u$. These values are exactly the elements of the sequence $keys_u$.

Finally, $\mathtt{STAT'}_{u_0}$ returns the same output as $\mathtt{STAT}_u(1^\tau, G, pub, corr_u, keys_u)$. Therefore, it is easy to see that

$$\mathbf{Adv}^{\mathtt{REC}}_{\mathtt{STAT'}_{u_0}}(1^\tau, G') = \mathbf{Adv}^{\mathtt{STRONG-REC}}_{\mathtt{STAT}_u}(1^\tau, G).$$

Since $\mathbf{Adv}^{\mathtt{STRONG-REC}}_{\mathtt{STAT}_u}(1^\tau, G)$ is non-negligible, it follows that the adversary $\mathtt{STAT'}_{u_0}$ is able to break the security of the scheme (Gen', Der') in the sense of REC-ST. Contradiction. □

Chapter 3

Cryptographic Hierarchical Access Control For Dynamic Structures

"Each problem that I solved became a rule, which served afterwards to solve other problems."

— Rene Descartes, *1596-1650*

3.1 Introduction

Sometimes, it is necessary to make some *dynamic updates* to the hierarchy, in order to implement an access control policy which evolves over time. For example, within a hospital system, whenever a new doctor is hired, it is necessary to assign him to a certain security class. Similarly, whenever a doctor retires, we need to remove him from the corresponding security class. The above situations, concerning single individuals, may be extended to the case where an entire security class needs to be inserted or deleted in the hierarchy. Moreover, also the relationships between the classes could change over time. For example, in a complex enterprise security system, an entire class of users with a different security profile may be added as a consequence of the acquisition of a new company or branch, or similarly, the role and mission of an entire company sector may change after a

fusion between enterprises, resulting in the need of redefining the structure of the access control hierarchy through the modification of several dependencies among the existing classes.

However, all security models proposed so far consider an operational scenario which is *fixed and immutable*. More precisely, the adversary is not allowed to make any changes to the hierarchy, which is fixed and chosen at the time of the attack. We remark that this fact represents *an important limitation*, since the existing models are not able to characterize the different networking scenarios which may arise in many operating environments.

For example, advances in wireless communication and electronics have enabled the development of *User-Centric Networks (UCNs)*, which can be considered as an abstraction of the so called *infrastructureless networks* (e.g., *AdHoc Networks*, *Sensor Networks*, etc.). UCNs have a lot of applications and an even wider spectrum of future applications is likely to follow [3]. The topology of such networks changes very frequently, due to failures or mobility [71]. In this context, each node represents a potential point of attack, thus making impractical to monitor and protect each individual node from either physical or logical attacks [40]. In particular, nodes may be susceptible to several attacks, such as *capture* and *physical tampering*. Once a node has been compromised, the adversary can physically access to this node and extract its sensitive information. Again, a node may also be *altered* or *replaced*, resulting in a compromised node under the control of the adversary. Furthermore, a node may also be *permanently destroyed* or *turned off*, so the losses are irreversible. Notice that data loss or damage can even occur due to *harsh communication environments*. Finally, also the *communication links* may become *lost or unavailable*.

As it can be easily noticed from above considerations, it is necessary to extend and improve the existing security models, by providing the adversary with further attack capabilities. That is, the adversary should be given the possibility of *performing a polynomial number of arbitrary changes to the hierarchy*. Such changes should emulate as closely as possible all the attacks that can be performed in the real world, e.g., *addition, deletion and modification of classes (nodes) and relations (edges), as well as the revocation of users*.

In this chapter we consider hierarchical key assignment schemes supporting

dynamic updates. We first propose a new security model which extends those that have been defined in the literature. In particular, we extend the notions of security against key recovery and with respect to key indistinguishability provided by Atallah et al., to address the further challenges introduced by the updates to the hierarchy. In this way, we provide the adversary with the ability to emulate all operations that can be performed in a real networking context. Moreover, we provide the first formal definition of hierarchical key assignment schemes supporting dynamic updates. Finally, we show how to construct a hierarchical key assignment scheme supporting dynamic updates, by using as a building block a symmetric encryption scheme. The proposed construction is provably secure with respect to key indistinguishability, provides efficient key derivation and updating procedures, while requiring each user to store only a single private key.

The Chapter is organized as follows. In Section 3.2 we formalize the concept of hierarchical key assignment scheme supporting dynamic updates, in particular by focusing on the types of updates as well as on the security notions. In Section 3.3 we show how to construct a hierarchical key assignment scheme which supports dynamic updates, by using as a building block a symmetric encryption scheme.

3.2 Hierarchical Key Assignment Schemes with Dynamic Updates

A hierarchical key assignment scheme for a family Γ of graphs, corresponding to partially ordered hierarchies, supporting dynamic updates is defined as follows.

Definition 3.2.1. *A* hierarchical key assignment scheme *for* Γ *supporting* dynamic updates *is a triple* (Gen, Der, Upd) *of algorithms satisfying the following conditions:*

1. *The* information generation algorithm *Gen, executed by the* TA*, is probabilistic polynomial-time. It takes as inputs the security parameter* 1^τ *and a graph* $G = (V, E)$ *in* Γ*, and produces as outputs*
   (a) *a private information* s_u*, for any class* $u \in V$*;*
   (b) *a key* $k_u \in \{0,1\}^\tau$*, for any class* $u \in V$*;*

(c) a public information pub.

We denote by (s, k, pub) *the output of the algorithm Gen on inputs* 1^τ *and* G*, where* s *and* k *denote the sequences of private information and of keys, respectively.*

2. *The* key derivation algorithm *Der, executed by some authorized user, is deterministic polynomial-time. It takes as inputs the security parameter* 1^τ*, a graph* $G = (V, E)$ *in* Γ*, two classes* $u \in V$ *and* $v \in A_u^G$*, the private information* s_u *assigned to class* u *and the public information pub, and produces as output the key* $k_v \in \{0,1\}^\tau$ *assigned to class* v*.*

   We require that for each class $u \in V$*, each class* $v \in A_u^G$*, each private information* s_u*, each key* $k_v \in \{0,1\}^\tau$*, each public information pub which can be computed by Gen on inputs* 1^τ *and* G*, it holds that*

$$Der(1^\tau, G, u, v, s_u, pub) = k_v.$$

3. *The* update algorithm *Upd, executed by the* TA*, is probabilistic polynomial-time. It takes as inputs the security parameter* 1^τ*, a graph* $G = (V, E)$ *in* Γ*, the tuple* (s, k, pub) *(generated either by Gen or by Upd itself), an* update type *up, a sequence of additional parameters params, and produces as outputs*

   (a) a updated graph $G' = (V', E')$ *in* Γ*;*

   (b) a private information s'_u*, for any class* $u \in V'$*;*

   (c) a key $k'_u \in \{0,1\}^\tau$*, for any class* $u \in V'$*;*

   (d) a public information pub′.

   The sequence params, if not empty, is used to generate new keys and secret information as a consequence of the update type up. We denote by (s', k', pub') *the sequences of private information, keys, and public information output by* $Upd(1^\tau, G, s, k, pub, up, params)$*.*

In the above definition, it is required that the updated graph G' still belongs to the family Γ of partially ordered hierarchies, i.e., only updates which preserve the partial order relation between the classes in the hierarchy are allowed.

3.2.1 Types of Updates

In this section we consider different types of updates which can be performed by using the algorithm *Upd* and we discuss how such updates modify the access rights of the classes in the hierarchy obtained after the update. The update types we consider are the following: *insertion of an edge*, *insertion of a class*, *deletion of an edge*, *deletion of a class*, *key replacement*, and *revocation of a user from a class*. Notice that some types of updates can be seen as a sequence of other types of updates. For example, the deletion of a class u can be performed by executing a sequence of edge deletions, one for each edge ingoing u and outgoing from u. On the other hand, the deletion of the edge (u, v) requires a key replacement operation for the class v. Finally, the revocation of a user from a class u requires a sequence of key replacement operations. In the following we describe each type of update.

- **Insertion of an edge.** Let u and v be two classes in V such that $(u, v) \notin E$. The insertion of the edge (u, v) in the graph $G' = (V', E')$ requires to update the accessible set of any class which was able to access u in G, in order to include the new access rights. In particular, for any class w such that $u \in A_w^G$, the updated accessible set $A_w^{G'}$ is defined to be $A_w^{G'} = A_w^G \cup A_v^G$. Moreover, the insertion of the edge (u, v) in G' also requires to update the forbidden set of any class which was accessed by v in G, in order to remove all classes which are able to access u. In particular, for any class z such that $z \in A_v^G$, the updated forbidden set $F_z^{G'}$ is defined to be $F_z^{G'} = F_z^G \setminus \{w : u \in A_w^G\}$.

- **Insertion of a class.** Let $u \notin V$ be a class to be inserted in the graph G', along with new incoming and outgoing edges. Such an update can be seen as a composition of edge insertions, considering each edge ingoing u and outgoing from u as a separate update. Consequently, the accessible

and forbidden sets of classes in G' can be determined as explained for the case of edge insertions.

- **Deletion of an edge.** Let u and v be two classes in V such that $(u, v) \in E$. The deletion of the edge (u, v) from the graph G requires to check if any class z which was able to access class u in G is still able to access class v in the updated graph G'. More precisely, we have to investigate if there exists another path from z to v avoiding the deleted edge (u, v). If such a path exists, then the accessible set $A_z^{G'}$ is set to be equal to A_z^G. On the other hand, if such a path does not exist, then class v needs to be deleted from A_z^G, and we continue to check whether there exists a path from z and each class w which can be accessed by v, in order to decide whether w has to be deleted from A_z^G [55, 51]. Moreover, the forbidden set of each class w which can be accessed by class v, needs to be updated in order to include all classes whose unique path to w has been broken by the deleted edge (u, v).

- **Deletion of a class.** Let $u \in V$ be a class to be deleted in the graph G, along with its incoming and outgoing edges, thus yielding to the graph G'. This update requires to follow the above described procedure for edge deletion. Moreover, in order to preserve the partial order relation between the classes, such an update also requires to insert a new edge between each predecessor and each successor of the deleted class u.

- **Key replacement.** Let u be a class in G whose key k_u needs to be replaced, due either to a problem of loss, misuse or after an edge or class deletion in the hierarchy. Such an update does not change the structure of the hierarchy, consequently no accessible or forbidden set needs to be modified.

- **User revocation.** Let u be a class in G, containing a certain number of users which share the same access rights. Whenever a user in u needs to be revoked, we need to choose a new secret information s'_u, which is then distributed to each non-revoked user in class u. This update does not alter the composition of the accessible set $A_u^{G'}$, which is set equal to A_u^G. However, in order to avoid the so called *ex-member problem*, a *key replacement update* for each class $v \in A_u^{G'}$ is needed.

Notice that the first four update types result in a structural modification of the hierarchy, whereas, the last two do not affect its structure. In particular, the last type of update represents a modification of the access control policy.

The *efficiency* of a hierarchical key assignment scheme supporting dynamic updates is evaluated mainly according to the complexity of the updates due to dynamic changes to the hierarchy. In particular, we would like to support dynamic changes by means of only local updates to the public information, without re-distributing private information to the classes affected by such changes. It is important to note that such a re-distribution cannot be avoided in the case of user revocation from a class, which necessarily requires to re-distribute the secret values to the non-revoked users in that class. However, it is desirable that no other private information must be updated.

3.2.2 Security Issues

In the following we discuss the *security* issues for hierarchical key assignment schemes supporting dynamic updates. According to the *security reduction paradigm* introduced by Goldwasser and Micali [44], a scheme is *provably-secure* under a complexity assumption if the existence of an adversary A breaking the scheme is equivalent to the existence of an adversary B breaking the computational assumption [44]. The security notions proposed by Atallah et al. [4] and further extended by Ateniese et al. [7] have been designed to deal with *static hierarchies*.

The above definitions need to be extended in order to address the additional security challenges introduced by the algorithm *Upd* used for handling dynamic updates to the hierarchy. In order to evaluate the security of a hierarchical key assignment scheme supporting dynamic updates, we consider a *dynamic adaptive adversary* `ADAPT` attacking the scheme. Such an adversary can make three different types of operations: *performing a dynamic update*, *corrupting a class*, and *attacking a class*.

Performing a Dynamic Update. The first type of operation comprises all kinds of updates described in Section 3.2.1, i.e., *insertions* and *deletions* of classes or edges, *key replacements* and *user revocations*. We assume the existence of

an *updating oracle* $\mathcal{U}$, modeling the behavior of the *TA*, which performs the required updates on the hierarchy. At the beginning, the state of the updating oracle is represented by the tuple (G^0, s^0, k^0, pub^0), where (s^0, k^0, pub^0) is the output of algorithm *Gen* on inputs 1^τ and the initial graph G^0. For any $i \geq 0$, the $(i+1)$-th adversary's query to the updating oracle consists of a pair $(up^{i+1}, params^{i+1})$, where up^{i+1} is an update operation on the graph G^i and $params^{i+1}$ is the sequence of parameters associated to the update, which the oracle answers with the updated graph G^{i+1}, the public information pub^{i+1} associated to G^{i+1}, and with a sequence of keys, denoted by old_k^i, which have been modified as a consequence of the update, according to the specification of the algorithm *Upd*. More precisely, the updating oracle $\mathcal{U}_{(1^\tau, G^i, s^i, k^i, pub^i)}(\cdot,\cdot)$, given the query $(up^{i+1}, params^{i+1})$, runs algorithm $Upd(1^\tau, G^i, s^i, k^i, pub^i, up^{i+1}, params^{i+1})$ and returns G^{i+1}, pub^{i+1}, and old_k^i to the adversary, where old_k^i is a subsequence of k^i. Thus, $\mathcal{U}_{(1^\tau, G^i, s^i, k^i, pub^i)}(up^{i+1}, params^{i+1})$ behaves as $Upd(1^\tau, G^i, s^i, k^i, pub^i, up^{i+1}, params^{i+1})$. Moreover, in order to be ready to reply to the next update query, the oracle updates its state to be $(G^{i+1}, s^{i+1}, k^{i+1}, pub^{i+1})$. In the following, for the sake of simplicity, we denote by $\mathcal{U}^i(\cdot,\cdot)$ the oracle $\mathcal{U}_{(1^\tau, G^i, s^i, k^i, pub^i)}(\cdot,\cdot)$. Due to its adaptive nature, the adversary may require a polynomial number $m = poly(|V|, 1^\tau)$ of dynamic updates, where each update is decided on the basis of the answers obtained from the updating oracle at the previous steps.

Corrupting a Class. The second type of operation is the *class corruption*, which can be performed again in an adaptive order and for a polynomial number of classes. For any $i \geq 0$, we assume the existence of a *corrupting oracle* $\mathcal{C}^i$, which provides the adversary with the private information held by the corrupted classes in the graph G^i. In particular, an adversary's query to the corrupting oracle $\mathcal{C}^i$ consists of a class v in the graph G^i, which the oracle answers with the private information held by class v in all graphs $G^0, G^1, \ldots, G^i$ (if v belongs to them). More precisely, on input a class v in G^i, the corrupting oracle $\mathcal{C}^i_{(s^0, s^1, \ldots, s^i)}(\cdot)$ returns the private information s^j_v, for any $j = 0, \ldots, i$ such that v is in the hierarchy G^j. In the following, for the sake of simplicity, we denote by $\mathcal{C}^i(\cdot)$ the oracle $\mathcal{C}^i_{(s^0, s^1, \ldots, s^i)}(\cdot)$.

Attacking a Class. The third type of operation is the *class attack*, where the adversary chooses an update index t and a class u in the hierarchy G^t and is challenged either in computing the key k^t_u or in distinguishing k^t_u from a random

string in $\{0,1\}^\tau$, depending on the security requirement.

In particular, we consider an adversary $\mathtt{ADAPT} = (\mathtt{ADAPT}_1, \mathtt{ADAPT}_2)$ running in two stages. In advance of the adversary's execution, the algorithm Gen is run on inputs 1^τ and G and outputs the tuple (s, k, pub), which is kept hidden from the adversary, with the exception of the public information pub. During the first stage, the adversary $\mathtt{ADAPT}_1$ is given access to both updating and corrupting oracles for a polynomial number m of times. The responses obtained by the oracles are saved in some state information denoted as *history*. In particular, *history* contains the following information: 1) all graphs $G^0, G^1, \ldots, G^m$; 2) the sequence of updating operations $up^1, \ldots, up^m$ queried to the updating oracle; 3) the corresponding sequences of public information $pub^0, pub^1, \ldots, pub^m$; 4) the corresponding sequences of keys $old\_k^0, \ldots, old\_k^{m-1}$, which have been modified according to each update; 5) the private information held by all corrupted classes. After interacting with the updating and corrupting oracles, the adversary chooses an update index t and a class u in G^t, among all the classes in G^t which cannot be accessed by the corrupted classes. In particular, the chosen class u is such that, for any class v already queried to the corrupting oracle $\mathcal{C}^i(\cdot)$ and any $i = 0, \ldots, m$, v cannot access u in the hierarchy G^i. In the second stage, the adversary $\mathtt{ADAPT}_2$ is given again access to the corrupting oracle and is then challenged either in computing the key k_u^t assigned to u or in distinguishing k_u^t from a random string $\rho \in \{0,1\}^\tau$. Clearly, it is required that the key k_u^t on which the adversary will be challenged is not included in the sequence old_k^{t-1} of keys which have been updated in the graph G^t.

Security with respect to Key Indistinguishability. The next definition formalizes the *key indistinguishability* requirement for hierarchical key assignment schemes supporting dynamic updates.

Definition 3.2.2. `[IND-DYN-AD]` *Let Γ be a family of graphs corresponding to partially ordered hierarchies, let $G = (V, E) \in \Gamma$ be a graph, and let (Gen, Der, Upd) be a hierarchical key assignment scheme for Γ supporting dynamic updates. Let $m = poly(|V|, 1^\tau)$ and let $\mathtt{ADAPT} = (\mathtt{ADAPT}_1, \mathtt{ADAPT}_2)$ be a dynamic adaptive adversary that during the first stage of the attack is given access both to the updating oracle $\mathcal{U}^i(\cdot,\cdot)$ and the corrupting oracle $\mathcal{C}^i(\cdot)$, for $i = 1, \ldots, m$,*

and during the second stage of the attack is given access only to the corrupting oracle. Consider the following two experiments:

Experiment $\mathbf{Exp}^{\texttt{IND-DYN-1}}_{\texttt{ADAPT}}(1^\tau, G)$
$(s, k, pub) \leftarrow Gen(1^\tau, G)$
$(t, u, history) \leftarrow \texttt{ADAPT}_1^{\mathcal{U}^i(\cdot,\cdot),\mathcal{C}^i(\cdot)}(1^\tau, G, pub)$
$d \leftarrow \texttt{ADAPT}_2^{\mathcal{C}^i(\cdot)}(1^\tau, t, u, history, k_u^t)$
return d

Experiment $\mathbf{Exp}^{\texttt{IND-DYN-0}}_{\texttt{ADAPT}}(1^\tau, G)$
$(s, k, pub) \leftarrow Gen(1^\tau, G)$
$(t, u, history) \leftarrow \texttt{ADAPT}_1^{\mathcal{U}^i(\cdot,\cdot),\mathcal{C}^i(\cdot)}(1^\tau, G, pub)$
$\rho \leftarrow \{0,1\}^\tau$
$d \leftarrow \texttt{ADAPT}_2^{\mathcal{C}^i(\cdot)}(1^\tau, t, u, history, \rho)$
return d

It is required that the class u output by $\texttt{ADAPT}_1$ *is such that v cannot access u in the graph G^i, for any class v already queried to the corrupting oracle $\mathcal{C}^i(\cdot)$. Moreover, it is also required that* $\texttt{ADAPT}_2$ *never queries the corrupting oracle $\mathcal{C}^i(\cdot)$ on a class v such that v can access u in the graph G^t. The advantage of* ADAPT *is defined as*

$$\begin{aligned}\mathbf{Adv}^{\texttt{IND-DYN}}_{\texttt{ADAPT}}(1^\tau, G) &= |Pr[\mathbf{Exp}^{\texttt{IND-DYN-1}}_{\texttt{ADAPT}}(1^\tau, G) = 1] \\ &- Pr[\mathbf{Exp}^{\texttt{IND-DYN-0}}_{\texttt{ADAPT}}(1^\tau, G) = 1]|\end{aligned}$$

The scheme is said to be secure in the sense of IND-DYN-AD *if for each graph $G = (V, E)$ in Γ, the function $\mathbf{Adv}^{\texttt{IND-DYN}}_{\texttt{ADAPT}}(1^\tau, G)$ is negligible, for each adaptive adversary* ADAPT *whose time complexity is polynomial in τ.*

Notice that if the adversary $\texttt{ADAPT}_1$ never queries the updating oracle during the first stage of the attack, the above definition reduces to that of security with respect to key indistinguishability against adaptive adversaries for hierarchical key assignment schemes with static hierarchies, referred to as IND-AD in [7]. For

such kind of schemes, it has been proven that adaptive adversaries are *polynomialy equivalent* to *static* adversaries, i.e., such that the class to be attacked is chosen in advance to the execution of the scheme.

Security against Key Recovery. Now, we consider the weaker requirement of *security against key recovery*. As done before, we assume the existence of the oracles $\mathcal{U}^i$ and $\mathcal{C}^i$. We require that the adversary will guess the key k_u^t with probability only negligibly different from $1/2^\tau$.

Definition 3.2.3. [REC-DYN-AD] *Let Γ be a family of graphs corresponding to partially ordered hierarchies, let $G = (V, E) \in \Gamma$ be a graph and let (Gen, Der, Upd) be a hierarchical key assignment scheme for Γ supporting dynamic updates. Let $m = poly(|V|, 1^\tau)$ and let $\mathtt{ADAPT} = (\mathtt{ADAPT}_1, \mathtt{ADAPT}_2)$ be a dynamic adaptive adversary that during the first stage of the attack is given access both to the updating oracle $\mathcal{U}^i(\cdot,\cdot)$ and the corrupting oracle $\mathcal{C}^i(\cdot)$, for $i = 1, \ldots, m$, and during the second stage of the attack is given access only to the corrupting oracle. Consider the following experiment:*

Experiment $\mathbf{Exp}^{\mathtt{REC-DYN}}_{\mathtt{ADAPT}}(1^\tau, G)$
$(s, k, pub) \leftarrow Gen(1^\tau, G)$
$(t, u, history) \leftarrow \mathtt{ADAPT}_1^{\mathcal{U}^i(\cdot,\cdot),\mathcal{C}^i(\cdot)}(1^\tau, G, pub)$
$k_u^{t,*} \leftarrow \mathtt{ADAPT}_2^{\mathcal{C}^i(\cdot)}(1^\tau, t, u, history)$
return $k_u^{t,*}$

It is required that the class u output by $\mathtt{ADAPT}_1$ is such that v cannot access u in the graph G^i, for any class v already queried to the corrupting oracle $\mathcal{C}^i(\cdot)$. Moreover, it is also required that $\mathtt{ADAPT}_2$ never queries the corrupting oracle $\mathcal{C}^i(\cdot)$ on a class v such that v can access u in the graph G^t. The advantage of $\mathtt{ADAPT}$ is defined as

$$\mathbf{Adv}^{\mathtt{REC-DYN}}_{\mathtt{ADAPT}}(1^\tau, G) = Pr[k_u^{t,*} = k_u^t].$$

The scheme is said to be secure in the sense of REC-DYN-AD *if, for each graph $G = (V, E)$ in Γ, the function $\mathbf{Adv}^{\mathtt{REC-DYN}}_{\mathtt{ADAPT}}(1^\tau, G)$ is negligible, for each adaptive adversary $\mathtt{ADAPT}$ whose time complexity is polynomial in τ.*

If the adversary $\texttt{ADAPT}_1$ never queries the updating oracle during the first stage of the attack, the above definition reduces to that of security against key recovery in presence of adaptive adversaries for hierarchical key assignment schemes with static hierarchies, referred to as `REC-AD` in [7].

3.3 A Construction based on Symmetric Encryption Schemes

In this section we consider the problem of constructing a hierarchical key assignment scheme supporting dynamic updates using as a building block a symmetric encryption scheme. In particular, we consider the *Two-Level Encryption-Based Construction* (TLEBC) proposed in [7]. Such a construction belongs to the class of *time-bound* hierarchical key assignment schemes, since the key derivation depends not only on the relations between the classes, but also on time constraints. However, since in this thesis we are not interested in time-bound schemes, we describe a simplified version of the scheme, without time constraints. Later on, we prove that the security property of the TLEBC depends on the security property of the underlying encryption scheme. We need to recall the definition of a symmetric encryption scheme and its notions of security.

3.3.1 Symmetric Encryption Schemes

We first recall the definition of a symmetric encryption scheme.

Definition 3.3.1. *A* symmetric encryption scheme *is a triple* $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ *of algorithms satisfying the following conditions:*

1. *The* key-generation algorithm $\mathcal{K}$ *is probabilistic polynomial-time. It takes as input the security parameter* 1^τ *and produces as output a string key.*

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

3. *The* decryption algorithm $\mathcal{D}$ *is deterministic polynomial-time. It takes as inputs* 1^τ*, a string key produced by* $\mathcal{K}(1^\tau)$*, and a ciphertext* y*, and produces as output a message* m*.*

We require that for any string key which can be output by $\mathcal{K}(1^\tau)$*, for any message* $m \in \{0,1\}^*$*, and for all* y *that can be output by* $\mathcal{E}(1^\tau, key, m)$*, we have that* $\mathcal{D}(1^\tau, key, y) = m$*.*

Now, we define what we mean by a *secure* symmetric encryption scheme. We consider security with respect to *plaintext indistinguishability*, which is an adaption of the notion of *polynomial security* as given in [44]. We imagine an adversary $\mathtt{A} = (\mathtt{A}_1, \mathtt{A}_2)$ running in two stages. In advance of the adversary's execution, a random key *key* is chosen and kept hidden from the adversary. During the first stage, the adversary $\mathtt{A}_1$ outputs a triple $(x_0, x_1, state)$, where x_0 and x_1 are two messages of the same length, and *state* is some state information which could be useful later. One message between x_0 and x_1 is chosen at random and encrypted to give the challenge ciphertext y. In the second stage, the adversary $\mathtt{A}_2$ is given y and *state* and has to determine whether y is the encryption of x_0 or x_1. Informally speaking, the encryption scheme is said to be secure with respect to a non-adaptive chosen plaintext attack, denoted by `IND-P1-CO` in [53], if every polynomial-time adversary $\mathtt{A}$, which has access to the encryption oracle only during the first stage of the attack and never has access to the decryption oracle, succeeds in determining whether y is the encryption of x_0 or x_1 with probability only negligibly different from $1/2$.

Definition 3.3.2. `[IND-P1-CO]` *Let* $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ *be a symmetric encryption scheme and let* τ *be a security parameter. Let* $\mathtt{A} = (\mathtt{A}_1, \mathtt{A}_2)$ *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 $\mathbf{Exp}^{\mathtt{IND-P1-CO-1}}_{\Pi,\mathtt{A}}(1^\tau)$	Experiment $\mathbf{Exp}^{\mathtt{IND-P1-CO-0}}_{\Pi,\mathtt{A}}(1^\tau)$
$key \leftarrow \mathcal{K}(1^\tau)$	$key \leftarrow \mathcal{K}(1^\tau)$
$(x_0, x_1, state) \leftarrow \mathtt{A}_1^{\mathcal{E}_{key}(\cdot)}(1^\tau)$	$(x_0, x_1, state) \leftarrow \mathtt{A}_1^{\mathcal{E}_{key}(\cdot)}(1^\tau)$
$y \leftarrow \mathcal{E}_{key}(x_1)$	$y \leftarrow \mathcal{E}_{key}(x_0)$
$d \leftarrow \mathtt{A}_2(1^\tau, y, state)$	$d \leftarrow \mathtt{A}_2(1^\tau, y, state)$
return d	**return** d

The advantage of A *is defined as*

$$\begin{aligned}\mathbf{Adv}^{\mathtt{IND-P1-CO}}_{\Pi,\mathtt{A}}(1^\tau) &= |Pr[\mathbf{Exp}^{\mathtt{IND-P1-CO-1}}_{\Pi,\mathtt{A}}(1^\tau)=1] \\ &- Pr[\mathbf{Exp}^{\mathtt{IND-P1-CO-0}}_{\Pi,\mathtt{A}}(1^\tau)=1]|\end{aligned}$$

The scheme is said to be secure *in the sense of* IND-P1-CO *if the advantage function* $\mathbf{Adv}^{\mathtt{IND-P1-CO}}_{\Pi,\mathtt{A}}(1^\tau)$ *is negligible, for any adversary* A *whose time complexity is polynomial in* τ*.*

The XOR Construction for Symmetric Encryption Schemes. In order to construct an encryption scheme secure in the sense of IND-P1-CO we could use a *pseudorandom function family*, an important cryptographic primitive originally defined by Goldreich, Goldwasser, and Micali [43]. Loosely speaking, a distribution of functions is pseudorandom if it satisfies the following requirements: 1) It is easy to sample a function according to the distribution and to evaluate it at a given point; 2) It is hard to tell apart a function sampled according to the distribution from a uniformly distributed function, given access to the function as a block-box. A first construction, based on pseudorandom generators, was proposed in [43]. It is well known that pseudorandom generators can be constructed from one-way functions [18] [47]. The two most efficient constructions were proposed by Naor and Reingold [62]. In particular, they showed a first construction, based on the hardness of factoring Blum integers, and a second one, based on the decisional version of the Diffie-Hellman assumption. In their constructions, the cost of evaluating such functions is comparable to two modular exponentiations.

Consider the following construction, called the *XOR construction* [11], of a symmetric encryption scheme $\Pi_{XOR,\mathcal{F}} = (\mathcal{K}_{XOR}, \mathcal{E}_{XOR}, \mathcal{D}_{XOR})$ which is based on a pseudorandom function family $\mathcal{F} : \{0,1\}^\tau \times \{0,1\}^\tau \to \{0,1\}^\tau$:

- The key generation algorithm $\mathcal{K}_{XOR}$ outputs a random τ-bit key ρ for the pseudorandom function family $\mathcal{F}$, thus specifying a function F_ρ of the family.
- The encryption algorithm $\mathcal{E}_{XOR}$ considers the message x to be encrypted as a sequence of τ-bits blocks $x = x_1 \cdots x_n$ (padding is done on the last block, if necessary), chooses a random string r of τ bits and computes, for

$i = 1, \ldots, n$ the value $y_i = F_\rho(r + i) \oplus x_i$. The ciphertext is $r||y_1 \cdots y_n$, where $||$ denotes string concatenation.

- The decryption algorithm $\mathcal{D}_{XOR}$, on input a ciphertext z, parses it as $r||y_1 \cdots y_n$ and computes, for $i = 1, \ldots, n$ the value $x_i = F_\rho(r + i) \oplus y_i$. The corresponding plaintext is $x = x_1 \cdots x_n$.

The encryption scheme $\Pi_{XOR,\mathcal{F}}$ has been shown to be secure in the sense of `IND-P1-CO` (see [11, 53]), assuming that $\mathcal{F}$ is a pseudorandom function family. An efficient implementation of such a scheme could be obtained by using the HMAC [10] to realize the pseudorandom function family $\mathcal{F}$.

3.3.2 The Two-Levels Encryption-Based Construction (TLEBC)

The construction we are going to describe uses a *graph transformation*, starting from the graph $G = (V, E)$. The output of such a transformation is a *two-levels graph* $G_{TL} = (V_{TL}, E_{TL})$, where $V_{TL} = V^\ell \cup V^r$ and $V^\ell \cap V^r = \emptyset$, constructed as follows:

- for each class $u \in V$, we place two classes u^ℓ and u^r in V_{TL}, where $u^\ell \in V^\ell$ and $u^r \in V^r$;
- for each class $u \in V$, we place the edge (u^ℓ, u^r) in E_{TL};
- for each pair of classes u and v connected by a path in G, we place the edge (u^ℓ, v^r) in E_{TL}.

Thus, we consider a two-level partially ordered hierarchy, where each level contains the same number of classes and there are no edges between classes at the same level. We remark that this is not a restriction, since any directed graph representing an access control policy can be transformed into a two-level partially ordered hierarchy having the above features, using a technique proposed in [34]. Figure 3.1 shows an example of the graph transformation described above.

Let Γ be a family of graphs corresponding to partially ordered hierarchies, let $G = (V, E) \in \Gamma$, and let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme. The

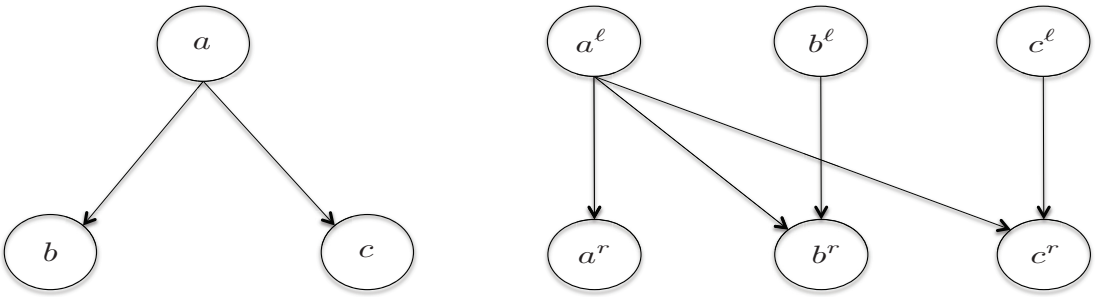


Figure 3.1: The graph transformation used in our construction.

Two-Levels Encryption-Based Construction (TLEBC) works as explained in the following.

Algorithm 1

procedure $Gen(1^\tau, G)$
  Transform the graph G into the two-levels graph $G_{TL} = (V_{TL}, E_{TL})$
  for all classes $u^\ell \in V^\ell$ **do**
    $s_u \leftarrow \mathcal{K}(1^\tau)$
  end for
  for all classes $u^r \in V^r$ **do**
    $k_u \leftarrow \{0,1\}^\tau$
  end for
  ▷ Let s and k be the sequences of private information and keys computed above
  for any pair of classes $u^\ell \in V^\ell$ and $v^r \in V^r$ such that $(u^\ell, v^r) \in E_{TL}$ **do**
    $p_{(u,v)} \leftarrow \mathcal{E}_{s_u}(k_v)$
  end for
  ▷ Let *pub* be the sequence of public information computed above
  return (s, k, pub)
end procedure

Before describing the algorithm *Upd* we recall that the update types we consider are the following: *key replacement*, *insertion of an edge*, *deletion of an edge*, *insertion of a class*, *deletion of a class*, and *revocation of a user from a class*. Each update type requires a different procedure. Notice that some updates do not require the sequence of additional parameters *params*, whereas, for other updates, such a sequence might be not empty. In particular, the *insertion of an edge* and the *deletion of a class* do not require additional parameters, since they

Algorithm 2

procedure $Der(1^\tau, G, u, v, s_u, pub)$
    Extract the public value $p_{(u,v)}$ from pub
    $k_v \leftarrow \mathcal{D}_{s_u}(p_{(u,v)})$
    return k_v
end procedure

do not involve the choice of fresh private information or keys.

As we will see, no update, with the exception of user revocation, requires re-distribution of the secret information to classes. Thus, in the TLEBC, dynamic changes to the hierarchy can be accomplished by means of only local updates to the public information only.

Algorithm 3

procedure $Upd(1^\tau, G, s, k, pub, up, params)$
Transform G in the two-levels graph $G_{TL} = (V_{TL}, E_{TL})$
    if $up ==$ **Replace**(k_v) **then**
        $replace\_key(1^\tau, G, s, k, pub, k_v, params)$
    else if $up ==$ **Insert_edge**$((u, v))$ **then**
        $insert\_edge(1^\tau, G, s, k, pub, (u, v))$
    else if $up ==$ **Delete_edge**$((u, v))$ **then**
        $delete\_edge(1^\tau, G, s, k, pub, (u, v), params)$
    else if $up ==$ **Insert_class**(v) **then**
        $insert\_class(1^\tau, G, s, k, pub, v, params)$
    else if $up ==$ **Delete_class**(v) **then**
        $delete\_class(1^\tau, G, s, k, pub, v)$
    else if $up ==$ **Revoke**(v, λ) **then**
        $revoke\_user(1^\tau, G, s, k, pub, v, \lambda, params)$
    end if
    return (G', s', k', pub')
end procedure

3.3.2.1 Analysis of the Scheme

In the following we show that the security property of the TLEBC depends on the security property of the underlying encryption scheme. We prove that if the encryption scheme $\Pi = (\mathcal{K}, \mathcal{D}, \mathcal{E})$ is secure in the sense of `IND-P1-C0`, then the

Algorithm 4

procedure $replace\_key(1^\tau, G, s, k, pub, k_v, params)$
  if $params$ is not empty **then**
    Parse $params$ as k_v^{params}
    $k'_v \leftarrow k_v^{params}$
  else
    $k'_v \leftarrow \{0,1\}^\tau$
  end if
  $s' \leftarrow s$
  $k' \leftarrow k$, with k'_v instead of k_v
  for all classes $u^\ell \in V^\ell$ such that $(u^\ell, v^r) \in E_{TL}$ **do**
    Compute $p'_{(u,v)} \leftarrow \mathcal{E}_{s'_u}(k'_v)$ and replace it in pub, obtaining pub'
  end for
end procedure

Algorithm 5

procedure $insert\_edge(1^\tau, G, s, k, pub, (u, v))$
  $k' \leftarrow k$
  $s' \leftarrow s$;
  Compute $p'_{(u,v)} \leftarrow \mathcal{E}_{s'_u}(k'_v)$ and add it to pub, obtaining pub'
end procedure

Algorithm 6

procedure $delete\_edge(1^\tau, G, s, k, pub, (u, v), params)$
  $replace\_key(1^\tau, G, s, k, pub, k_v, params)$
  Remove the public value $p'_{(u,v)}$ from pub'
end procedure

Algorithm 7

procedure $insert\_class(1^\tau, G, s, k, pub, v, params)$
  if $params$ is not empty **then**
    Parse $params$ as s_v^{params} and k_v^{params}
    $s'_v \leftarrow s_v^{params}$
    $k'_v \leftarrow k_v^{params}$
  else
    $s'_v \leftarrow \mathcal{K}(1^\tau)$
    $k'_v \leftarrow \{0,1\}^\tau$
  end if
  Add above values into s and k, obtaining s' and k'
  Compute $p'_{(v,v)} \leftarrow \mathcal{E}_{s'_v}(k'_v)$
  Use aforementioned procedure for adding incoming and outgoing edges from v
  Add all the public values to pub, obtaining pub'
end procedure

Algorithm 8

procedure $delete\_class(1^\tau, G, s, k, pub, v)$
  Use aforementioned procedure for deleting incoming and outgoing edges from v
  Remove s_v and k_v from s and k, obtaining s' and k'
end procedure

Algorithm 9

procedure $revoke\_user(1^\tau, G, s, k, pub, v, \lambda, params)$
  if $params$ is not empty **then**
    parse $params$ as s_v^{params}
    $s'_v \leftarrow s_v^{params}$
  else
    $s'_v \leftarrow \mathcal{K}(1^\tau)$
  end if
  Distribute s'_v to each non-revoked user in the class v
  $s' \leftarrow s$, with s'_v instead of s_v
  for all classes $u^r \in V^r$ such that $(v^\ell, u^r) \in E_{TL}$ **do**
    $replace\_key(1^\tau, G, s, k, pub, k_u, NULL)$
  end for
end procedure

TLEBC is secure in the sense of IND-DYN-AD, respectively.

We first give an informal description of the ideas on which the proof is based. The proof uses two well known techniques: *black-box reductions* [44] and *hybrid arguments* [18]. In general, a black-box reduction is used to show that, given a protocol constructed from a cryptographic primitive, if the protocol can be broken somehow, then also the underlying primitive can be broken. On the other hand, the hybrid argument technique is used to argue that two *probability ensembles*, i.e., two sequences of probability distributions defined over the same probability space, are *computationally indistinguishable*. In this type of proof, one defines a sequence, constituted by a *polynomial number* (in the security parameter) of probability ensembles, also called the *hybrids*, where the extreme hybrids correspond to the two ensembles to be shown indistinguishable. Since the total number of hybrids is polynomial, a non-negligible gap between the extreme hybrids translates into a non-negligible gap between a pair of adjacent hybrids.

In our security proof, the probability ensembles are given by the view of a dynamic adaptive adversary ADAPT which, after making a polynomial number of updating and corrupting queries, attacks a class v^r in the two-levels hierarchy obtained after the t-th update on the initial graph $G = (V, E)$. In particular, such a view contains the public information pub^i associated to the graph G^i, for $i = 1, \ldots, t$, the private information held by the corrupted classes, along with a final value, which corresponds to the key k_h assigned to the chosen class v^r after the t-th update. The two extreme hybrids we consider are characterized in one case by the adversary's view when the public values are generated according to the TLEBC, thus containing encryptions of the key k_h, while in the other case by the adversary's view when *part* of the public values is generated by encrypting a randomly chosen value ρ having the same length as k_h. More precisely, the public values which are modified in the last hybrid correspond to those associated to the edges, say $(v_1^\ell, v^r), (v_2^\ell, v^r), \ldots, (v_m^\ell, v^r)$, entering class v^r in the two-levels hierarchy obtained after the t-th update. Thus, in the last hybrid, the public information is completely independent on the last input of the adversary, i.e., k_h, since the values associated to all edges entering class v^r are computed as encryptions of a randomly chosen value ρ. We define a sequence of $m+1$ hybrids, where each pair of adjacent ones, say the j-th and the $(j+1)$-th hybrid, differ only

in the public value associated to a certain edge entering v^r, say (v^ℓ_{j+1}, v^r), which is equal to the encryption of the key k_h in the latter one and to the encryption of a random value ρ, having the same length as k_h, in the former one. For each pair of adjacent hybrids we show, by means of a black-box reduction, that the corresponding views are computationally indistinguishable by the adversary ADAPT, otherwise we could construct an adversary $\mathtt{A} = (\mathtt{A}_1, \mathtt{A}_2)$ which breaks the security of the symmetric encryption scheme $\Pi = (\mathcal{K}, \mathcal{D}, \mathcal{E})$ in the sense of IND-P1-CO.

The algorithm $\mathtt{A}_1$, on input 1^τ, makes queries to its encryption oracle $\mathcal{E}_{key}(\cdot)$ and outputs a triple $(x_0, x_1, state)$, where $x_0, x_1 \in \{0,1\}^\tau$ and *state* is some state information. One message between x_0 and x_1 is chosen at random and encrypted to give the *challenge* ciphertext y. Then, algorithm $\mathtt{A}_2$ is given y and *state* and has to determine whether y is the encryption of x_0 or x_1. More precisely, the algorithm A, in order to exploit ADAPT's ability in distinguishing between the j-th and the $(j+1)$-th hybrid, has first to prepare the inputs for it. Such an information, with the exception of the value associated to the $(j+1)$-th edge entering class v^r, can be easily constructed by $\mathtt{A}_1$ following the same lines as the *Gen* and *Upd* algorithms in the TLEBC, with an important difference: whenever $\mathtt{A}_1$ has to construct the public information associated to the first j edges entering class v^r, it computes the encryptions, with the appropriate private keys, of the random value x_0. On the other hand, the $(j+1)$-th edge entering class v^r will be assigned the value of the challenge ciphertext y, whereas, all subsequent edges entering class v^r will be encryptions of the value x_1, which plays the role of the key k_h assigned to v^r.

It is important to notice that, due to the dynamic behavior of ADAPT, adversary A has no control on the sequence of updating queries asked by ADAPT, thus, it cannot decide in advance to which class its encryption oracle $\mathcal{E}_{key}(\cdot)$ will be associated. Therefore, A makes its guess at the beginning, by randomly choosing a class whose private information will be implicitly set equal to the unknown *key*, but it might fail. Indeed, if the chosen class does not correspond to the $(j+1)$-th one having an edge entering class v^r, then A cannot continue its simulation and needs to restart itself. On the other hand, if the simulation goes well, A outputs the same output as ADAPT: if ADAPT states that its view corresponds to that in

the j-th experiment, then the adversary $\mathtt{A}$ can be sure that the received challenge y comes from the encryption of the message x_1, otherwise by the encryption of the message x_0. Clearly, the success probability of $\mathtt{A}_1$ is equal to $1/q$, where q denotes the number of choices which can be made by $\mathtt{A}_1$, meaning that, in the worst case, $\mathtt{A}_1$ will need to restart itself q times. Thus, the next result holds.

Theorem 3.3.1. *If the encryption scheme* $\Pi = (\mathcal{K}, \mathcal{D}, \mathcal{E})$ *is secure in the sense of* IND-P1-CO*, then the TLEBC is secure in the sense of* IND-DYN-AD*.*

Proof. Let Γ be a family of graphs corresponding to partially ordered hierarchies. Assume by contradiction that the TLEBC is not secure in the sense of IND-DYN-AD. Thus, there exists a graph $G = (V, E)$ in Γ and a dynamic adaptive adversary $\mathtt{ADAPT} = (\mathtt{ADAPT}_1, \mathtt{ADAPT}_2)$ which is able to distinguish between experiments $\mathbf{Exp}^{\mathtt{IND-DYN-1}}_{\mathtt{ADAPT}}(1^\tau, G)$ and $\mathbf{Exp}^{\mathtt{IND-DYN-0}}_{\mathtt{ADAPT}}(1^\tau, G)$ with non-negligibile advantage. Recall that the only difference between $\mathbf{Exp}^{\mathtt{IND-DYN-1}}_{\mathtt{ADAPT}}$ and $\mathbf{Exp}^{\mathtt{IND-DYN-0}}_{\mathtt{ADAPT}}$ is the last input of ADAPT, which corresponds to a real key assigned by the TLEBC in the former experiment and to a random value chosen in $\{0,1\}^\tau$ in the latter. Thus, while in $\mathbf{Exp}^{\mathtt{IND-DYN-1}}_{\mathtt{ADAPT}}$ the public information is related to the last input of ADAPT, in $\mathbf{Exp}^{\mathtt{IND-DYN-0}}_{\mathtt{ADAPT}}$ it is completely independent on such a value.

Without loss of generality, let $V = \{v_1, \ldots, v_n\}$ and let $k_1, \ldots, k_n$ be the keys assigned to the classes in V^r by algorithm *Gen* of the TLEBC. For any $i \geq 1$, denote by k_{n+i} the i-th key which either has been created using the *insert_class* procedure (see **Algorithm 7**) or has been modified using the *replace_key* procedure (see **Algorithm 4**). We restrict our interest to the two above procedures, since those corresponding to the other types of updates either do not require to choose new keys (see **Algorithm 5** and **Algorithm 8**) or invoke the *replace_key* procedure (see **Algorithm 6** and **Algorithm 9**). Moreover, let $s_1, \ldots, s_n$ be the private information assigned to the classes in V^ℓ by algorithm *Gen* of the TLEBC and, for any $i \geq 1$, denote by s_{n+i} the i-th private information which either has been created using the *insert_class* procedure (see **Algorithm 7**) or has been modified using the *revoke_user* procedure (see **Algorithm 9**). We restrict our interest to the two above procedures, since those corresponding to the other types of updates do not require to choose new private information. For example, consider the following sequence of updates on the two-levels graph in

Figure 3.1, yielding to the graph depicted in Figure 3.2: insert class d and edge (d, c), replace key k_c, insert class e and edge (e, c), replace key k'_c. According to the above defined enumeration, the corresponding sequence of keys is $k_1 = k_a$, $k_2 = k_b$, $k_3 = k_c$, $k_4 = k_d$, $k_5 = k'_c$, $k_6 = k_e$, and $k_7 = k''_c$. On the other hand, the corresponding sequence of private information is $s_1 = s_a$, $s_2 = s_b$, $s_3 = s_c$, $s_4 = s_d$, and $s_5 = s_e$.

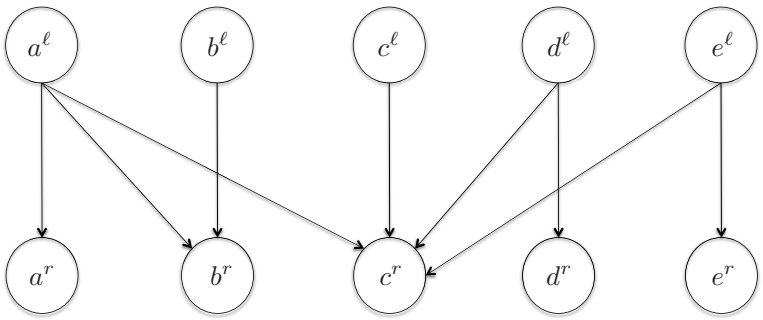


Figure 3.2: Two-levels hierarchy obtained after a list of updates.

Let $q(n, 1^\tau)$ be the running-time of $\texttt{ADAPT}$, where q is a bivariate polynomial. For any $i = 1, \ldots, q(n, 1^\tau)$, let $\texttt{S}_i$ be an adversary which behaves as $\texttt{ADAPT}_1$ until the choice of the key to be attacked. If the chosen key is equal to k_i, then $\texttt{S}_i$ continues to follow $\texttt{ADAPT}_2$, otherwise it outputs 0. The advantage of $\texttt{ADAPT}$ can be written as

$$
\begin{aligned}
&\mathbf{Adv}^{\mathtt{IND-DYN}}_{\mathtt{ADAPT}}(1^\tau, G) \\
&\quad = \; |Pr[\mathbf{Exp}^{\mathtt{IND-DYN-1}}_{\mathtt{ADAPT}}(1^\tau, G) = 1] \\
&\qquad -Pr[\mathbf{Exp}^{\mathtt{IND-DYN-0}}_{\mathtt{ADAPT}}(1^\tau, G) = 1]| \\
&\quad \leq \; \sum_{i=1}^{q(n,1^\tau)} \; | \; Pr[\mathtt{ADAPT}_1 \text{ chooses } k_i] \\
&\qquad \cdot Pr[\mathbf{Exp}^{\mathtt{IND-DYN-1}}_{\mathtt{ADAPT}}(1^\tau, G) = 1 | \mathtt{ADAPT}_1 \text{ chooses } k_i] \\
&\qquad -Pr[\mathtt{ADAPT}_1 \text{ chooses } k_i] \\
&\qquad \cdot Pr[\mathbf{Exp}^{\mathtt{IND-DYN-0}}_{\mathtt{ADAPT}}(1^\tau, G) = 1 | \mathtt{ADAPT}_1 \text{ chooses } k_i] \; | \\
&\quad = \; \sum_{i=1}^{q(n,1^\tau)} Pr[\mathtt{ADAPT}_1 \text{ chooses } k_i] \\
&\qquad \cdot \; | \; Pr[\mathbf{Exp}^{\mathtt{IND-DYN-1}}_{\mathtt{S}_i}(1^\tau, G) = 1] \\
&\qquad -Pr[\mathbf{Exp}^{\mathtt{IND-DYN-0}}_{\mathtt{S}_i}(1^\tau, G) = 1] \; | \\
&\quad = \; \sum_{i=1}^{q(n,1^\tau)} Pr[\mathtt{ADAPT}_1 \text{ chooses } k_i] \cdot \mathbf{Adv}^{\mathtt{IND-DYN}}_{\mathtt{S}_i}(1^\tau, G).
\end{aligned}
$$

Since $\mathbf{Adv}^{\mathtt{IND-DYN}}_{\mathtt{ADAPT}}(1^\tau, G)$ is non-negligible, then there exists at least an index h, where $1 \leq h \leq q(n, 1^\tau)$, such that $\mathbf{Adv}^{\mathtt{IND-DYN}}_{\mathtt{S}_h}(1^\tau, G)$ is non-negligible. Thus, there exists an adversary $\mathtt{S}_h$ which distinguishes between $\mathbf{Exp}^{\mathtt{IND-DYN-1}}_{\mathtt{S}_h}(1^\tau, G)$ and $\mathbf{Exp}^{\mathtt{IND-DYN-0}}_{\mathtt{S}_h}(1^\tau, G)$ with non-negligible advantage. We distinguish the following two cases:

- **Case 1:** $h \geq n + 1$. This case corresponds to the scenario where the key k_h chosen by the adversary either has been created using the *insert_class* procedure (see **Algorithm 7**) or has been modified using the *replace_key* procedure (see **Algorithm 4**).
- **Case 2:** $1 \leq h \leq n$. This case corresponds to the scenario where the key k_h chosen by the adversary has been assigned to some class in the initial graph G.

Analysis of Case 1. Assume that the key k_h chosen by the adversary either has been

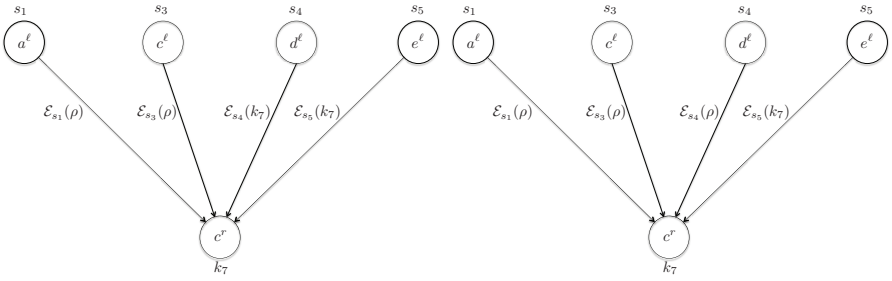
created or has been modified by the t-th update operation, which has assigned such a key to a certain class v in the graph G^t. Thus, attacking the key k_h, corresponds to attack the class v^r in the two-levels hierarchy $G_{TL} = (V_{TL}, E_{TL})$ obtained after the t-th update. Assume that there are m classes which are able to access class v^r in G_{TL}, without loss of generality, let $v_1^\ell, \ldots, v_m^\ell$ be such classes. We construct a sequence of $m + 1$ experiments

$$\mathbf{Exp}_{h,0}, \mathbf{Exp}_{h,1}, \ldots, \mathbf{Exp}_{h,m},$$

all defined over the same probability space. In each experiment we modify the way the view of adversary $\mathtt{S}_h$ is computed, while maintaining the view's distributions indistinguishable among any two consecutive experiments. For any $j = 1, \ldots, m$, experiment $\mathbf{Exp}_{h,j}$ is defined as follows:

Experiment $\mathbf{Exp}_{h,j}(1^\tau, G)$
  $(s, k, pub) \leftarrow Gen(1^\tau, G)$
  $(t, v, history) \leftarrow \mathtt{S}_h^{\mathcal{U}^{i,j}(\cdot,\cdot),\mathcal{C}^i(\cdot)}(1^\tau, G, pub)$
  $d \leftarrow \mathtt{S}_h^{\mathcal{C}^i(\cdot)}(1^\tau, t, v, history, k_h)$
  return d

In experiment $\mathbf{Exp}_{h,j}$ we first use the algorithm Gen of the TLEBC to assign private information and keys to the classes, as well as public information to the edges of the two-levels hierarchy. Then, in the first stage of the attack, the updating oracle queried by adversary $\mathtt{S}_h$ uses an algorithm Upd^j, which is a modification of the algorithm Upd used in the TLEBC. We remark that for this reason the updating oracle is denoted by $\mathcal{U}^{i,j}(\cdot,\cdot)$ in the experiment. The algorithm Upd^j differs from Upd for the way it computes the public information associated to the first j edges, say $(v_1^\ell, v^r), (v_2^\ell, v^r), \ldots, (v_j^\ell, v^r)$, entering class v^r in the two-levels hierarchy. In particular, the public values associated to such edges are computed as encryptions of a value ρ randomly chosen in $\{0,1\}^\tau$, instead of the encryption of the key k_h assigned to v^r, whereas, the public values associated to subsquent edges entering v^r are not modified. Notice that experiment $\mathbf{Exp}_{h,0}$ is the same as $\mathbf{Exp}_{\mathtt{S}_h}^{\mathtt{IND-DYN-1}}$. Indeed, the public information is related to the last input of $\mathtt{S}_h$, since the values associated to all edges entering class v^r are computed as encryptions of k_h. On the other hand, experiment $\mathbf{Exp}_{h,m}$ is the



(a) $\mathbf{Exp}_{7,2}$ (b) $\mathbf{Exp}_{7,3}$

Figure 3.3: Two adjacent experiments.

same as $\mathbf{Exp}_{\mathtt{S}_h}^{\mathtt{IND-DYN-O}}$. In fact, the public information is completely independent on the last input of $\mathtt{S}_h$, since the values associated to all edges entering class v^r are computed as encryptions of a random value ρ chosen in $\{0,1\}^\tau$. Figure 3.3 shows two adjacent experiments in the sequence $\mathbf{Exp}_{7,0}, \mathbf{Exp}_{7,1}, \ldots, \mathbf{Exp}_{7,4}$ of five experiments obtained when attacking the key k_7 in Figure 3.2.

Indistinguishability of any pair of adjacent experiments. In the following we show that, for any $j = 0, \ldots, m-1$, experiments $\mathbf{Exp}_{h,j}$ and $\mathbf{Exp}_{h,j+1}$ cannot be distinguished with non-negligible advantage.

Assume by contradiction that there exists a polynomial-time distinguisher $\mathtt{B}_j$ which is able to distinguish between the adversary $\mathtt{S}_h$'s views in experiments $\mathbf{Exp}_{h,j}$ and $\mathbf{Exp}_{h,j+1}$ with non-negligible advantage. Notice that the views of the distinguisher $\mathtt{B}_j$ in such two experiments differ only for the public value associated to the edge (v_{j+1}^ℓ, v^r), which is equal to the encryption of the real key k_h in the latter experiment and to the encryption of a random value having the same length as the real key in the former experiment. We show how to construct a polynomial-time adversary $\mathtt{A} = (\mathtt{A}_1, \mathtt{A}_2)$, using adversary $\mathtt{B}_j$, which breaks the security of the encryption scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ in the sense of `IND-P1-CO`.

Description of Algorithm $\mathtt{A}_1^{\mathcal{E}_{key}(\cdot)}(1^\tau)$.

1. First, $\mathtt{A}_1$ randomly chooses an index $1 \leq \gamma \leq q(n, 1^\tau)$, such that the private information s_γ will be implicitly set equal to the unknown *key* for its encryption oracle $\mathcal{E}_{key}(\cdot)$.

2. Afterwards, $\mathtt{A}_1$ simulates algorithm $Gen(1^\tau, G)$ as follows:

(a) If $1 \leq \gamma \leq n$, it constructs the private information s_β for all $\beta \in \{1, \ldots, n\} \setminus \{\gamma\}$, as well as all encryption keys $k_1, \ldots, k_n$. On the other hand, if $\gamma \geq n+1$, it constructs all private information $s_1, \ldots, s_n$ as well as all encryption keys $k_1, \ldots, k_n$.

(b) Then, it constructs the sequence *pub* of public values associated to all edges in the two-levels hierarchy obtained from G. In particular, if $1 \leq \gamma \leq n$, $\mathtt{A}_1$ makes use of its encryption oracle $\mathcal{E}_{key}(\cdot)$ for all edges outgoing class v_γ^ℓ.

3. Moreover, $\mathtt{A}_1$ chooses at random two messages $x_0, x_1 \in \{0,1\}^\tau$: in particular, x_1 will play the role of the key k_h, assigned to class v^r, in the next steps.

4. Then, $\mathtt{A}_1$ calls adversary $\mathtt{B}_j$ on inputs 1^τ, G and *pub*. In the first stage of the attack, adversary $\mathtt{B}_j$ can make a polynomial number of *updating* and *corrupting queries*. In particular, the t-th updating query of $\mathtt{B}_j$ consists of a pair $(up^{t+1}, params^{t+1})$, where up^{t+1} is an update operation on the graph G^t and $params^{t+1}$ is a sequence of parameters associated to the update. On the other hand, the t-th corrupting query consists of a class c in the graph G^t.

   (a) Each *updating query* can be answered by adversary $\mathtt{A}_1$ by means of a simulation of a modified version of the algorithm Upd, which corresponds either to Upd^j or to Upd^{j+1}. We recall that the algorithm Upd^j differs from Upd only for the way it computes the public information associated to the first j edges entering class v^r in the two-levels hierarchy. More precisely, the algorithm used by $\mathtt{A}_1$ to answer updating queries, and in particular, to compute the public information associated to each edge (v_i^ℓ, v^r), works as follows:

      i. For each $i = 1, \ldots, j$, $\mathtt{A}_1$ computes $p'_{(v_i,v)}$ as the encryption, with the current private information assigned to class v_i^ℓ, of the random value x_0.

      ii. If v_{j+1}^ℓ does not correspond to the γ-th class whose private information has been either inserted or modified, where γ is the index chosen at step 1, then $\mathtt{A}_1$ restarts from the beginning; otherwise,

it leaves to $\mathtt{A}_2$ the task of associating the challenge y to the edge (v_{j+1}^{ℓ}, v^r).

iii. For each $i = j+2, \ldots, m$, $\mathtt{A}_2$ computes $p'_{(v_i,v)}$ as the encryption, with the current private information assigned to class v_i^{ℓ}, of the value x_1.

On the other hand, the public information associated to each other edge is computed as follows:

i. For each edge (v_{j+1}^{ℓ}, z^r), where $z \neq v$, if v_{j+1}^{ℓ} does not correspond to the γ-th class whose private information has been either inserted or modified, where γ is the index chosen at step 1, then $\mathtt{A}_1$ restarts from the beginning; otherwise, it uses its encryption oracle $\mathcal{E}_{key}(\cdot)$ to compute $p'_{(v_{j+1},z)}$ as the encryption, with the unknown key, of the current key assigned to class z^r.

ii. For each edge (w^{ℓ}, z^r), where $w^{\ell} \neq v_{j+1}^{\ell}$ and $z \neq v$, $\mathtt{A}_1$ computes the public value $p'_{(w,z)}$ as the encryption, with the current private information assigned to class w^{ℓ}, of the current key assigned to class z^r.

(b) The t-th *corrupting query*, corresponding to a class c in G^t can be answered by adversary $\mathtt{A}_1$ with the sequence of private information held by the corrupted class c in all graphs up to G^t, if c belongs to them. We remark that $\mathtt{A}_1$ is able to respond to such a query since in step 1. it has generated the sequence of all private information $s_1, \ldots, s_n$, with the exception of s_γ, corresponding to the unknown key, if $1 \leq \gamma \leq n$; moreover, all subsequent private information $s_{n+1}, \ldots, s_{q(1^\tau,n)}$ has been generated by $\mathtt{A}_1$ when answering updating queries involving class insertions or user revocations.

(c) Upon finishing its updating and corrupting queries, adversary $\mathtt{B}_j$ outputs the triple $(v^r, t, history^-)$, where $history^-$ contains the following information:

- The initial graph G along with all updated graphs $G^1, \ldots, G^t$.
- The sequence of updating operations $up^1, \ldots, up^t$ queried by $\mathtt{B}_j$.

- The corresponding sequences of public information obtained by the interaction with $\mathtt{A}_1$ as a response for the updating queries, with the exception of the public value associated to the edge (v_{j+1}^{ℓ}, v^r).
- The corresponding sequences of keys $old\_k^0, \ldots, old\_k^{t-1}$, which have been modified according to each update. Notice that such sequences have also been obtained by the interaction with $\mathtt{A}_1$, which has generated $k_1, \ldots, k_n$ in step 2.(a), and all subsequent keys in response to *insert_class* and *replace_key* queries.
- The private information held by all corrupted classes, obtained by the interaction with $\mathtt{A}_1$ as a response for the corrupting queries.

5. Finally, $\mathtt{A}_1$ outputs the triple $(x_0, x_1, state)$, where *state* contains the triple $(v^r, t, history^-)$ output by $\mathtt{B}_j$.

Notice that, in step 4.(a), algorithm $\mathtt{A}_1$ needs to restart itself from the beginning in case $j + 1 \neq \gamma$, where γ is the index chosen at step 1. This happens because $\mathtt{A}_1$ has no control on the sequence of updating queries asked by adversary $\mathtt{B}_j$, that is, it cannot decide in advance to which class the encryption oracle $\mathcal{E}_{key}(\cdot)$ will be associated. Therefore, $\mathtt{A}_1$ makes its guess at the beginning, by randomly choosing an index $\gamma \in \{1, \ldots, q(1^\tau, n)\}$, but it might fail. Clearly, the success probability of $\mathtt{A}_1$ is equal to $1/q(1^\tau, n)$, meaning that, in the worst case, $\mathtt{A}_1$ will need to restart itself $q(1^\tau, n)$ times.

Description of Algorithm $\mathtt{A}_2(1^\tau, y, state)$.

1. First, $\mathtt{A}_2$ parses *state* in order to obtain the sequence $history^-$.

2. Then, $\mathtt{A}_2$ adds the missing value y in the sequence $history^-$, associating it to the edge (v_{j+1}^{ℓ}, v^r). The updated sequence of public information is denoted by *history*.

3. Afterwards, $\mathtt{A}_2$ calls adversary $\mathtt{B}_j$ on inputs 1^τ, t, v^r, *history*, and x_1.

4. Finally, $\mathtt{A}_2$ outputs the same output as $\mathtt{B}_j$.

Notice that, if y corresponds to the encryption of x_1, then the random variable associated with the adversary's view is exactly the same as the one associated

with the adversary view in experiment $\mathbf{Exp}_{h,j+1}$, whereas, if y corresponds to the encryption of x_0, it has the same distribution as the one associated with the adversary's view in experiment $\mathbf{Exp}_{h,j}$. Thus, if adversary $\mathtt{B}_j$ is able to distinguish between such two views with non-negligible advantage, it follows that adversary $\mathtt{A}$ is able to break the security of the encryption scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ in the sense of IND-P1-C0. Contradiction.

Hence, for any $j = 0, \ldots, m-1$, experiments $\mathbf{Exp}_{h,j}$ and $\mathbf{Exp}_{h,j+1}$ cannot be distinguished with non-negligible advantage. It follows that experiments $\mathbf{Exp}_{h,0}$ and $\mathbf{Exp}_{h,m}$ cannot be distinguished with non-negligible advantage, for any $h = n+1, \ldots, q(n, 1^\tau)$. Therefore, no adversary $\mathtt{S}_h$, for $h = n+1, \ldots, q(n, 1^\tau)$, distinguishes between $\mathbf{Exp}_{\mathtt{S}_h}^{\mathtt{IND-DYN-1}}(1^\tau, G)$ and $\mathbf{Exp}_{\mathtt{S}_h}^{\mathtt{IND-DYN-0}}(1^\tau, G)$ with non-negligible advantage.

Analysis of Case 2. As done for **Case 1**, we can show that no adversary $\mathtt{S}_h$, where $1 \leq h \leq n$, distinguishes between $\mathbf{Exp}_{\mathtt{S}_h}^{\mathtt{IND-DYN-1}}(1^\tau, G)$ and $\mathbf{Exp}_{\mathtt{S}_h}^{\mathtt{IND-DYN-0}}(1^\tau, G)$ with non-negligible advantage. The proof is similar to that for **Case 1** and uses a sequence of $m+1$ slightly different experiments

$$\widehat{\mathbf{Exp}}_{h,0}, \widehat{\mathbf{Exp}}_{h,1}, \ldots, \widehat{\mathbf{Exp}}_{h,m},$$

where m denotes the number of edges entering class v^r. More precisely, let $\mu < m$ be the number of edges entering class v^r before the first updating operation; for each $j = 1, \ldots, \mu$, experiment $\widehat{\mathbf{Exp}}_{h,j}$ uses an algorithm Gen^j, which is a modified version of the algorithm Gen used in the TLEBC. Such an algorithm differs from Gen for the way it computes the public information associated to the first j edges entering class v^r in the two-levels hierarchy. On the other hand, for each $\mu + 1 \leq j \leq m$, experiment $\widehat{\mathbf{Exp}}_{h,j}$ first uses the algorithm Gen^μ to compute the public information associated to the first μ edges entering v^r, then uses the algorithm Upd^j, described when analyzing **Case 1**, in order to compute the public information associated to edges from $\mu + 1$ to j. As done before, we can show that experiments $\widehat{\mathbf{Exp}}_{h,0}$ and $\widehat{\mathbf{Exp}}_{h,m}$ cannot be distinguished with non-negligible advantage, for any $h = 1, \ldots, n$. Therefore, no adversary $\mathtt{S}_h$, for $h = 1, \ldots, n$, distinguishes between $\mathbf{Exp}_{\mathtt{S}_h}^{\mathtt{IND-DYN-1}}(1^\tau, G)$ and $\mathbf{Exp}_{\mathtt{S}_h}^{\mathtt{IND-DYN-0}}(1^\tau, G)$ with non-negligible advantage.

To conclude, we have proven that no adversary $\mathtt{S}_h$, for $h = 1, \ldots, q(n, 1^\tau)$, distinguishes between $\mathbf{Exp}_{\mathtt{S}_h}^{\mathtt{IND-DYN-1}}(1^\tau, G)$ and $\mathbf{Exp}_{\mathtt{S}_h}^{\mathtt{IND-DYN-0}}(1^\tau, G)$ with non-negligible advantage. Therefore, no dynamic adaptive adversary ADAPT has non-negligible advantage in distinguishing between experiments $\mathbf{Exp}_{\mathtt{ADAPT}}^{\mathtt{IND-DYN-1}}(1^\tau, G)$ and $\mathbf{Exp}_{\mathtt{ADAPT}}^{\mathtt{IND-DYN-0}}(1^\tau, G)$. Thus, the TLEBC is secure in the sense of IND-DYN-AD. □

Chapter 4

Hierarchical and Shared Access Control

"Indistinguishable things are identical (or should be considered as identical)."
— G. W. Leibniz, *1646-1714*

4.1 Introduction

Besides the conventional hierarchical access, sometimes it is necessary to provide some particular sets of users, having specific access credentials, with access to the key of a certain security class. This novel access control model finds a natural field of application even when there is the need to manage unusual, exceptional or emergency situations, which in general require special permissions. In particular, consider the case in which the trust is based on a single entity, let it be a person or an organization. Obviously, this may lead to abuses or violations by such entity, as in the Snowden event [69], where a great deal of confidential information held by the U.S. *National Security Agency* (*NSA*) was stolen. However, the NSA itself has defined in the past some strict guidelines for limiting such abuses, namely, the *Orange Book* [63] and *Two-Person Authorization* [17] [39], whose main goal was to prevent a single user from viewing top-secret documents. The concept upon which the guidelines are based is that, in general, somebody is less inclined

to do something dishonest if someone else is watching. In addition, the two guidelines clearly state that the information within a system must be organized in a "*compartmental manner*", providing different levels of access and security to each compartment. In this case, a simple protection may be to use two or more "locks" to protect a given resource or activity, where each lock needs a different key, owned by a different person. Thus, two or more people are needed in order to grant the access to that resource or activity.

The Snowden event highlights the fact that the collaboration among several users and organizations is preferable for gaining the permission to carry out a given task or to access sensitive information. Such collaboration is needed so as to ensure that the requested permission has been granted through the acceptance and agreement among all the involved entities, thus preventing users from any kind of abuse. In general, the collaboration characterizes scenarios where more than one entity is required to achieve a specific authorization. More precisely, there are many real-world scenarios in which such a collaborative access is necessary, i.e., where a user might have a sort of "pre-authorization" for the access, but he may need to get the approval from someone else. For example, consider the healthcare environment, which typically consists of several professional profiles, such as doctors, nurses, etc.. In this environment, nurses may access a subset of stored patient's clinical data, while a doctor can usually access all the data. However, it is important to emphasize that the doctor and nurse must have the patient's consent to access clinical information. In addition, a nurse should not access all the information concerning a patient, unless she does not gain the permission from both patient and doctor. Moreover, if a doctor wants to access some clinical data without the explicit consent of the patient, he should be granted permission from several entities, e.g., hospital administration, medical committee, government authority, etc..

Again, the access to the workspace of a specific project branch could be granted either directly to the project manager or to a set of project team members. The same arguments apply to distributed cryptographic file systems [22]. A further real field of application lies in the collaborative access to logs concerning accesses and events, where the access can be achieved either by a single entity (e.g., a communications authority) or by more of them, which cooperate with

each other. For example, the access might be allowed only if the judicial authority cooperates with a given service provider. Another concrete example arises from the military field, in which a decision can be taken by a single person with a specific rank, by a certain number of his subordinates or more generally, by a given number of people with certain credentials, which do not have the authority to decide on their own. Furthermore, consider a committee board composed of several members and a general chair. In this context, the chair might be away for personal reasons or could be in a situation which prevents him from making any decisions for a given action. Only one member of the board cannot independently take such a decision on behalf of the chair. However, the board members can collectively take such a decision on behalf of the chair, as long as their number is greater than or equal to a certain threshold.

All the aforementioned considerations and examples bring to light the fact that hierarchical and shared key assignment schemes are required in many cases. A *hierarchical and shared key assignment scheme* (HSKAS) should assign an encryption key and some private information to each class in the system in such a way that the private information of a group of qualified users is jointly required in order to compute the key assigned to a class lower down in the hierarchy.

In this chapter we first propose and formalize a novel access control model which prevents the abuse of permissions, defines alternative methods for gaining such permissions and allows the separation of duties. The model also enables collaboration among a set of users to gain specific permissions, defining the way in which such collaboration takes place. Furthermore, we formalize the notion of key indistinguishability regarding a new access model. Again, we propose two constructions which implement such a novel access control model. In particular, our first proposal, denoted as the *Shared Encryption Based Construction* (*SEBC*), uses as its basic building blocks a *symmetric encryption scheme* and a *perfect secret sharing scheme*, and offers security with respect to key indistinguishability. Our second proposal, denoted as the *Threshold Broadcast Encryption Based Construction (TBEBC)*, is based on a public-key *threshold broadcast encryption scheme* and provides security against key indistinguishability.

This chapter is organized as follows. In Section 4.2 we formally define the model we propose, by focusing on its specific security properties. In Section 4.3

we provide our constructions for hierarchical and shared key assignment, along with the relative security proofs.

4.2 The Model

There are several practical situations in which collaboration from users belonging to different security classes is needed, in order to access sensitive data belonging to a certain class lower down in the hierarchy. For example, a user could be part of multiple groups at the same time, and each of them could be associated to different access permissions. On the other hand, even the resource being accessed may require different access policies, according to those who access it. This problem can be solved by using a *hierarchical and shared access control*, where collaboration between classes is required to make sure that the access right to the private data held by a class v has been granted with the agreement of some particular users.

All of these situations and usage scenarios can be modeled by means of a *directed multigraph*, namely, a directed graph that can have more than one edge between the same pair of vertices. The presence of an edge e connecting a class u to a class v means that class u is involved in a shared access control for class v's data. Moreover, since each class can obviously access the resources held by itself, the multigraph also contains "self-loops". However, from now on, for the sake of simplicity, we will not mention of such type of loops. Formally, a *directed multigraph* is a triple $G = (V, E, \phi)$, where V is a set of vertices, E is a set of edges, and ϕ is a function which associates each edge in E to its endpoints, that is: $\phi : E \to V \times V$. The edges e_1 and e_2 are said to be *multiple or parallel* if it holds that $\phi(e_1) = \phi(e_2)$. For example, consider the directed multigraph $G = (V, E, \phi)$ depicted in Figure 4.1, where $V = \{a, b, c, d, e, f, g, h, i, l\}$, $E = \{e_1, e_2, \cdots, e_{18}\}$ and ϕ is the function defined in Table 4.1. Multiple edges are represented by dashed lines. In general, a subgraph $H = (V', E', \phi')$ of a multigraph $G = (V, E, \phi)$ is a multigraph whose underlying graph is a subgraph of that of G and its function ϕ' is dominated by ϕ, that is, the multiplicity of an edge in H does not exceed its multiplicity in G.

Why do we need multiple edges to model hierarchical and shared access con-

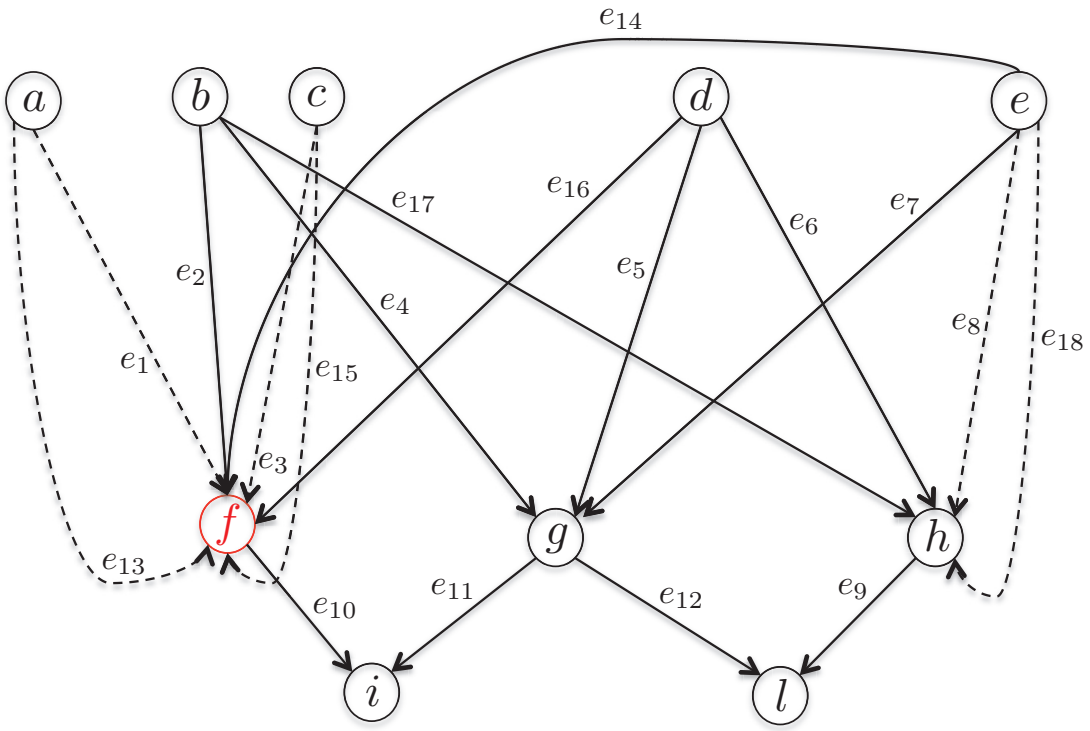


Figure 4.1: Example of a directed multigraph characterizing our novel access control model.

trol? The reason is that, in our new model, each class might be associated to different access structures, corresponding to different access rights/permissions. Given a directed multigraph $G = (V, E, \phi)$, for each class $v \in V$, let $\mathcal{I}_v \subset E$ be the set of edges ending in v. In this chapter we consider the general situation where *more than one access structure* can be associated to each class in the multigraph. More precisely, for any $v \in V$, let $m_v \geq 1$ be an integer, let $\mathcal{P}_v^1, \ldots, \mathcal{P}_v^{m_v}$ be m_v subsets of $\mathcal{I}_v$, and let $\mathcal{A}_v^1, \ldots, \mathcal{A}_v^{m_v}$ be m_v access structures for class v on the sets $\mathcal{P}_v^1, \ldots, \mathcal{P}_v^{m_v}$, respectively. For each $v \in V$, let $\mathcal{A}_v = \{\mathcal{A}_v^1, \ldots, \mathcal{A}_v^{m_v}\}$ be the family

Table 4.1: Function ϕ of the directed multigraph shown in Figure 4.1

$\boldsymbol{e}$	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9
$\phi(\boldsymbol{e})$	(a, f)	(b, f)	(c, f)	(b, g)	(d, g)	(g, h)	(e, g)	(e, h)	(h, l)
$\boldsymbol{e}$	e_{10}	e_{11}	e_{12}	e_{13}	e_{14}	e_{15}	e_{16}	e_{17}	e_{18}
$\phi(\boldsymbol{e})$	(f, i)	(g, i)	(g, l)	(a, f)	(e, f)	(c, f)	(d, f)	(b, h)	(e, h)

of all access structures associated to class v and let $\mathcal{A}_G = \{\mathcal{A}_v\}_{v \in V}$. Notice that, if a hierarchical, but not shared, access control for a class v is required, we can simply associate to the class v a single access structure containing all the edges in $\mathcal{I}_v$, thus requiring no collaboration among classes in order to access v's data. In this case, two parallel edges make no difference, since they represent the same access right.

The proposed access model finds natural application especially in the so-called "*multi-domain environments*", namely, in all those environments in which there are different cooperating entities, each of them with different interests, responsibilities and tasks to perform. It is important to emphasize that a particular entity, depending on the context and the role which it assumes, may have different roles towards another given entity to which it intends to be granted access. Informally speaking, an entity can take on several tasks, and according to the assumed role it may have different access rights. On the other hand, a certain security class can provide the same entity with different access rights, according to the tasks performed by the latter in that particular context at that particular time. Clearly, at a given time an entity may need to take simultaneously all of its different tasks. In detail, an entity may assume one or more roles, each belonging to the characterization of a specific access structure, for a certain security class.

4.2.1 A Motivating Example

One of the most popular examples of multi-domain environments is obviously the one concerning healthcare. In particular, consider the multigraph shown in Figure 4.1, let f be the security class characterizing all the data related to a specific patient. It is easy to note that the set $\mathcal{I}_f$ is composed of seven elements, namely, $\mathcal{I}_f = \{e_1, e_2, e_3, e_{13}, e_{14}, e_{15}, e_{16}\}$. Without loss of generality, one of the possible characterizations for such set may be the following:

- $Entity_f^a$ (*Health director*) $= \{e_1, e_{13}\}$, where $Role_f^{e_1} = \{$Scientific Manager$\}$ and $Role_f^{e_{13}} = \{$Administrative Manager$\}$;

- $Entity_f^b = \{e_2\}$, where $Role_f^{e_2} = \{$Insurance Company$\}$;

- $Entity_f^c$ $(Doctor) = \{e_3, e_{15}\}$, where $Role_f^{e_3} = \{\text{Specialist}\}$ and $Role_f^{e_{15}} = \{\text{Common Practitioner}\}$;
- $Entity_f^d = \{e_{16}\}$, where $Role_f^{e_{16}} = \{\text{Patient's Family}\}$;
- $Entity_f^e = \{e_{14}\}$, where $Role_f^{e_{14}} = \{\text{Administrative Office}\}$.

Therefore, starting from the $\mathfrak{I}_f$ set, the following three access structures for the class f, that is to say, $\mathcal{A}_f^1$=$\{\{e_1, e_2, e_3\}, \{e_{13}, e_{14}\},\{e_{15}, e_{16}\}\}$, $\mathcal{A}_f^2$=$\{\{e_{13}, e_{16}\}, \{e_2, e_{14}\}, \{e_1, e_3, e_{15}\}\}$ and $\mathcal{A}_f^3$=$\{\{e_2, e_{13}\}, \{e_{15}, e_{16}\}, \{e_1, e_3, e_{14}\}\}$, might be characterized. For example, the first access structure, denoted as $\mathcal{A}_f^1$, could model the scenario in which the patient belonging to the class f is involved in a legal dispute for insurance issues. Instead, the access structure denoted as $\mathcal{A}_f^2$ can model the situation in which the patient intends to participate in the experimentation of a particular drug, therefore he needs to be subjected to some specific investigations and clinical evaluations. Finally, the access structure denoted as $\mathcal{A}_f^3$ can model the case in which the patient finds out to have a specific congenital disease, and for this reason needs to take out a new insurance policy on his health, consequently canceling the old one. Obviously, the three examples of access structure we provide for the class f, along with their relative characterizations, although concrete are extremely trivial. However, as it is easy to observe, the proposed model, because of its generality, is virtually able to represent any set of scenarios which may potentially occur in real life.

4.2.2 Hierarchical and Shared Key Assignment Schemes

Let $G = (V, E, \phi)$ be a directed multigraph and let $\mathcal{A}_G = \{\mathcal{A}_v\}_{v \in V}$ be the family of all access structures associated to the classes in V. For any $X \subseteq V$, we are going to define the set of classes A_X which can be accessed when classes in X collaborate together. Such a set will be constructed by using a *Breadth-First-Search (BFS)* on G, starting from the set X.

More precisely, starting from X, we will explore the multigraph G outgoing from X in all possible directions, adding classes one *layer* at a time, according to the access structures associated to the classes. First, notice that $X \in A_X$, since each class in X is authorized to access itself, with no need of cooperation.

In particular, we denote by $A_X^0 = X$ the set of classes added at this step, also called *zero level of cooperation*. Then, we start from X and include all the classes $u \in V$ for which there exists a subset $Y \subseteq X$ such that the set E_Y of edges whose source classes are in Y belongs to *at least one* of the access structures associated to u. This is the *first layer of cooperation*, and the corresponding set of classes is denoted by A_X^1. We then include all the additional classes $u \in V$ for which there exists a subset $Y \subseteq A_X^1$ such that the set E_Y of edges whose source classes are in Y belongs to *at least one* of the access structures associated to u. This is the *second layer of cooperation*, and the corresponding set of classes is denoted by A_X^2. We continue in this way until there are no more layers to be explored, i.e., until the set $A_X^{diam(G)}$ has been constructed, where $diam(G)$ denotes the diameter of G. The set of classes A_X which can be accessed when classes in X collaborate together is naturally defined to be the union of the sets constructed in the different layers.

Formally, for each set of classes $X \subseteq V$ and for each level $j = 0, \ldots, diam(G)$, we define the set A_X^j corresponding to the set of classes which can be accessed by X at the j-th *level of cooperation*, as follows:

- $A_X^0 = X$;
- $A_X^j = \{v \in V : \exists Y \subseteq A_X^{j-1} \text{ s. t. } E_Y \in \mathcal{A}_v^i, \text{ for some } i \in \{1, \ldots, m_v\}\}$.

The set of classes A_X which can be accessed when classes in X collaborate together is defined to be

$$A_X = \cup_{j=0}^{diam(G)} A_X^j.$$

Consider the multigraph depicted in Figure 4.1 and let $\mathcal{A}_f^1$=$\{\{e_1, e_2, e_3\}, \{e_{13}, e_{14}\},\{e_{15}, e_{16}\}\}$, $\mathcal{A}_f^2$=$\{\{e_{13}, e_{16}\}, \{e_2, e_{14}\}, \{e_1, e_3, e_{15}\}\}$ and $\mathcal{A}_f^3$=$\{\{e_2, e_{13}\}, \{e_{15}, e_{16}\}, \{e_1, e_3, e_{14}\}\}$ be three access structures associated to class f. Moreover, let $\mathcal{A}_g = \{\{e_4, e_5, e_7\}\}$, $\mathcal{A}_h = \{\{e_6, e_7\}, \{e_{17}, e_{18}\}\}$, $\mathcal{A}_i = \{\{e_{10}, e_{11}\}\}$, and $\mathcal{A}_l = \{\{e_9, e_{12}\}\}$ be the access structures associated to classes g, h, i, and l, respectively. Let $X = \{b, c, d, e\}$. It is easy to see that $A_X = \{b, c, d, e, f, g, h, i, l\}$. Indeed, $A_X^0 = \{b, c, d, e\}$, $A_X^1 = \{f, g, h\}$, and $A_X^2 = \{i, l\}$.

We are now ready to give a formal definition for *hierarchical and shared key assignment schemes*.

Definition 4.2.1. *A* hierarchical and shared key assignment scheme for $(G, \mathcal{A}_G)$ *is a pair* (Gen, Der) *of algorithms satisfying the following conditions:*

1. *The* information generation algorithm *Gen is probabilistic polynomial-time. It takes as inputs the security parameter* 1^τ*, a directed multigraph* $G = (V, E, \phi)$ *and the corresponding set of families of access structures* $\mathcal{A}_G$*, and produces as outputs:*

   (a) *a private information* s_u*, for any class* $u \in V$*;*

   (b) *a key* $k_u \in \{0,1\}^\tau$*, for any class* $u \in V$*;*

   (c) *a public information pub.*

   We denote by (s, k, pub) *the output of the algorithm Gen on inputs* 1^τ*,* G *and* $\mathcal{A}_G$*, where* s *and* k *denote the sequences of private information and of keys, respectively. Moreover, for any* $X \subseteq V$*, we denote by* s_X *the sequence of private information associated to the classes in* X*.*

2. *The* key derivation algorithm *Der is deterministic polynomial-time. It takes as inputs the security parameter* 1^τ*, a directed multigraph* $G = (V, E, \phi)$ *and the corresponding set of families of access structures* $\mathcal{A}_G$*, a set of classes* $X \subseteq V$*, the private information* s_X *held by classes in* X*, a class* $u \in A_X$*, and the public information pub, and produces as output the key* k_u *assigned to a class* u*.*

   We require that for each class $u \in V$*, each set of classes* $X \subseteq V$*, each sequence of private information* s_X *associated to classes in* X*, each class* $u \in A_X$*, each key* k_u*, each public information pub which can be computed by Gen on inputs* 1^τ*,* G *and* $\mathcal{A}_G$*, it holds that* $Der(1^\tau, G, \mathcal{A}_G, X, s_X, u, pub) = k_u$*.*

4.2.3 Evaluation Criteria and Notions of Security

The efficiency of a hierarchical and shared key assignment scheme is evaluated according to several parameters, such as the amount of secret data that needs to be distributed to and stored by users, the amount of public data, the complexity of key derivation, and the resistance to collusive attacks. More precisely, for each

class $u \in V$, the key k_u should be protected against any coalition of users which are not allowed to access such class, even when pooling together their private information.

We consider a *static adversary* $\mathtt{STAT}_{u,X}$ that wants to attack a class $u \in V$ and which is able to corrupt a set of classes X such that $u \notin A_X$. We define an algorithm $Corrupt_u(s, X)$ which, starting from the private information s generated by the algorithm Gen, and a coalition of classes X such that $u \notin A_X$, extracts the sequence of private information s_X associated to the classes in X. Two experiments are considered. In the first one, the adversary is given the key k_u, whereas, in the second one, it is given a random string ρ having the same length as k_u. It is the adversary's job to determine whether the received challenge corresponds to k_u or to a random string. We require that the adversary will succeed with probability only negligibly different from 1/2.

Definition 4.2.2. `[IND-ST]` *Let $G = (V, E, \phi)$ be a directed multigraph and let $\mathcal{A}_G$ be the corresponding set of families of access structures, let (Gen, Der) be a hierarchical and shared key assignment scheme, and let $\mathtt{STAT}_{u,X}$ be a static adversary which attacks a class $u \in V$ and corrupts a set of classes X such that $u \notin A_X$. Consider the following two experiments:*

Experiment $\mathbf{Exp}^{\mathtt{IND-1}}_{\mathtt{STAT}_{u,X}}(1^\tau, G, \mathcal{A}_G)$
$(s, k, pub) \leftarrow Gen(1^\tau, G, \mathcal{A}_G)$
$s_X \leftarrow Corrupt_u(s, X)$
$d \leftarrow \mathtt{STAT}_{u,X}(1^\tau, G, \mathcal{A}_G, pub, s_X, k_u)$
return d

Experiment $\mathbf{Exp}^{\mathtt{IND-0}}_{\mathtt{STAT}_{u,X}}(1^\tau, G, \mathcal{A}_G)$
$(s, k, pub) \leftarrow Gen(1^\tau, G, \mathcal{A}_G)$
$s_X \leftarrow Corrupt_u(s, X)$
$\rho \leftarrow \{0,1\}^\tau$
$d \leftarrow \mathtt{STAT}_{u,X}(1^\tau, G, \mathcal{A}_G, pub, s_X, \rho)$
return d

The advantage of $\mathtt{STAT}_{u,X}$ is defined as

$$\begin{aligned}\mathbf{Adv}^{\mathtt{IND}}_{\mathtt{STAT}_{u,X}}(1^\tau, G, \mathcal{A}_G) &= |Pr[\mathbf{Exp}^{\mathtt{IND-1}}_{\mathtt{STAT}_{u,X}}(1^\tau, G, \mathcal{A}_G) = 1] \\ &- Pr[\mathbf{Exp}^{\mathtt{IND-0}}_{\mathtt{STAT}_{u,X}}(1^\tau, G, \mathcal{A}_G) = 1]|\end{aligned}$$

The scheme is said to be secure in the sense of IND-ST *if, for each directed multigraph* $G = (V, E, \phi)$*, each family of access structures* $\mathcal{A}_G$*, each class* $u \in V$ *and each set of classes* X *such that* $u \notin A_X$*, the function* $\mathbf{Adv}^{\mathtt{IND}}_{\mathtt{STAT}_{u,X}}(1^\tau, G, \mathcal{A}_G)$ *is negligible, for each static adversary* $\mathtt{STAT}_{u,X}$ *whose time complexity is polynomial in* τ*.*

In Definition 4.2.2 we have considered a static adversary attacking a class. A different kind of adversary, the *adaptive* one, could also be considered. Such an adversary is first allowed to access all public information as well as all private information of a number of classes of its choice; afterwards, it chooses the class it wants to attack. However, following the lines of [7], it can be shown that security against adaptive adversaries is (polynomially) equivalent to security against static adversaries. Hence, in this chapter we will only consider static adversaries.

4.3 Constructions

In this section we propose two different constructions for hierarchical and shared key assignment schemes. The former, denoted as the *Shared Encryption Based Construction (SEBC)*, is based on symmetric encryption and perfect secret sharing schemes, whereas, the latter, denoted as the *Threshold Broadcast Encryption Based Construction (TBEBC)*, is based on threshold broadcast encryption schemes. Both the proposed constructions are provably secure with respect to key indistinguishability.

4.3.1 A Construction based on Symmetric Encryption

In the following we consider the problem of constructing a hierarchical and shared key assignment scheme by using as its basic building blocks a symmetric encryption scheme and a perfect secret sharing scheme. Before describing our construction, we first recall the definition of perfect secret sharing schemes. Instead, for what concerns the definition of symmetric encryption schemes, the reader can refer to the Section 3.3.1.

4.3.1.1 Symmetric Encryption Schemes

A *symmetric encryption scheme* is a triple $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ of algorithms satisfying the following conditions:

1. The *key-generation algorithm* $\mathcal{K}$ is probabilistic polynomial-time. It takes as input the security parameter 1^τ and produces as output a string key.

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

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

In the following we define what we mean by a *secure* symmetric encryption scheme. We formalize security with respect to *plaintext indistinguishability*, which is an adaption of the notion of *polynomial security* as given in [44]. We consider an adversary $A = (A_1, A_2)$ running in two stages. In advance of the adversary's execution, a random key (key) is chosen and kept hidden from the adversary. During the first stage, the adversary A_1 outputs a triple $(x_0, x_1, state)$, where x_0 and x_1 are two messages of the same length, and $state$ is some state information which could be useful later. One message between x_0 and x_1 is chosen at random and encrypted to give the challenge ciphertext y. In the second stage, the adversary A_2 is given y and $state$ and has to determine whether y is the encryption of x_0 or x_1. Informally, the encryption scheme is said to be secure with respect to a *non-adaptive chosen plaintext attack*, denoted by `IND-P1-C0` in [53], if every polynomial-time adversary A, which has access to the encryption oracle only during the first stage of the attack and has never access to the decryption oracle, succeeds in determining whether y is the encryption of x_0 or x_1 with probability only negligibly different from $1/2$ (*random guess*).

In an *adaptive chosen plaintext attack* the adversary is also allowed to access the encryption oracle during the second stage of the attack. Notice that security

with respect to such attack has been shown to be equivalent to the one with respect to a non-adaptive chosen plaintext attack in [53], thus in this chapter we will only consider security with respect to IND-P1-CO.

Definition 4.3.1. [IND-P1-CO] *Let* $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ *be a symmetric encryption scheme and let* τ *be a security parameter. Let* $A = (A_1, A_2)$ *be an adversary which has access to the encryption oracle only during the first stage of the attack and has never access to the decryption oracle. Consider the following two experiments:*

Experiment $\mathbf{Exp}^{\mathtt{IND-P1-CO-1}}_{\Pi,A}(1^\tau)$	Experiment $\mathbf{Exp}^{\mathtt{IND-P1-CO-0}}_{\Pi,A}(1^\tau)$
$key \leftarrow \mathcal{K}(1^\tau)$	$key \leftarrow \mathcal{K}(1^\tau)$
$(x_0, x_1, state) \leftarrow A_1^{\mathcal{E}_{key}(\cdot)}(1^\tau)$	$(x_0, x_1, state) \leftarrow A_1^{\mathcal{E}_{key}(\cdot)}(1^\tau)$
$y \leftarrow \mathcal{E}_{key}(x_1)$	$y \leftarrow \mathcal{E}_{key}(x_0)$
$d \leftarrow A_2(1^\tau, y, state)$	$d \leftarrow A_2(1^\tau, y, state)$
return d	**return** d

The advantage of A is defined as

$$\begin{aligned}\mathbf{Adv}^{\mathtt{IND-P1-CO}}_{\Pi,A}(1^\tau) &= |Pr[\mathbf{Exp}^{\mathtt{IND-P1-CO-1}}_{\Pi,A}(1^\tau) = 1] \\ &- Pr[\mathbf{Exp}^{\mathtt{IND-P1-CO-0}}_{\Pi,A}(1^\tau) = 1]|.\end{aligned}$$

The scheme is said to be secure *in the sense of* IND-P1-CO *if the advantage function* $\mathbf{Adv}^{\mathtt{IND-P1-CO}}_{\Pi,A}(1^\tau)$ *is negligible, for any adversary A whose time complexity is polynomial in* τ*.*

4.3.1.2 Perfect Secret Sharing Schemes

A *secret sharing scheme* $\Sigma = (Share, Recover)$ is a pair of algorithms run by a *dealer* and a set $\mathcal{P}$ of n participants. The *Share* algorithm is executed by the dealer who, given a secret s, computes some shares $s_1, \ldots, s_n$ of such secret and gives each participant one share. The shares are computed in such a way that only qualified subsets of participants can reconstruct the value of s, by using the *Recover* algorithm on input their shares, whereas, any other subset of participants, non-qualified to know s, cannot determine anything about the

value of the secret. Secret sharing schemes were introduced by Shamir [65] and Blakley [16] and have found applications in several areas of data security. Shamir and Blakley analyzed the case in which only subsets of participants of cardinality at least h, for a fixed integer $h \leq n$, can reconstruct the secret. These schemes are called (h, n)-threshold schemes. Subsequently, Ito et al. [52] and Benaloh and Leichter [13] described a more general method of secret sharing. They showed how to realize a secret sharing scheme for any *access structure*, where the access structure is the family of all the subsets of participants that are able to reconstruct the secret. The survey by Stinson [67] contains a unified description of results in the area of secret sharing schemes.

In this chapter with a boldface capital letter, say $\mathbf{Y}$, we denote a random variable taking values on a set, denoted by the corresponding capital letter Y, according to some probability distribution $\{Pr_{\mathbf{Y}}(y)\}_{y \in Y}$. The values such a random variable can take are denoted by the corresponding lower case letter. Let S be the set of secrets, $\{Pr_{\mathbf{S}}(s)\}_{s \in S}$ be a probability distribution on S and let a secret sharing scheme for secrets in S be fixed. Assume for the rest of the chapter that $Pr(\mathbf{S} = s) > 0$ for all $s \in S$. Let $\mathcal{P}$ be the set of participants, and for any $P \in \mathcal{P}$, let us denote by $Sh(P)$ the set of all possible shares given to participant P. Given a set of participants $X = \{P_{i_1}, \ldots, P_{i_r}\}$, where $i_1 < i_2 < \cdots < i_r$, let $Sh(X) = Sh(P_{i_1}) \times \cdots \times Sh(P_{i_r})$. Any secret sharing scheme for secrets in S and a probability distribution $\{Pr_{\mathbf{S}}(s)\}_{s \in S}$ naturally induce a probability distribution on $Sh(X)$, for any $X \subseteq \mathcal{P}$. We denote such probability distribution by $\{Pr_{\mathbf{X}}(x)\}_{x \in Sh(X)}$.

In terms of the probability distribution on the secret and on shares given to participants, we say that a secret sharing scheme $\Sigma = (Share, Recover)$ for the access structure $\mathcal{A}$ is *perfect* if the following two conditions hold:

1. *Any subset $X \subseteq \mathcal{P}$ of participants enabled to recover the secret can compute the secret.* Formally, if $X \in \mathcal{A}$, then, for all $x \in Sh(X)$ with $Pr(\mathbf{X} = x) > 0$, a unique secret $s \in S$ exists such that $Pr(\mathbf{S} = s | \mathbf{X} = x) = 1$.

2. *Any subset $X \subseteq \mathcal{P}$ of participants not enabled to recover the secret has no information about the secret.* Formally, if $X \notin \mathcal{A}$, then, for all $s \in S$ and for all $x \in Sh(X)$ with $Pr(\mathbf{X} = x) > 0$, it holds that $Pr(\mathbf{S} = s | \mathbf{X} = x) =$

$Pr(\mathbf{S} = s)$.

In detail, condition 1 means that the value of the shares held by participants in the qualified set X completely determines the secret $s \in S$. Instead, condition 2 means that the probability that the secret is equal to s given that the shares held by participants in the non-qualified set X correspond to the sequence x, is equal to the a priori probability that the secret is s. Therefore, no amount of knowledge of shares of participants not qualified to reconstruct the secret enables a Bayesian opponent to modify an a priori guess regarding which the secret is.

It is well known that, in any perfect secret sharing scheme, the size of the share given to any participant is at least the size of the secret [21]. The sample space of shares given to any group of participants in a perfect secret sharing scheme, as a function of the size of the set of secrets has also been considered [19]. In particular, the authors of [19] proved the following result, which will be useful later.

Remark 4.3.1. *Let $\mathcal{A}$ be an access structure on the set of participants $\mathcal{P}$. In any perfect secret sharing scheme for $\mathcal{A}$ for any $X \notin \mathcal{A}$, it holds that $Pr(\mathbf{X} = x) = 1/|S|$, for any $x \in Sh(X)$.*

Shamir's Threshold Schemes. In the following we recall the (h, n)-threshold scheme proposed by Shamir [65]. Let $q > n$ be a prime number, let $s \in \mathbb{Z}_q$ be the secret to be shared among the n participants and let $h \leq n$ be a fixed threshold. Let $x_1, \ldots, x_n$ be n distinct non-zero elements in $\mathbb{Z}_q$ known to all the parties (since q is a prime, then we can take $x_j = j$). To set up the scheme, the dealer constructs a random polynomial $a(x)$ of degree at most $h - 1$, having coefficients in $\mathbb{Z}_q$, in which the constant term is the secret s. The share for participant P_i is the point (x_i, y_i) of the polynomial $a(x)$.

The correctness and privacy of Shamir's scheme derive from the *Lagrange's interpolation theorem*, which states that for any h distinct values $x_{i_1}, \ldots, x_{i_h}$ and any h values $y_{i_1}, \ldots, y_{i_h}$, there exists a unique polynomial $a'(x)$ of degree at most $h - 1$ over $\mathbb{Z}_q$ such that $a'(x_{i_j}) = y_{i_j}$, for $j = 1, \ldots, h$. To see that Shamir's scheme is correct, notice that every set of participants $\{P_{i_1}, \ldots, P_{i_h}\}$ holds h points $s_{i_1}, \ldots, s_{i_h}$ of the polynomial $a(x)$, hence each set can reconstruct it using

Lagrange's interpolation and compute $s = a(0)$. The set of participants computes

$$a'(x) = \sum_{\ell=1}^{h} s_{i_\ell} \prod_{1 \le j \le h, j \ne \ell} \frac{x_{i_j} - x}{x_{i_j} - x_{i_\ell}}.$$

Notice that $a'(x_{i_\ell}) = s_{i_\ell} = a(x_{i_\ell})$, for $\ell = 1, \ldots, h$. That is to say, $a'(x)$ and $a(x)$ are polynomial of degree at most $h-1$ which agree on h points, thus, by the uniqueness in the interpolation theorem, they are equal, and, in particular, $a'(0) = a(0) = s$.

In fact, they know that $y_{i_j} = a(x_{i_j})$, for $1 \le j \le h$. Since $a(x)$ has degree at most $h-1$, $a(x)$ can be written as $a(x) = a_0 + a_1 x + \cdots + a_{h-1} x^{h-1}$, where $a_0, \ldots, a_{h-1}$ are unknown elements in Z_q and $a_0 = s$ is the secret. Thus, the participants obtain a system of h linear equations in the h unknowns $a_0 \ldots, a_{h-1}$. Such a system can be represented in matrix form as $Aa = y$, where the coefficient matrix A is a Vandermonde matrix, whose determinant can be computed as $det(A) = \prod_{1 \le j < t \le h} (x_{i_h} - x_{i_j}) \bmod q$. Since the x_i's are all distinct, then $det(A) \neq 0$ and it follows that the system has a unique solution over the field Z_q. Therefore, the h participants can reconstruct the whole polynomial $a(x)$ and compute the secret $s = a(0)$.

On the other hand, any $h-1$ participants have no information about the secret s. Proceeding as above, the group of participants obtain a system of $h-1$ equations in h unknowns. Suppose they hypothesize a value s' for the secret. Since the secret is $a(0) = a_0$, this will yield a further equation, and the coefficient matrix of the resulting system of h equations in h unknowns will again be a Vandermonde matrix. As before, there will be a unique solution. Hence, for every hypothesized value s' of the secret, there is a unique polynomial $a'(x)$ such that $y_{i_j} = a'(x_{i_j})$ for any $j = 1, \ldots, h-1$ and such that $s' = a'(0)$. Hence, no value of the secret can be ruled out, and thus a group of $h-1$ participants obtain no information about the secret.

4.3.1.3 The Shared Encryption Based Construction

In the following we consider the problem of constructing a hierarchical and shared key assignment scheme by using as building blocks a symmetric encryption

scheme and a perfect secret sharing scheme.

Rationale behind the construction. The idea behind our construction is similar to the one used in the *EBC* (*Encrypted Based Construction*) [33]. In the proposed construction, each class $v \in V$ is assigned a private information s_v, an encryption key k_v, and a public information π_v, which is the encryption of k_v using the private information s_v as a key. Moreover, for each class $v \in V$ and for each edge $e \in \mathcal{I}_v$, there is a public value p_e. If no shared access control for class v's data is needed (recall that in this case we can consider the trivial access structure $\mathcal{A}_v = \mathcal{I}_v$), the value p_e is computed as the encryption of the secret s_v, using the private information s_u as a key, where u and v are the endpoints of the edge e, i.e., $\phi(e) = (u, v)$. On the other hand, if a shared access control on class v's data is needed, we have to consider the m_v access structures $\mathcal{A}_v^1, \ldots, \mathcal{A}_v^{m_v}$ associated to class v. The idea is to use a perfect secret sharing scheme for each $j = 1, \ldots, m_v$, in order to compute the shares of the private information s_v according to the access structure $\mathcal{A}_v^j$ on the set of edges $\mathcal{P}_v^j$. More precisely, let $e \in \mathcal{P}_v^j$ such that $\phi(e) = (u, v)$, and let $s_v^{j,u}$ be the share for the secret s_v associated to the edge e, according to the access structure $\mathcal{A}_v^j$. Such a share is encrypted with the private information s_u as a key and corresponds to the public value p_e associated to the edge e.

Given a class $v \in V$, any set of classes X such that $v \in A_X$ can obtain a set of shares for the secret s_v, decrypting some public values. Such shares allow for the computation of the secret s_v, which can then be used to decrypt the public value π_v, in order to get the key k_v. We will show that a static adversary attacking a class u and corrupting a set of classes X such that $u \notin A_X$, is not able to distinguish the key k_u from a random string of the same length unless it is able to break the underlying encryption scheme.

Let $G = (V, E, \phi)$ be a directed multigraph and let $\mathcal{A}_G$ be a family of access structures associated to classes in V. Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme, and, for any $v \in V$ and any $j = 1, \ldots, m_v$, let $\Sigma_v^j = (Share_v^j, Recover_v^j)$ be a perfect secret sharing scheme for the access structure $\mathcal{A}_v^j$. The information generation algorithm *Gen* of the SEBC is shown in Algorithm 10, whereas, the relative key derivation algorithm *Der* is shown in

Algorithm 11.

Algorithm 10 *Gen* algorithm of the Shared Encryption Based Construction.

procedure $Gen(1^\tau, G, \mathcal{A}_G)$
▷ Generation of sequences s and k
  for each class $u \in V$ **do**
    $s_u \leftarrow \mathcal{K}(1^\tau)$, $k_u \leftarrow \{0,1\}^\tau$
  end for
  ▷ Generation of sequence *pub*
  for each class $u \in V$ **do**
    if v requires access control according to $\mathcal{A}_v^1, \ldots, \mathcal{A}_v^{m_v}$ **then**
      for any $j = 1, \ldots, m_v$ **do**
        Use the $Share_v^j$ algorithm to generate the shares for the secret s_v, according to $\mathcal{A}_v^j$
        for each edge $e \in \mathcal{P}_v^j$ s.t. $\phi(e) = (u, v)$ **do**
        ▷ Let $s_v^{j,u}$ be the corresponding share
          Compute the public value $p_e \leftarrow \mathcal{E}_{s_u}(s_v^{j,u})$
        end for
      end for
    else
      for each edge $e \in \mathcal{I}_v$, where $\phi(e) = (u, v)$ **do**
        Compute the public value $p_e \leftarrow \mathcal{E}_{s_u}(s_v)$
      end for
    end if
    Compute the public value $\pi_v \leftarrow \mathcal{E}_{s_v}(k_v)$
  end for
end procedure

The SEBC associates a public value p_e to each edge $e \in E$, as well as a public value π_u to each class $u \in V$.

4.3.1.4 Analysis of the Scheme

In this section we show that the security property of the SEBC depends on the security properties of the underlying encryption scheme and of the perfect secret sharing scheme.

In particular, we show that if there exists an adversary able to break the security of the SECB in the sense of `IND-ST`, that is to say, which it is able to distinguish a value assigned by the SEBC from a randomly chosen one, then such

Algorithm 11 *Der* algorithm of the Shared Encryption Based Construction.

procedure $Der(1^\tau, G, \mathcal{A}_G, X, s_X, u, pub)$
  if $u \in X$ **then**
    Extract s_u from s_X and the public value π_u from pub
    $k_u \leftarrow \mathcal{D}_{s_u}(\pi_u)$
  else
  ▷ Let $1 \leq i \leq diam(G)$ be an index s.t. $u \in A_X^i$
    for each $\ell = 1, \ldots, i$ **do**
      for each class $v \in A_X^\ell$ **do**
      ▷ Let $Y_v \subseteq A_X^{\ell-1}$ be s.t. $E_{Y_v} \in \mathcal{A}_v^j$ and $1 \leq j \leq m_v$
        for each class $w \in Y_v$ **do**
          Extract the public value p_e from pub
          ▷ Let $\phi(e) = (w, v)$
          Compute the share $s_v^{j,w} \leftarrow \mathcal{D}_{s_w}(p_e)$
        end for
        ▷ On input the shares associated to edges in E_{Y_v}
        Use $Recover_v^j$ algorithm to compute s_v
      end for
    end for
  end if
  Extract the public value π_u from pub and compute $k_u \leftarrow \mathcal{D}_{s_u}(\pi_u)$
end procedure

an adversary can be used as a "*black-box*" to construct an adversary which breaks the underlying symmetric encryption scheme with respect of IND-P1-C0.

Our proof is essentially based on two well known concepts, referred to as *black-box reductions* [44] and *hybrid arguments* [18].

Theorem 4.3.1. *If the encryption scheme* $\Pi = (\mathcal{K}, \mathcal{D}, \mathcal{E})$ *is secure in the sense of* IND-P1-C0 *and* Σ *is a perfect secret sharing scheme, then the SEBC is secure in the sense of* IND-ST.

Proof. Let $G = (V, E, \phi)$ be a directed multigraph, let $u \in V$ and let $\mathtt{STAT}_{u,X}$ be a static adversary which attacks class u and corrupts a set of classes $X \subset V$ such that $u \notin A_X$. Let $G_u = (V_u, E_u, \phi_u)$ be the subgraph of G induced by the set of vertices $V_u = \{v \in V : \text{ there is a path from } v \text{ to } u \text{ in } G\}$ and let $G_{u,X} = (V_{u,X}, E_{u,X}, \phi_{u,X})$ be the subgraph of G_u induced by the set of vertices $V_{u,X} = V_u \setminus X$, containing the classes in V_u which have not been corrupted by $\mathtt{STAT}_{u,X}$. Without loss of generality, let $(u_1, \ldots, u_m)$, where $u_m \equiv u$, be any topological ordering of the vertices in $V_{u,X}$ and let $(e_1, \ldots, e_{h-1})$ be the sequence of edges in $E_{u,X}$ such that $\phi_{u,X}(e_i) = (u_a, u_b)$ precedes $\phi_{u,X}(e_j) = (u_c, u_d)$ if and only if either $a < c$ or $a = c$ and $b < d$. Moreover, let $\phi_{u,X}(e_h) = (u, u')$.

In order to prove the theorem, we need to show that the adversary's views in experiments $\mathbf{Exp}^{\mathtt{IND-1}}_{\mathtt{STAT}_{u,X}}$ and $\mathbf{Exp}^{\mathtt{IND-0}}_{\mathtt{STAT}_{u,X}}$ are indistinguishable. Notice that the only difference between $\mathbf{Exp}^{\mathtt{IND-1}}_{\mathtt{STAT}_{u,X}}$ and $\mathbf{Exp}^{\mathtt{IND-0}}_{\mathtt{STAT}_{u,X}}$ is the last input of $\mathtt{STAT}_{u,X}$, which corresponds to the real key k_u in the former experiment and to a random value chosen in $\{0,1\}^\tau$ in the latter. Thus, while in $\mathbf{Exp}^{\mathtt{IND-1}}_{\mathtt{STAT}_{u,X}}$ the public information is related to the last input of $\mathtt{STAT}_{u,X}$, in $\mathbf{Exp}^{\mathtt{IND-0}}_{\mathtt{STAT}_{u,X}}$ it is completely independent on such a value. For ease of exposition, we define the following experiment:

Experiment $\mathbf{Exp}_{u,X}(1^\tau, G, \mathcal{A}_G)$
  $(s, k, pub) \leftarrow Gen(1^\tau, G, \mathcal{A}_G)$
  $s_X \leftarrow Corrupt_u(s, X)$
  $d \leftarrow \mathtt{STAT}_{u,X}(1^\tau, G, \mathcal{A}_G, pub, s_X, \alpha_u)$
  return d

which corresponds either to $\mathbf{Exp}^{\mathtt{IND-1}}_{\mathtt{STAT}_{u,X}}$, if α_u is the real key k_u, or to $\mathbf{Exp}^{\mathtt{IND-0}}_{\mathtt{STAT}_{u,X}}$, if α_u is a random value in $\{0,1\}^\tau$.

We will show that the adversary's view in the experiment $\mathbf{Exp}_{u,X}$ is indistinguishable from the adversary's view in an experiment $\mathbf{Exp}^{*}_{u,X}$, where the public information, at the same time, does not carry any information about the key k_u and is independent on the last input of $\mathtt{STAT}_{u,X}$. More formally, the experiment $\mathbf{Exp}^{*}_{u,X}$ is defined as follows:

$$
\begin{array}{l}
\text{Experiment } \mathbf{Exp}^{*}_{u,X}(1^{\tau}, G, \mathcal{A}_G) \\
\quad (s, k, pub^{*}) \leftarrow Gen^{*}(1^{\tau}, G, \mathcal{A}_G) \\
\quad s_X \leftarrow Corrupt_u(s, X) \\
\quad d \leftarrow \mathtt{STAT}_{u,X}(1^{\tau}, G, \mathcal{A}_G, pub^{*}, s_X, \alpha_u) \\
\quad \textbf{return } d
\end{array}
$$

In the algorithm Gen^{*} the public value π_u associated to class u is computed as the encryption $\mathcal{E}_{s_u}(\rho)$ of a random value $\rho \in \{0,1\}^{\tau}$, rather than the encryption of the key k_u. Moreover, the public value associated to each edge e_i, where $\phi_{u,X}(e_i) = (u_a, u_b) \in E_{u,X}$ is computed as the encryption $\mathcal{E}_{s_{u_a}}(r_i)$ of a random value $r_i \in \{0,1\}^{\tau}$, rather than the encryption $\mathcal{E}_{s_{u_a}}(s^{u_a}_{u_b})$ of the share $s^{u_a}_{u_b}$ for the private information s_{u_b}. Therefore, in such an experiment, all public information is independent on the value of the key k_u. Moreover, the distributions of the experiment $\mathbf{Exp}^{*}_{u,X}$ when $\mathtt{STAT}_{u,X}$ is given as last input either the real key k_u or a random value in $\{0,1\}^{\tau}$ are the same. In such an experiment, the key k_u is just a random value independent on the public and private information in the adversary's view.

Now, in order to show that $\mathbf{Exp}^{\mathtt{IND-1}}_{\mathtt{STAT}_{u,X}}$ and $\mathbf{Exp}^{\mathtt{IND-0}}_{\mathtt{STAT}_{u,X}}$ are indistinguishable, we only need to show that the adversary's views in experiments $\mathbf{Exp}_{u,X}$ and $\mathbf{Exp}^{*}_{u,X}$ are indistinguishable. This implies that $\mathbf{Exp}^{\mathtt{IND-1}}_{\mathtt{STAT}_{u,X}}$ and $\mathbf{Exp}^{\mathtt{IND-0}}_{\mathtt{STAT}_{u,X}}$ are both indistinguishable from the same experiment $\mathbf{Exp}^{*}_{u,X}$, which also means that they are indistinguishable from each other.

We construct a sequence of $h+1$ experiments $\mathbf{Exp}^{1}_{u,X}, \ldots, \mathbf{Exp}^{h+1}_{u,X}$, all defined over the same probability space, where the first and the last experiments of the sequence correspond to $\mathbf{Exp}_{u,X}$ and $\mathbf{Exp}^{*}_{u,X}$. In each experiment, we modify the way in which the view of $\mathtt{STAT}_{u,X}$ is computed, while maintaining the view's distributions indistinguishable among any two consecutive experiments. For any $q = 2, \ldots, h$, experiment $\mathbf{Exp}^{q}_{u,X}$ is defined as follows:

Experiment $\mathbf{Exp}^{q}_{u,X}(1^{\tau}, G, \mathcal{A}_G)$
$(s, k, pub^q) \leftarrow Gen^q(1^{\tau}, G, \mathcal{A}_G)$
$s_X \leftarrow Corrupt_u(s, X)$
$d \leftarrow \mathtt{STAT}_{u,X}(1^{\tau}, G, \mathcal{A}_G, pub^q, s_X, \alpha_u)$
return d

The algorithm Gen^q used in $\mathbf{Exp}^{q}_{u,X}$ differs from Gen by the way in which part of the public information pub^q is computed. For any $i = 1, \ldots, q-1$, the public values associated to the edge e_i such that $\phi_{u,X}(e_i) = (u_a, u_b) \in E_{u,X}$ is computed as the encryption $\mathcal{E}_{s_{u_a}}(r_i)$ of a random value $r_i \in \{0,1\}^{\tau}$, instead of the encryption $\mathcal{E}_{s_{u_a}}(s^{u_a}_{u_b})$ of the share $s^{u_a}_{u_b}$.

In the following we show that, for any $q = 1, \ldots, h$, the adversary's view in the q-th experiment is indistinguishable from the adversary's view in the $(q+1)$-th one.

Assume by contradiction that there exists a polynomial-time distinguisher B_q, which is able to distinguish between the adversary $\mathtt{STAT}_{u,X}$'s views in experiments $\mathbf{Exp}^{q}_{u,X}$ and $\mathbf{Exp}^{q+1}_{u,X}$ with non-negligible advantage. Notice that such views differ only for the way the public information associated to the edge e_q, such that $\phi_{u,X}(e_q) = (a, b)$, is computed. We show how to construct a polynomial-time adversary $A = (A_1, A_2)$ that uses B_q to break the security of the encryption scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ in the sense of `IND-P1-CO`.

In particular, algorithm A_1, on input 1^{τ}, randomly chooses the key k_v for any class $v \in V$, as well as the private information for any class $v \in V \setminus \{a\}$.

If $I_a \neq \emptyset$, for any class $v \in I_a$, A_1 considers the m_a access structures $\mathcal{A}^1_a, \cdots, \mathcal{A}^{m_a}_a$ associated to the secret s_a and, for each of them, the set of edges characterizing that access structure. Then, A_1 computes the public value associated to each edge e such that $\phi_{u,X}(e) = (v, a)$ as the encryption of a random value chosen in $\{0,1\}^{\tau}$, using the private information s_v as a secret key. Afterwards, for any class $v \in V \setminus \{a\}$ such that $I_v \neq \emptyset$, A_1 considers the m_v access structures $\mathcal{A}^1_v, \cdots, \mathcal{A}^{m_v}_v$ associated to the secret s_v and, for each of them, it considers the set of edges characterizing such access structure. Subsequently, A_1 uses the algorithm $Share$, on input the secret information s_v, to compute the sequence of shares according to each access structure defined for the node v, which is characterized by the relative set of edges. Such shares will be used to compute part of the public

information. More precisely, A_1 computes the public values associated to each edge e_r such that $\phi_{u,X}(e_r) = (v, z) \notin \{e_1, \ldots, e_q\}$ as the encryption of a random share relative to the secret information s_z by using s_v as a secret key. Notice that, in order to compute all public values associated to the outgoing edges of class a, except for the edge e_q such that $\phi_{u,X}(e_q) = (a, b)$, A_1 can make queries to the encryption oracle $\mathcal{E}_{s_a}(\cdot)$. Afterwards, A_1 computes the public values associated to each edge e_k such that $\phi_{u,X}(e_k) = (v, z) \in \{e_1, \ldots, e_{q-1}\}$ as the encryption of a random value chosen in $\{0,1\}^\tau$, using as a secret key the private information s_v. Again, A_1 computes the public values associated to all edges e_v such that $\phi_{u,X}(e_v) = (v, v')$, where $e_v \neq e_q$, as the encryption of the key k_v with the private information s_v. Finally, A_1 sets x_1 to be equal either to the key k_u, if $e_q = e_u$, where $\phi_{u,X}(e_u) = (u, u')$, or to a random share relative to the secret s_b, otherwise. Notice that, since Σ is a perfect secret sharing scheme, by Remark 4.3.1, such a share has the same distribution of a random value in $\{0,1\}^\tau$. The sequences s', k and pub' of all private information, keys, and public values constructed by A_1, along with the values x_0 and x_1, are saved in the state information *state*. Recall that the sequence s' contains the private information s_v assigned to all classes $v \in V \setminus \{a\}$. Similarly, the sequence pub' contains the public information associated to all edges in $E \setminus \{e_q\}$. Formally, the algorithm A_1 is defined as shown in Algorithm 12.

Let y be the challenge for the algorithm A, corresponding to the encryption of either x_0 or x_1 with the unknown key s_a. The algorithm A_2, on input $1^\tau, y$, and *state*, constructs the view for the distinguisher B_q as follows: it first extracts from s' the private information s_X held by corrupted users, through the algorithm $Corrupt_u(s', X)$. Then, it computes the public value associated to the edge e_q, not included in pub', in order to obtain the sequence pub. In particular, such a public value is set equal to the challenge y. Finally, A_2 outputs the same output as $B_q(1^\tau, G, \mathcal{A}_G, pub, s_X, x_1)$. More formally, A_2 works as shown in Algorithm 13.

Notice that if y corresponds to the encryption of x_1, then the random variable associated to the adversary's view is exactly the same as the one associated to the adversary view in experiment $\mathbf{Exp}^q_{u,X}$, whereas, if y corresponds to the encryption of x_0, it has the same distribution as the one associated to the adversary's view in experiment $\mathbf{Exp}^{q+1}_{u,X}$. Therefore, if the algorithm B_q is able to distinguish between

Algorithm 12 First stage of the adversary A attacking the $SEBC$.

procedure $A_1^{\mathcal{E}_{s_a}(\cdot)}(1^\tau)$
  $x_0, k_a \leftarrow \{0,1\}^\tau$
  for each $v \in V \setminus \{a\}$ **do**
    $s_v, k_v \leftarrow \{0,1\}^\tau$
  end for
  if $I_a \neq \emptyset$ **then**
    for each $v \in I_a$ **do**
    ▷ Consider the m_a access structures $\mathcal{A}_a^1, \cdots, \mathcal{A}_a^{m_a}$ for s_a
      for $i = 1$ to m_a **do**
        for each $e \in P_a^i$ **do**
        ▷ Let $\phi(e) = (v, a)$
          $r_a \leftarrow \{0,1\}^\tau$
          $p_{(v,a)} \leftarrow \mathcal{E}_{s_v}(r_a)$
        end for
      end for
    end for
  end if
  for each $v \in V \setminus \{a\}$ s.t. $I_v \neq \emptyset$ **do**
  ▷ Consider the m_v access structures $\mathcal{A}_v^1, \cdots, \mathcal{A}_v^{m_v}$ for s_v
    for j $= 1$ to m_v **do**
      $s_{v,A_v^j} \leftarrow Share_v^j(s_v)$
      for each $e \in P_v^j$ s. t. $e \notin \{e_1, \ldots, e_q\}$ **do**
      ▷ Let $\phi(e) = (z, v)$
        $p(z,v) \leftarrow \mathcal{E}_{s_z}(s_v^{j,z})$
      end for
      if $q > 1$ **then**
        for each $e \in P_v^j$ s. t. $e \in \{e_1, \ldots, e_{q-1}\}$ **do**
        ▷ Let $\phi(e) = (z, v)$
          $r_z \leftarrow \{0,1\}^\tau$
          $p_{(z,v)} \leftarrow \mathcal{E}_{s_z}(r_z)$
        end for
      end if
    end for
  end for
  for each $(v, v') \in E$ **do**
    $\pi_v \leftarrow \mathcal{E}_{s_v}(k_v)$
  end for
  $pub' \leftarrow$ public values constructed above
  if $(a, b) = (u, u')$ **then**
    $x_1 \leftarrow k_u$
  else
  ▷ Consider the m_b access structures $\mathcal{A}_b^1, \cdots, \mathcal{A}_b^{m_b}$ for s_b
    for $\ell = 1$ **to** m_b **do**
      for each $e \in P_b^\ell$ **do**
        if $\phi(e) = (a, b)$ **then**
          $x_1 \leftarrow s_b^{\ell,a}$
        end if
      end for
    end for
  end if
  $state \leftarrow (s', k, pub', x_0, x_1)$
  return $(x_0, x_1, state)$
end procedure

Algorithm 13 Second stage of the adversary A attacking the $SEBC$.

procedure $A_2(1^\tau, y, state)$
  $state = (s', k, pub', x_0, x_1)$
  $s_X \leftarrow Corrupt_u(s', X)$
  ▷ Construction of the missing public values
  if $(a, b) = (u, u')$ **then**
    $\pi_u \leftarrow y$
  else
    $p_{(a,b)} \leftarrow y$
  end if
  $d \leftarrow B_q(1^\tau, G, \mathcal{A}_G, pub, s_X, x_1)$
  return d
end procedure

such views with non negligible advantage, it follows that algorithm A is able to break the security of the encryption scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ in the sense of `IND-P1-CO`. Contradiction.

Hence, for any $q = 1, \ldots, h$, the adversary's view in the q-th experiment is indistinguishable from the adversary's view in the $(q+1)$-th one. Therefore, the adversary's views in experiments $\mathbf{Exp}_{u,X}$ and $\mathbf{Exp}^*_{u,X}$ are indistinguishable. This concludes the proof. □

4.3.2 A Construction based on Threshold Broadcast Encryption

In this section we propose a construction for hierarchical and shared key assignment which uses as building block a *threshold broadcast encryption* scheme. We denote such a construction as the *Threshold Broadcast Encryption Based Construction (TBEBC)*. The TBEBC can be instantiated using the construction proposed by Daza et al. [31], which is secure under the *Decisional Bilinear Diffie-Hellman (DBDH)* assumption.

4.3.2.1 Threshold Broadcast Encryption

A broadcast encryption scheme allows a sender to broadcast an encrypted message to a set of users in such a way that only legitimate users can decrypt it.

Broadcast encryption schemes can be either public-key or symmetric-key based. In the symmetric-key setting, only a trusted authority can broadcast data to the receivers. Conversely, in the public-key setting, a public key published by a trusted authority allows anybody to broadcast a message.

In a threshold public key broadcast encryption scheme (TBE) a message is encrypted and sent to a group of receivers, in such a way that the cooperation of at least t of them (where t is the threshold) is necessary in order to recover the original message. Such schemes have many applications in situations where one wants to avoid that a single party has all the power/responsibility to protect or obtain some critical information. In those schemes, the sender of the message who wants to protect some information may want to decide who will be the designated receivers in an ad-hoc way, just before encrypting the message, and also decide the threshold of receivers which will be necessary to recover the information. More precisely, a TBE scheme has the following properties:

- There is no setup phase or predefined groups. Each potential receiver has his own pair of secret/public keys.
- The sender chooses the set of receivers $\mathcal{P}$ and the threshold t for the decryption. Then he encrypts the message by using the public keys of all the receivers in $\mathcal{P}$.
- A ciphertext corresponding to the pair ($\mathcal{P}$, t) can be decrypted only if at least t members of $\mathcal{P}$ cooperate by using their secret keys. Otherwise, it is computationally infeasible to obtain any information about the plaintext.

The next definition was proposed in [31].

Definition 4.3.2. *A threshold broadcast encryption (TBE) scheme consists of five algorithms:*

1. *The randomized setup algorithm TBE.Setup takes as input a security parameter 1^τ and outputs some public parameters params, which will be common to all the users of the system. We write params $\leftarrow$ TBE.Setup(1^τ).*

2. *The randomized key generation algorithm $TBE.KG$ is run by each user i. It takes as input some public parameters params and returns a pair (PK_i,SK_i) consisting of a public key and a matching secret key. We write $(PK_i, SK_i) \leftarrow TBE.KG(params)$.*

3. *The randomized encryption algorithm $TBE.Enc$ takes as input a set of public keys $\{PK_i\}_{i\in\mathcal{P}}$ corresponding to a set $\mathcal{P}$ of receivers, a threshold t satisfying $1 \leq i \leq n$, and a message m. The output is a ciphertext C. We write $C \leftarrow TBE.Enc(1^\tau, \{PK_i\}_{i\in\mathcal{P}}, t, m)$.*

4. *The (possibly randomized) partial decryption algorithm $TBE.PartDec$ takes as input a ciphertext C for the pair ($\mathcal{P}$,t) and a secret key SK_i of a receiver $i \in \mathcal{P}$. The output is a partial decryption value k_i or a special symbol $\perp$. We write $k_i \leftarrow TBE.PartDec(C, SK_i)$.*

5. *The deterministic final decryption algorithm $TBE.Dec$ takes as input a ciphertext C for the pair ($\mathcal{P}$,t) and t partial decryptions $\{k_i\}_{i\in A}$, corresponding to receivers in some subset $A \subset \mathcal{P}$. The output is a message m or a special symbol $\perp$. We write $\widetilde{m} \leftarrow TBE.Dec(C, \{k_i\}_{i\in A}, A)$.*

4.3.2.2 The Threshold Broadcast Encryption Based Construction

In the following we consider the problem of constructing a hierarchical and shared key assignment scheme by using as the building block a threshold broadcast encryption scheme.

Rationale behind the construction. The idea behind our construction, referred to in the following as the Threshold Broadcast Encryption Based Construction (TBEBC), is to compute the private and public information by using the threshold broadcast encryption scheme. More precisely, the public information associated to each security class v will contain a public key generated by a TBE, let PK_v be such a key. Given a class v with its relative access structures $\mathcal{A}_v^1, \ldots, \mathcal{A}_v^{m_v}$ and a qualified set $X \in \mathcal{A}_v^j$, for some $j \in \{1 \ldots m_v\}$, we denote with Q_X^v the set of classes having an outgoing edge in X. Furthermore, the public information will contain for each set Q_X^v a value given by the encryption of the

secret key k_v, through the public keys of all the classes in Q_X^v. Subsequently, the public value relative to the set Q_X^v can be decrypted through the collaboration among all the classes that constitute this set; that is to say, through their private keys, the classes belonging to Q_X^v are allowed to compute the key k_v.

In detail, the generation algorithm of our TBEBC takes as inputs a security parameter 1^τ, a multigraph G, and the corresponding set of families of access structures $\mathcal{A}_G$. This algorithm generates a pair of public/private keys (PK_v, SK_v), for each class $v \in V$. Subsequently, for each class v, we assign secret information s_v, corresponding to the private key SK_v. Moreover, for each class v, the public key PK_v corresponds to the public information pub_v. In addition, for each class v, a secret key k_v is generated. Again, for each class v and for each set Q_X^v, k_v is encrypted through the public keys of the classes belonging to such a set. Finally, this encryption is assigned to the public information associated with the class v.

On the other hand, derivation algorithm of the TBEBC takes as inputs a security parameter 1^τ, a multigraph G, the corresponding set of families of access structures $\mathcal{A}_G$, the class u to be accessed, a set Q_X^u, the private information s_X^u associated to classes in Q_X^u and finally, all public information *pub*. This algorithm first extracts from *pub* the value C_X^u, relative to Q_X^u. Subsequently, it extracts from s_X^u the secret values associated to each class belonging to Q_X^u. Finally, through these values, the partial decryptions related to C_X^u are computed, for being used later to obtain the secret key k_u.

Formally, let $G = (V, E, \phi)$ be a directed multigraph and let $\mathcal{A}_G$ be a family of access structures associated to classes in V. Let $TBE = (TBE.Setup, TBE.KG, TBE.Enc, TBE.PartDec, TBE.Dec)$ a threshold broadcast encryption scheme. The information generation algorithm *Gen* of the TBEBC is shown in Algorithm 14, whereas, the relative key derivation algorithm *Der* is shown in Algorithm 15.

4.3.2.3 Analysis of the Scheme

In the following we show that the security property of the TBEBC depends on the security property of the underlying threshold broadcast encryption scheme.

Algorithm 14 *Gen* algorithm of the Threshold Broadcast Encryption Based Construction.

procedure $Gen(1^\tau, G, \mathcal{A}_G)$
  $params \leftarrow TBE.Setup(1^\tau)$
  for each class $v \in V$ **do**
    $(PK_v, SK_v) \leftarrow TBE.KG(params)$
    $k_v \leftarrow \{0,1\}^\tau$, $s_v \leftarrow SK_v$, $pub_v \leftarrow PK_v$
    ▷ Let $m_v \geq 1$ be an integer
    ▷ Let $P_v^1, \cdots, P_v^{m_v}$ be m_v subsets of I_v
    ▷ Let $\mathcal{A}_i^1, \cdots, \mathcal{A}_v^{m_v}$ be the m_v access structures for class v on the sets $P_v^1, \cdots, P_v^{m_v}$
    for $j = 1$ to m_v **do**
    ▷ Let $Q_1^{v,j}, \cdots, Q_z^{v,j}$ be the collection of qualified sets characterizing the access structure $\mathcal{A}_v^j$
      for $k = 1$ to z **do**
      ▷ Let $card_k^{v,j}$ be the cardinality of the set $Q_k^{v,j}$
        $C_k^{v,j} \leftarrow TBE.Enc(Q_k^{v,j}, \{PK_\ell\}_{\ell \in Q_k^{v,j}}, card_k^{v,j}, k_v)$
        $pub_v \leftarrow C_k^{v,j}$
      end for
    end for
  end for
end procedure

Algorithm 15 *Der* algorithm of the Threshold Broadcast Encryption Based Construction.

procedure $Der(1^\tau, G, \mathcal{A}_G, Q_X^u, s_X^u, u, pub)$
  Extract from pub the value C_X^u associated to the class u
  for each $\ell \in Q_X^u$ **do**
    Extract from s_X^u the secret value s_ℓ associated to the class ℓ
    $Share_\ell \leftarrow TBE.PartDec(C_X^u, s_\ell)$
  end for
  $k_u \leftarrow TBE.Dec(C_X^u, \{Share_\ell\}_{\ell \in Q_X^u}, Q_X^u)$
end procedure

Before analyzing the security of the TBEBC, we first need to define what we mean by a *secure public-key threshold broadcast encryption scheme*. In general, in such schemes an adversary can corrupt different users in two possible ways: registering new public keys for such users, or obtaining the secret key matching with the public key of some previously honest users. The final goal of the adversary is to obtain some information about a message which has been encrypted for a pair ($\mathcal{P}^*$, t^*), such that the number of corrupted players in $\mathcal{P}^*$ is less than t^*. For the ease of exposition, we consider the second kind of user corruption. More precisely, *indistinguishability* for TBE schemes is defined by considering the game shown in Figure 4.2, played by an adversary A_{atk} against a challenger [12].

$\mathcal{U} = \emptyset$
$params \leftarrow TBE.Setup(1^\tau)$
Each time A_{atk} requires the creation of a new user R_i
$(PK_i, SK_i) \leftarrow$ TBE.KG($params$)
$\mathcal{U} \leftarrow \mathcal{U} \cup R_i$
$(St, \mathcal{P}^*, t^*, m_0, m_1) \leftarrow A_{atk}^{Corr,\mathcal{O}_1}(\cdot)$(find, params, $\{PK_i\}_{i\in\mathcal{U}}$)
$\beta \xleftarrow{r} \{0,1\}$
$C^* \leftarrow$ TBE.Enc($\mathcal{P}^*$,$\{pk_i\}_{i\in\mathcal{P}^*}$,$t^*$,$m_\beta$)
$\beta' \leftarrow A_{atk}^{Corr,\mathcal{O}_2}(\cdot)(guess, C^*, St)$

Figure 4.2: Game played by an adversary A_{atk}.

In both phases of the attack, A_{atk} can access a corruption oracle $Corr$. In particular, A_{atk} submits to the oracle a user $i \in \mathcal{U}$ and receives as answer the relative secret key SK_i. Let $\mathcal{U}' \subset \mathcal{U}$ be the subset of users that A_{atk} has corrupted during the attack. Notice that $|\mathcal{P}^* \cap \mathcal{U}'| < t^*$ must hold, otherwise, A_{atk} knows the secret key of at least t^* players in P^* and can decrypt C^* autonomously, obtaining m_β. In detail, depending on the considered type of attack, A_{atk} can also access a decryption oracle for ciphertexts of his choice. As an answer, A_{atk} receives all the information that would be broadcasted in a complete decryption process, that is, all the partial decryption values and the resulting plaintext. More precisely, if *atk* is a *Chosen Plaintext Attack (CPA)*, then the adversary cannot access the decryption oracle at all, i.e., $\mathcal{O}_1 = \mathcal{O}_2 = \epsilon$. If *atk* is a partial *Chosen Ciphertext Attack (CCA1)*, then $\mathcal{O}_1$ = TBE.PartDec($\cdot$) $\cup$ TBE.Dec($\cdot$) and $\mathcal{O}_2 = \epsilon$. Finally, if *atk* is a full *Chosen Ciphertext Attack (CCA2)*, then $\mathcal{O}_1 = \mathcal{O}_2$=TBE.PartDec($\cdot$)

$\cup$ TBE.Dec$(\cdot)$. Obviously, in the last case, A_{CCA2} is not allowed to query the oracle $\mathcal{O}_2$ with the challenge ciphertext C^*.
The advantage of A_{atk} is defined as:

$$\mathbf{Adv}(A_{atk}) = Pr[\beta' = \beta] \quad - \quad \frac{1}{2}.$$

A threshold broadcast encryption scheme is said to be ϵ-indistinguishable under *atk* attacks if $\mathbf{Adv}(A_{atk}) < \epsilon$ for any adversary A_{atk} running in polynomial time. Daza et al. [31] proposed a construction for threshold broadcast encryption schemes and showed it to be ϵ-indistinguishable under different kinds of attacks. Now we are ready to prove the next theorem.

Theorem 4.3.2. *If the public-key threshold broadcast encryption scheme TBE = (TBE.Setup, TBE.KG, TBE.Enc, TBE.PartDec, TBE.Dec) is ϵ-indistinguishable under atk attacks, then the $TBEBC$ is secure in the sense of* `IND-ST`.

Proof. Assume by contradiction that the $TBEBC$ is not secure in the sense of `IND-ST`. Thus, there exists a multigraph $G = (V, E, \phi)$ in Γ and a class $u \in V$ for which there exists a polynomial-time adversary $\mathtt{STAT}_{u,X}$, whose advantage $\mathbf{Adv}^{\mathtt{IND}}_{\mathtt{STAT}_{u,X}}(1^\tau, G, \mathcal{A}_G)$ is non-negligible. We show how to construct a polynomial-time adversary A_{atk} that, by using $\mathtt{STAT}_{u,X}$, is able to break the ϵ-indistinguishability of the TBE scheme used as a building block in the $TBEBC$.

In particular, let a be the target class, let Q_a be a qualified set for a and finally, let $card_a = |Q_a|$. The first operation performed by the challenger is the generation of some parameters which will be used later on. The adversary A_{atk} chooses two messages, m_0 and m_1, both having the same length. For each node $v \in V$, A_{atk} asks the challenger for the creation of a new user v, along with the relative pair of public/private keys, denoted by PK_v and SK_v, respectively. It is important to remark that only public keys will be in the adversary's view. For each class $v \in V$, A_{atk} assigns the public key PK_v to the public information pub_v, besides assigning a secret key k_v to such a class. Again, the secret key k_a regarding the target class a is made to correspond to the message m_1. Subsequently, A_{atk} through its corruption oracle, corrupts a set X of users, such that a is not in

A_X. Let s_X be the output of such a corruption, constituted by private keys associated to users in X. This output is then stored in the state variable St. After those steps, A_{atk} outputs some information through which it intends to be challenged. Later, the challenger computes an encryption C^* of a message chosen at random between m_0 and m_1, relative to the qualified set Q_a; let m_β be such a message. The encryption C^*, which represents the challenge for A_{atk}, is assigned to the public information for the node a, denoted by pub_a. By means of the aforementioned steps, A_{atk} simulated the full view for the distinguisher B_a, which is able to attack the security of $TBEBC$ in the sense of `IND-ST` with non-negligible advantage. More precisely, B_a is able to distinguish between the encryption of k_a from that of a random value. Finally, A_{atk} returns the same output as B_a, denoted by d. Formally, the adversary A_{atk} is defined as shown in Algorithm 16.

Algorithm 16 Functioning of the adversary $A_{atk}^{Corr,\mathcal{O}_1,\mathcal{O}_2}$ attacking the $TBEBC$.

procedure $A_{atk}^{Corr,\mathcal{O}_1,\mathcal{O}_2}(1^\tau)$
  $params \leftarrow TBE.Setup(1^\tau)$
  $m_0, m_1 \leftarrow \{0,1\}^\tau$
  for each node $v \in V$ **do**
  ▷ A_{atk} asks the challenger for the creation of a new user v
    $(PK_v, SK_v) \leftarrow TBE.KG(params)$
    $pub_v \leftarrow PK_v$, $k_v \leftarrow \{0,1\}^\tau$, $k_a \leftarrow m_1$
  end for
  ▷ Corruption of a set X of users s.t. a is not in A_X. Let s_X be the output of such a corruption
  $St \leftarrow s_X$
  $(St, Q_a, card_a, m_0, m_1) \leftarrow A_{atk}^{Corr,\mathcal{O}_1(\cdot)}(find, params, \{PK_v\}_{v\in V})$
  ▷ The challenger chooses at random a message between m_0 and m_1 and provides the adversary with the encryption C^* of such a message
  $\beta \xleftarrow{r} \{0,1\}$
  $C^* \leftarrow TBE.Enc(Q_a, \{PK_j\}_{j\in Q_a}, card_a, m_\beta)$
  $pub_a \leftarrow C^*$
  ▷ Construction of the missing public values
  $d \leftarrow \mathcal{B}_a(1^\tau, G, A_G, pub, s_X, m_1)$
$A_{atk}^{Corr,\mathcal{O}_2(\cdot)}(guess, C^*, St)$ returns the same d as B_a
end procedure

Note that if the last input for $\texttt{STAT}_{u_X}$ is equal to the key hidden into the

public value C^*, then the random variable associated to $\mathtt{STAT}_{u_X}$'s view is exactly the same as in experiment $\mathbf{Exp}^{\mathtt{IND-1}}_{\mathtt{STAT}_{u_X}}$, whereas, if it is a random string, such a variable has the same distribution as the one associated to $\mathtt{STAT}_{u_X}$'s view in experiment $\mathbf{Exp}^{\mathtt{IND-0}}_{\mathtt{STAT}_{u_X}}$. Finally, A_{atk} outputs the same output as $\mathtt{STAT}_{u_X}(1^\tau, G, \mathcal{A}_G, pub, s_X, \alpha_u)$. Therefore, it holds that

$$\mathbf{Adv}(A_{atk}) = \mathbf{Adv}^{\mathtt{IND}}_{\mathtt{STAT}_{u_X}}(1^\tau, G, \mathcal{A}_G).$$

Since $\mathbf{Adv}^{\mathtt{IND}}_{\mathtt{STAT}_{u_X}}(1^\tau, G, \mathcal{A}_G)$ is non-negligible, it follows that the adversary A_{atk} is able to break the ϵ-indistinguishability of the threshold broadcast encryption scheme. Contradiction. □

4.3.3 Performance Evaluation

The SEBC provides constant private information and public information linear in the number of the edges in the multigraph G. More precisely, the public information can be at most $(|E| + |V|)ck$, where k corresponds to the size of the secret key in this construction and c is a constant depending on the underlying symmetric encryption scheme. For instance, c is equal to 2 for the so called XOR construction in [11].

On the other hand, in the SEBC, for any $X \subseteq V$, the complexity of key derivation depends on the set A_X of classes that can be accessed when classes in X collaborate together. Such a set is constructed by using a *Breadth-First-Search (BFS)* on G, starting from the set X. In detail, starting from X, we visit the multigraph G outgoing from X in all possible directions, adding classes one layer at a time, according to the access structures associated to the classes. Thus, besides the computational effort required by the BFS visit, the key derivation complexity in the SEBC is characterized by a number of decryptions which is equal to the number of classes corresponding to each layer of cooperation necessary to obtain the encryption key. Finally, in addition to the above number of decryptions, it is also necessary to employ the *Recover* algorithm of the perfect secret sharing scheme for reconstructing the secret key s_v.

Regarding the TBEBC, the number of public information associated to a security class v is given by $1 + z$, where z is the number of qualified sets that can

access the class v. On the other hand, the only secret information assigned to each class v is given by the private key generated by the TBE scheme. Instead, concerning the complexity of key derivation, in the case of TBEBC the number of decryptions is equal to the number of classes belonging to a given qualified set for a class v.

Chapter 5

General Conclusions

"There are in fact two things, science and opinion; the former begets knowledge, the latter ignorance."

— Hippocrates of Kos, *460 BC - 370 BC*

Nowadays the current network-centric world has given rise to several security concerns regarding access control management, which ensures that only authorized users are given access to certain resources or tasks. In particular, according to their respective roles and responsibilities, users are typically organized into *hierarchies* composed of several disjoint classes (*security classes*). A hierarchy is characterized by the fact that some users may have more access rights than others, according to a top-down inclusion paradigm following specific hierarchical dependencies. A user with access rights for a given class is granted access to objects stored in that class, as well as to all the descendant ones in the hierarchy. The problem of *key management* for such hierarchies is referred to as *hierarchical key assignment* and consists in assigning a key to each class of the hierarchy, so that the keys for descendant classes can be efficiently obtained from users belonging to classes at a higher level in the hierarchy.

In Chapter 2 we have explored the relations between all security notions for hierarchical key assignment schemes and, in particular, we have shown that security with respect to strong key indistinguishability is *not stronger* than the one with respect to key indistinguishability. We have also proposed a general construction yielding a hierarchical key assignment scheme offering security against strong key

recovery, given any hierarchical key assignment scheme which guarantees security against key recovery.

In Chapter 3 we have considered hierarchical key assignment schemes supporting dynamic updates, such as insertions and deletions of classes and relations between classes, as well as key replacements and user revocations. We have extended existing security notions for hierarchical key assignment schemes, namely, security with respect to *key indistinguishability* and against *key recovery*, by providing the adversary with further attack abilities. Then, we have shown how to construct a novel hierarchical key assignment scheme supporting dynamic updates by using as a building block a symmetric encryption scheme. It is important to emphasize that this is the first available scheme crafted for non-static environments, where the adversary is allowed to dynamically update the hierarchy. The proposed construction is provably secure with respect to key indistinguishability and requires a single computational assumption. Moreover, it provides efficient key derivation and updating procedures, while requiring each user to store only a single private key. For its simplicity, effectiveness and robustness the proposed scheme may result in a fundamental practice for hierarchical access control applications in dynamic scenarios.

In Chapter 4 we have proposed an access control model with some innovative features. In particular, starting from the consideration that in some cases, besides the conventional hierarchical access, access should be granted to some qualified sets of users, the above model provides the user with the ability to prevent abuse of permissions, to define alternative access methods and to allow the separation of duties. Such a novel access model finds a natural field of application in several contexts. In general, our model characterizes any scenario where more than one entity is required to gain a specific authorization. In addition, such a model is useful in environments where it is necessary to address situations requiring special permissions. Moreover, in this chapter we provided the first formal definition of hierarchical and shared key assignment schemes. Again, we proposed an efficient construction for those schemes, denoted as *Shared Encryption Based Construction* (*SEBC*), which assigns to each class a single private information, whereas, the public information depends on the number of classes, as well as on the number of edges in the hierarchy. The security of the proposed construction relies on the ones

of the underlying encryption and secret sharing schemes. Finally, we proposed a construction based on public-key threshold broadcast encryption, denoted as *Threshold Broadcast Encryption Based Construction (TBEBC)*, which assigns to each class a single private information, whereas, the public information depends on the number of qualified sets which can access such a class. The security of the proposed construction relies on the one of the underlying threshold broadcast encryption scheme.

Appendix A

List of Papers Not Covered in this Thesis

A.1 Papers in Journals

1. Arcangelo Castiglione, Raffaele Pizzolante, Francesco Palmieri, Barbara Masucci, Bruno Carpentieri, Alfredo De Santis and Aniello Castiglione: "On-board Format-independent Security of functional Magnetic Resonance Images". Accepted for pubblication in ACM Transactions on Embedded Computing Systems.

2. Arcangelo Castiglione, Paolo DArco, Alfredo De Santis, Rosario Russo: "Secure group communication schemes for dynamic heterogeneous distributed computing". Future Generation Computer Systems. DOI: 10.1016/j.future.2015.11.026

3. Arcangelo Castiglione, Raffaele Pizzolante, Francesco Palmieri, Alfredo De Santis, Bruno Carpentieri, Aniello Castiglione: "Secure and reliable data communication in developing regions and rural areas", Pervasive and Mobile Computing, Volume 24, December 2015, Pages 117-128, DOI: 10.1016/j.pmcj.2015.04.001

4. Arcangelo Castiglione, Francesco Palmieri, Ugo Fiore, Aniello Castiglione, Alfredo De Santis: "Modeling energy-efficient secure communications in multi-mode wireless mobile devices". Journal of Computer and System Sciences, Vol. 81, Issue 8, pp. 1464-1478, 2015, DOI: 10.1016/j.jcss.2014.12.022

5. Pietro Albano, Andrea Bruno, Bruno Carpentieri, Aniello Castiglione, Arcangelo Castiglione, Francesco Palmieri, Raffaele Pizzolante, Kangbin Yim, Ilsun You: "Secure and Distrbuted Video Surveillance via Portable Device", Journal of Ambient Intelligence Humanized Computing, Vol. 5, No. 2, pp. 205-213, 2013, DOI: 10.1007/s12652-013-0181-z

6. Arcangelo Castiglione, Raffaele Pizzolante, Alfredo De Santis, Bruno Carpentieri, Aniello Castiglione, Francesco Palmieri: "Cloud-based adaptive compression and secure management services for 3D healthcare data", Future Generation Computer Systems, Vol. 5, No. 1, pp. 120-134, 2015, DOI: 10.1016/j.future.2014.07.001

A.2 Papers in International Conferences

1. Arcangelo Castiglione, Alfredo De Santis, Barbara Masucci, Francesco Palmieri, Aniello Castiglione: "On the Relations Between Security Notions in Hierarchical Key Assignment Schemes for Dynamic Structures", 21-st Australasian Conference on Information Security and Privacy (ACISP 2016). Springer.

2. Arcangelo Castiglione, Aniello Castiglione, Alfredo De Santis, Barbara Masucci, Francesco Palmieri, Raffaele Pizzolante (2015, October): "Novel Insider Threat Techniques: Automation and Generation of Ad Hoc Digital Evidence". In Proceedings of the 7-th ACM CCS International Workshop on Managing Insider Security Threats (pp. 29-39). DOI: 10.1145/2808783.2808789. ACM.

3. Raffaele Pizzolante, Arcangelo Castiglione, Alfredo De Santis, Bruno Car-

pentieri, Francesco Palmieri, Aniello Castiglione: “Format-independent Protection of DNA Microarray Images”, 10-th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, 3PGCIC 2015, DOI: 10.1109/3PGCIC.2015.138. IEEE.

4. Raffaele Pizzolante, Arcangelo Castiglione, Alfredo De Santis, Bruno Carpentieri, Aniello Castiglione: “Reversible Copyright Protection for DNA Microarray Images”, Security and Privacy in Systems and Communication Networks (SecureSysComm 2015) workshop, 10-th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, 3PGCIC 2015, DOI: 10.1109/3PGCIC.2015.139. IEEE.

5. Arcangelo Castiglione, Alfredo De Santis, Raffaele Pizzolante, Aniello Castiglione, Vincenzo Loia, Francesco Palmieri: “On the Protection of fMRI Images in Multi-domain Environments”. AINA 2015: 476-481, DOI: 10.1109/AINA.2015.224. IEEE.

6. Arcangelo Castiglione, Alfredo De Santis, Aniello Castiglione, Francesco Palmieri: “An Efficient and Transparent One-Time Authentication Protocol with Non-interactive Key Scheduling and Update”. AINA 2014: 351-358. DOI: 10.1109/AINA.2014.45. IEEE.

7. Giovanni Acampora, Arcangelo Castiglione, Autilia Vitiello: “A fuzzy logic based reputation system for E-markets”. FUZZ-IEEE 2014: 865-872. DOI: 10.1109/FUZZ-IEEE.2014.6891810. IEEE.

8. Raffaele Pizzolante, Arcangelo Castiglione, Bruno Carpentieri, Alfredo De Santis, Aniello Castiglione: “Protection of Microscopy Images through Digital Watermarking Techniques”, The 6-th International Conference on Intelligent Networking and Collaborative Systems, Salerno, Italia, 2014, pp. 65-72, DOI: 10.1109/INCoS.2014.116. IEEE.

9. Arcangelo Castiglione, Raffaele Pizzolante, Alfredo De Santis, Ciriaco DAmbrosio: “A Collaborative Decision-Support System for Secure Analysis of Cranial Disorders”, The 6-th International Conference on Intelligent

Networking and Collaborative Systems, Salerno, Italia, 2014, pp. 65-72, DOI: 10.1109/INCoS.2014.115. IEEE.

10. Raffaele Pizzolante, Arcangelo Castiglione, Bruno Carpentieri, Alfredo De Santis: "Parallel Low-Complexity Lossless Coding of Three-Dimensional Medical Images", The 17-th International Conference on Network-Based Information Systems, Salerno, Italia, 2014, pp. 91-98, DOI: 10.1109/NBiS.2014.107. IEEE.

11. Arcangelo Castiglione, Raffaele Pizzolante, Bruno Carpentieri, Alfredo De Santis: "An Efficient Protocol for Reliable Data Communication on Dataless Devices", The 8-th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, Birmingham, Inghilterra, 2014, pp. 517-522, DOI: 10.1109/IMIS.2014.75. IEEE.

12. Arcangelo Castiglione, Ciriaco D'Ambrosio, Alfredo De Santis, Francesco Palmieri: "Fully Distributed Secure Video Surveillance Via Portable Device with User Awareness". CD-ARES Workshops 2013: 414-429. DOI: 10.1007/978-3-642-40588-4_29. Springer.

13. Arcangelo Castiglione, Ciriaco D'Ambrosio, Alfredo De Santis, Aniello Castiglione, Francesco Palmieri: "On Secure Data Management in Health-Care Environment". IMIS 2013: 666-671. DOI: 10.1109/IMIS.2013.120. IEEE.

14. Arcangelo Castiglione, Alfredo De Santis, Aniello Castiglione, Francesco Palmieri, Ugo Fiore: "An Energy-Aware Framework for Reliable and Secure End-to-End Ubiquitous Data Communications". INCoS 2013: 157-165. DOI: 10.1109/INCoS.2013.32. IEEE.

15. Raffaele Pizzolante, Bruno Carpentieri, Arcangelo Castiglione: "A Secure Low Complexity Approach for Compression and Transmission of 3-D Medical Images", The 8-th International Conference On Broadband and Wireless Computing, Communication and Applications, Compiegne, Francia, 2013, pp. 387-392, DOI: 10.1109/BWCCA.2013.68. IEEE.

16. Raffaele Pizzolante, Bruno Carpentieri, Aniello Castiglione, Arcangelo Castiglione, Francesco Palmieri: "Text Compression and Encryption through

Smart Devices for Mobile Communication", The 7-th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, Taichung, Taiwan, 2013, pp. 672-677, DOI: 10.1109/IMIS.2013.121. IEEE.

17. Pietro Albano, Andrea Bruno, Bruno Carpentieri, Aniello Castiglione, Arcangelo Castiglione, Francesco Palmieri, Raffaele Pizzolante, Ilsun You: "A Secure Distributed Video Surveillance System Based on Portable Devices", The International Cross Domain Conference and Workshop (CD-ARES 2012), Praga, Repubblica Ceca, 2012, pp. 403-415, DOI: 10.1007/978-3-642-32498-7_30. Springer.

References

[1] Alfred V. Aho, M. R. Garey, and Jeffrey D. Ullman. The Transitive Reduction of a Directed Graph. *SIAM J. Comput.*, 1(2):131–137, 1972.

[2] Selim G. Akl and Peter D. Taylor. Cryptographic Solution to a Problem of Access Control in a Hierarchy. *ACM Trans. Comput. Syst.*, 1(3):239–248, 1983.

[3] Alessandro Aldini and Alessandro Bogliolo. *User-Centric Networking*. Springer, 2014.

[4] M. J. Atallah, M. Blanton, N. Fazio, and K. B. Frikken. Dynamic and efficient key management for access hierarchies. *ACM Trans. on Inf. and System Security*, 12(3), 2009.

[5] Mikhail J. Atallah, Marina Blanton, and Keith B. Frikken. Key Management for Non-Tree Access Hierarchies. In David F. Ferraiolo and Indrakshi Ray, editors, *SACMAT 2006,11th ACM Symposium on Access Control Models and Technologies, Lake Tahoe, California, USA, June 7-9, 2006, Proceedings*, pages 11–18. ACM, 2006.

[6] Mikhail J. Atallah, Marina Blanton, and Keith B. Frikken. Incorporating Temporal Capabilities in Existing Key Management Schemes. In Joachim Biskup and Javier Lopez, editors, *Computer Security - ESORICS 2007, 12th European Symposium On Research In Computer Security, Dresden, Germany, September 24-26, 2007, Proceedings*, volume 4734 of *Lecture Notes in Computer Science*, pages 515–530. Springer, 2007.

[7] G. Ateniese, A. De Santis, A. L. Ferrara, and B. Masucci. Provably-secure time-bound hierarchical key assignment schemes. *J. of Cryptology*, 25(2):1–15, 2012.

[8] Giuseppe Ateniese, Alfredo De Santis, Anna Lisa Ferrara, and Barbara Masucci. Provably-Secure Time-Bound Hierarchical Key Assignment Schemes. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, Ioctober 30 - November 3, 2006*, pages 288–297. ACM, 2006.

[9] Giuseppe Ateniese, Alfredo De Santis, Anna Lisa Ferrara, and Barbara Masucci. A Note on Time-Bound Hierarchical Key Assignment Schemes. *Inf. Process. Lett.*, 113(5-6):151–155, 2013.

[10] M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. In *CRYPTO 1996, LNCS*, volume 1109, pages 1–15, 1996.

[11] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption. In *The 38th IEEE Symp. on Foundations of Comp. Sci.*, pages 394–403, 1997.

[12] Mihir Bellare, Alexandra Boldyreva, and Silvio Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In *Advances in Cryptology-EUROCRYPT 2000*, pages 259–274. Springer, 2000.

[13] J.C. Benaloh and J. Leichter. Generalized secret sharing and monotone functions. In *CRYPTO 1988, LNCS*, volume 403, pages 27–35, 1990.

[14] Elisa Bertino, Barbara Carminati, and Elena Ferrari. A temporal key management scheme for secure broadcasting of XML documents. In Vijayalakshmi Atluri, editor, *Proceedings of the 9th ACM Conference on Computer and Communications Security, CCS 2002, Washington, DC, USA, November 18-22, 2002*, pages 31–40. ACM, 2002.

[15] Elisa Bertino, Ning Shang, and Samuel S. Wagstaff Jr. An Efficient Time-Bound Hierarchical Key Management Scheme for Secure Broadcasting. *IEEE Trans. Dependable Sec. Comput.*, 5(2):65–70, 2008.

[16] G. R. Blakley. Safeguarding cryptographic keys. In *AFIPS Nat. Comp. Conference*, pages 313–317, 1979.

[17] Matt Blaze, Joan Feigenbaum, and Angelos D. Keromytis. Keynote: Trust management for public-key infrastructures (position paper). In Bruce Christianson, Bruno Crispo, William S. Harbison, and Michael Roe, editors, *Security Protocols, 6th International Workshop, Cambridge, UK, April 15-17, 1998, Proceedings*, volume 1550 of *Lecture Notes in Computer Science*, pages 59–63. Springer, 1998.

[18] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudorandom bits. *SIAM J. on Computing*, 13:850–864, 1984.

[19] Blundo. C., A. De Santis, and A. Giorgio Gaggia. Probability of shares in secret sharing schemes. *Inf. Proc. Letters*, 72:169–175, 1999.

[20] M. Cafaro, R. Civino, and B. Masucci. On the Equivalence of Two Security Notions for Hierarchical Key Assignment Schemes in the Unconditional Setting. *IEEE Trans. Dependable Sec. Comput.*, 2014.

[21] R. M. Capocelli, A. De Santis, L. Gargano, and Vaccaro U. On the size of shares for secret sharing schemes. *J.of Cryptology*, 6:157–167, 1993.

[22] Aniello Castiglione, Luigi Catuogno, Aniello Del Sorbo, Ugo Fiore, and Francesco Palmieri. A secure file sharing service for distributed computing environments. *The Journal of Supercomputing*, 67(3):691–710, 2014.

[23] Arcangelo Castiglione, Alfredo De Santis, and Barbara Masucci. Hierarchical and Shared Key Assignment. In *17th International Conference on Network-Based Information Systems, NBIS 2014, IEEE*, pages 263–270, 2014.

[24] Arcangelo Castiglione, Alfredo De Santis, and Barbara Masucci. Key Indistinguishability vs. Strong Key Indistinguishability for Hierarchical Key Assignment Schemes. *IEEE Trans. Dependable Sec. Comput.*, 2015.

[25] Arcangelo Castiglione, Alfredo De Santis, Barbara Masucci, Francesco Palmieri, Aniello Castiglione, and Xinyi Huang. Cryptographic Hierarchical Access Control For Dynamic Structures. Manuscript accepted for publication in IEEE Transactions of Information Forensics and Security, 2016.

[26] Arcangelo Castiglione, Alfredo De Santis, Barbara Masucci, Francesco Palmieri, Aniello Castiglione, Jin Li, and Xinyi Huang. Hierarchical and Shared Access Control. *IEEE Transactions on Information Forensics and Security*, 11(4):850–865, 2016.

[27] Chin-Chen Chang, Ren-Junn Hwang, and Tzong-Chen Wu. Crypographic key assignment scheme for access control in a hierarchy. *Inf. Syst.*, 17(3):243–247, 1992.

[28] Hung-Yu Chien. Efficient Time-Bound Hierarchical Key Assignment Scheme. *IEEE Trans. Knowl. Data Eng.*, 16(10):1301–1304, 2004.

[29] Paolo D'Arco, Alfredo De Santis, Anna Lisa Ferrara, and Barbara Masucci. Security and Tradeoffs of the Akl-Taylor Scheme and Its Variants. In Rastislav Královic and Damian Niwinski, editors, *Mathematical Foundations of Computer Science 2009, 34th International Symposium, MFCS 2009, Novy Smokovec, High Tatras, Slovakia, August 24-28, 2009. Proceedings*, volume 5734 of *Lecture Notes in Computer Science*, pages 247–257. Springer, 2009.

[30] Paolo D'Arco, Alfredo De Santis, Anna Lisa Ferrara, and Barbara Masucci. Variations on a theme by Akl and Taylor: Security and Tradeoffs. *Theor. Comput. Sci.*, 411(1):213–227, 2010.

[31] Vanesa Daza, Javier Herranz, Paz Morillo, and Carla Ràfols. CCA2-Secure Threshold Broadcast Encryption with Shorter Ciphertexts. In Willy Susilo, Joseph K. Liu, and Yi Mu, editors, *Provable Security, First International Conference, ProvSec 2007, Wollongong, Australia, November 1-2, 2007, Proceedings*, volume 4784 of *Lecture Notes in Computer Science*, pages 35–50. Springer, 2007.

[32] A. De Santis, A. L. Ferrara, and B. Masucci. Efficient provably-secure hierarchical key assignment schemes. In *MFCS 2007, LNCS*, volume 4708, pages 371–382, 2007.

[33] A. De Santis, A. L. Ferrara, and B. Masucci. Efficient provably-secure hierarchical key assignment schemes. *Theoretical Comp. Sci.*, 412(41):5684–5699, 2011.

[34] Alfredo De Santis, Anna Lisa Ferrara, and Barbara Masucci. Cryptographic Key Assignment Schemes for any Access Control Policy. *Information Processing Letters*, 92(4):199–205, 2004.

[35] Alfredo De Santis, Anna Lisa Ferrara, and Barbara Masucci. A new key assignment scheme for access control in a complete tree hierarchy. In Øyvind Ytrehus, editor, *Coding and Cryptography, International Workshop, WCC 2005, Bergen, Norway, March 14-18, 2005. Revised Selected Papers*, volume 3969 of *Lecture Notes in Computer Science*, pages 202–217. Springer, 2005.

[36] Alfredo De Santis, Anna Lisa Ferrara, and Barbara Masucci. Enforcing the Security of a Time-Bound Hierarchical Key Assignment Scheme. *Inf. Sci.*, 176(12):1684–1694, 2006.

[37] Alfredo De Santis, Anna Lisa Ferrara, and Barbara Masucci. New Constructions for Provably-Secure Time-Bound Hierarchical Key Assignment Schemes. In Volkmar Lotz and Bhavani M. Thuraisingham, editors, *SACMAT 2007, 12th ACM Symposium on Access Control Models and Technologies, Sophia Antipolis, France, June 20-22, 2007, Proceedings*, pages 133–138. ACM, 2007.

[38] Alfredo De Santis, Anna Lisa Ferrara, and Barbara Masucci. New Constructions for Provably-Secure Time-Bound Hierarchical Key Assignment Schemes. *Theor. Comput. Sci.*, 407(1-3):213–230, 2008.

[39] D. E Denning and M. Smid. Key escrowing today. *Comm. Magazine, IEEE*, 32(9):58–68, 1994.

[40] Pantelis Frangoudis and George Polyzos. Security and performance challenges for user-centric wireless networking. *Communications Magazine, IEEE*, 52(12):48–55, 2014.

[41] Eduarda S. V. Freire and Kenneth G. Paterson. Provably Secure Key Assignment Schemes from Factoring. In Udaya Parampalli and Philip Hawkes, editors, *Information Security and Privacy - 16th Australasian Conference, ACISP 2011, Melbourne, Australia, July 11-13, 2011. Proceedings*, volume 6812 of *Lecture Notes in Computer Science*, pages 292–309. Springer, 2011.

[42] Eduarda S. V. Freire, Kenneth G. Paterson, and Bertram Poettering. Simple, Efficient and Strongly KI-Secure Hierarchical Key Assignment Schemes. In Ed Dawson, editor, *Topics in Cryptology - CT-RSA 2013 - The Cryptographers' Track at the RSA Conference 2013, San Francisco,CA, USA, February 25-March 1, 2013. Proceedings*, volume 7779 of *Lecture Notes in Computer Science*, pages 101–114. Springer, 2013.

[43] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *J. of the ACM*, 33(4):792–807, 1986.

[44] S. Goldwasser and S. Micali. Probabilistic encryption. *J. of Comp. and System Sci.*, 28:279–299, 1984.

[45] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. *SIAM J. Comput.*, 17(2):281–308, 1988.

[46] Lein Harn and Hung-Yu Lin. A Cryptographic Key Generation Scheme for Multilevel Data Security. *Computers & Security*, 9(6):539–546, 1990.

[47] J. Hastad, R. Impagliazzo, and L. A. Levin. A Pseudorandom Generator from any One-Way Function. *SIAM J. on Computing*, 28(4):1364–1396, 1999.

[48] Hui-Feng Huang and Chin-Chen Chang. A New Cryptographic Key Assignment Scheme with Time-Constraint Access Control in a Hierarchy. *Computer Standards & Interfaces*, 26(3):159–166, 2004.

[49] Min-Shiang Hwang. An improvement of a dynamic cryptographic key assignment scheme in a tree hierarchy. *Computers & Mathematics with Applications*, 37(3):19–22, 1999.

[50] Min-Shiang Hwang. Cryptanalysis of ycn key assignment scheme in a hierarchy. *Information Processing Letters*, 73(3):97–101, 2000.

[51] Giuseppe F Italiano. Finding Paths and Deleting Edges in Directed Acyclic Graphs. *Information Processing Letters*, 28(1):5–11, 1988.

[52] M. Ito, A. Saito, and T. Nishizeki. Secret sharing scheme realizing general access structure. In *Proc. of IEEE Global Telecommunication Conf. Globecom 87*, pages 99–102, 1987.

[53] J. Katz and M. Yung. Characterization of Security Notions for Probabilistic Private-Key Encryption. *J. of Cryptology*, 19:67–95, 2006.

[54] Jon Kleinberg and Eva Tardos. *Algorithm Design*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

[55] Johannes A La Poutré and Jan van Leeuwen. Maintenance of Transitive Closures and Transitive Reductions of Graphs. In *Graph-theoretic concepts in computer science*, pages 106–120. Springer, 1988.

[56] Derrick H Lehmer. An extended theory of lucas' functions. *Annals of Mathematics*, pages 419–448, 1930.

[57] Horng-Twu Liaw and Chin-Laung Lei. An optimal algorithm to assign cryptographic keys in a tree structure for access control. *BIT Numerical Mathematics*, 33(1):46–56, 1993.

[58] H.T. Liaw, S.J. Wang, and C.L. Lei. A Dynamic Cryptographic Key Assignment Scheme in a Tree Structure. *Computers & Mathematics with Applications*, 25(6):109 – 114, 1993.

[59] Iuon-Chang Lin, Min-Shiang Hwang, and Chin-Chen Chang. A New Key Assignment Scheme for Enforcing Complicated Access Control Policies in Hierarchy. *Future Generation Computer Systems*, 19(4):457 – 462, 2003.

Selected papers from the IEEE/ACM International Symposium on Cluster Computing and the Grid, Berlin-Brandenburg Academy of Sciences and Humanities, Berlin, Germany, 21-24 May 2002.

[60] Stephen J. MacKinnon, Peter D. Taylor, Henk Meijer, and Selim G. Akl. An Optimal Algorithm for Assigning Cryptographic Keys to Control Access in a Hierarchy. *IEEE Trans. Computers*, 34(9):797–802, 1985.

[61] Hwang Min-Shiang. A Cryptographic Key Assignment Scheme in a Hierarchy for Access Control. *Math. Comput. Model.*, 26(2):27–31, July 1997.

[62] M. Naor and O. Reingold. Number-theoretic constructions of efficient pseudo-random functions. *J. of the ACM*, 51(2):231–262, 2004.

[63] L. Qiu, Y. Zhang, F. Wang, M. Kyung, and H. R. Mahajan. Trusted computer system evaluation criteria. In *National Computer Security Center*. Citeseer, 1985.

[64] Ravi S. Sandhu. Cryptographic Implementation of a Tree Hierarchy for Access Control. *Inf. Process. Lett.*, 27(2):95–98, 1988.

[65] A. Shamir. How to share a secret. *C. ACM*, 22(11):612–613, 1979.

[66] Mohamed Shehab, Elisa Bertino, and Arif Ghafoor. Efficient hierarchical key generation and key diffusion for sensor networks. In *Proceedings of the Second Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, SECON 2005, September 26-29, 2005, Santa Clara, CA, USA*, pages 76–84. IEEE, 2005.

[67] D. R. Stinson. An explication of secret sharing schemes. *Design, Codes and Cryptography*, 2:357–390, 1992.

[68] Qiang Tang and Chris J. Mitchell. Comments On a Cryptographic Key Assignment Scheme. *Computer Standards & Interfaces*, 27(3):323–326, 2005.

[69] B. Toxen. The NSA and Snowden: securing the all-seeing eye. *Comm. of the ACM*, 57(5):44–51, 2014.

[70] Wen-Guey Tzeng. A Time-Bound Cryptographic Key Assignment Scheme for Access Control in a Hierarchy. *IEEE Trans. Knowl. Data Eng.*, 14(1):182–188, 2002.

[71] Xiaoshuang Xing, Tao Jing, Wei Zhou, Xiuzhen Cheng, Yan Huo, and Hang Liu. Routing in User-Centric Networks. *Communications Magazine, IEEE*, 52(9):44–51, 2014.

[72] J. Yeh, R. Chow, and R. Newman. A Key Assignment for Enforcing Access Control Policy Exceptions. In *Proc. of the International Symposium on Internet Technology*, pages 54–59, 1998.

[73] Xun Yi and Yiming Ye. Security of Tzeng's Time-Bound Key Assignment Scheme for Access Control in a Hierarchy. *IEEE Trans. Knowl. Data Eng.*, 15(4):1054–1055, 2003.