OPTIMIZATION TECHNIQUES AND NEW METHODS FOR BROADCAST ENCRYPTION AND TRAITOR TRACING SCHEMES

A DISSERTATION SUBMITTED TO
THE DEPARTMENT OF COMPUTER ENGINEERING
AND THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

By
Murat Ak
December, 2012

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Assist. Prof. Dr. Ali Aydın Selçuk(Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Assoc. Prof. Dr. İbrahim Körpeoğlu

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Assist. Prof. Dr. Alper Şen

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Prof. Dr. Fazlı Can

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Assoc. Prof. Dr. Ali Doğanaksoy

Approved for the Graduate School of Engineering and Science:

Prof. Dr. Levent Onural
Director of the Graduate School

ABSTRACT

OPTIMIZATION TECHNIQUES AND NEW METHODS FOR BROADCAST ENCRYPTION AND TRAITOR TRACING SCHEMES

Murat Ak
Ph.D. in Computer Engineering
Supervisor: Assist. Prof. Dr. Ali Aydın Selçuk
December, 2012

In the last few decades, the use of digital content increased dramatically. Many forms of digital products in the form of CDs, DVDs, TV broadcasts, data over the Internet, entered our life. Classical cryptography, where encryption is done for only one recipient, was not able to handle this change, since its direct use leads to intolerably expensive transmissions. Moreover, new concerns regarding the commercial aspect arised. Since digital commercial contents are sold to various customers, unauthorized copying by malicious actors became a major concern and it needed to be prevented carefully. Therefore, a new research area called *digital rights management (DRM)* has emerged. Within the scope of DRM, new cryptographic primitives are proposed. In this thesis, we consider three of these: *broadcast encryption (BE)*, *traitor tracing (TT)*, and *trace and revoke (T&R)* schemes and propose methods to improve the performances and capabilities of these primitives. Particularly, we first consider profiling the recipient set in order to improve transmission size in the most popular BE schemes. We then investigate and solve the optimal free rider assignment problem for one of the most efficient BE schemes so far. Next, we attempt to close the non-trivial gap between BE and T&R schemes by proposing a generic method for adding traitor tracing capability to BE schemes and thus obtaining a T&R scheme. Finally, we investigate an overlooked problem: privacy of the recipient set in T&R schemes. Right now, most schemes do not keep the recipient set anonymous, and everybody can see who received a particular content. As a generic solution to this problem, we propose a method for obtaining anonymous T&R scheme by using anonymous BE schemes as a primitive.

Keywords: Broadcast encryption, traitor tracing, digital rights management.

ÖZET

YAYIN ŞİFRELEMEDE VE HAİN TAKİBİNDE ENİYİLEMELER VE YENİ YÖNTEMLER

Murat Ak
Bilgisayar Mühendisliği, Doktora
Tez Yöneticisi: Yrd. Doç. Dr. Ali Aydın Selçuk
Aralık, 2012

Özellikle son yirmi yıl içerisinde dijital içeriğin kullanımı oldukça arttı. CD'ler, DVD'ler, TV yayınları, İnternet gibi çok sayıda dijital urun formları hayatımıza girdi. Şifrelemenin tek bir kullanıcıdan tek bir kullanıcıya şeklinde modellendiği klasik kriptografi bu değişime tam anlamıyla ayak uyduramadı, çünkü doğrudan klasik kriptografi kullanımı çoklu alıcı kümesine gönderimlere uygun değildi ve bu gönderilerin fazla büyümesine yol açmakta. Dahası, işin ticari yanı da düşünüldüğünde, yeni kaygılar ortaya çıkıyor. Ticari dijital içerikler çok sayıda müşteriye satılabildiği ve kopyalanmaları kolay olduğu için izinsiz kopyalanmalarını engellemek önem arzediyor. Tam da bu nedenlerle dijital hak yönetimi adında yeni bir araştırma alanı ortaya çıktı. Ve bu alanın çerçevesinde yeni kriptografik primitif yöntemler önerildi. Bu tezde, bu yöntemlerden üçünü, yayın şifreleme, hain tespiti, ve izleme ve iptal yöntemlerini, ele alıyoruz ve bu yöntemlerin performanslarını ve yapabildiklerini artırmaya yönelik metotlar ortaya koyuyoruz. Öncelikle en popüler yayın şifreleme yöntemlerinde kullanıcıların profillerini hesaba katarak gönderi maliyetinin düşürülmesini öneriyoruz. Daha sonra, halen en verimli yayın şifreleme yöntemlerden bir tanesi için en iyi bedava alıcı yerleştirme algoritması vererek maliyetin önemli ölçüde düşürülebileceğini gösteriyoruz. Bir sonraki çalışmamızda yayın şifreleme yöntemlerine hain tespit mekanizması eklemenin jenerik bir yolunu vererek yayın şifreleme ile izleme ve iptal yöntemleri arasındaki boşluğu ortadan kaldırıyoruz. Son olarak izleme ve iptal yöntemlerinde uzun zamandır gözardı edilmiş gizlilik problemini inceliyoruz. Şaşırtıcı biçimde yayın şifreleme yöntemleri için dahi gizlilik çok yakında yayınlanan birkaç makale dışında gözardı edildi. Dolayısıyla halihazırdaki

yayın şifreleme yöntemleri, gönderinin yapıldığı kişilerin kim olduğunu gizlemiyor ve kimin hangi dijital içeriğe ulaşabildiği yayınla birlikte açıktan gönderiliyor. Bu konuda gizliliği sağlayan ilk anonim izleme ve iptal yöntemini de öneriyoruz.

Anahtar sözcükler: Yayın şifreleme, hain tespiti, dijital hak yönetimi.

Acknowledgement

I would like to express my sincere gratitudes to my supervisor, Dr. Ali Aydın Selçuk, who has been an incredible and inspiring mentor. I consider myself very fortunate to have worked under his supervision. During all these years, he has always been available and ready for discussions. He never withheld his support and encouragement even during times of slow progress. He guided throughout technical problems I had encountered, he motivated me by his incredible positive attitude all the time. I learned quite a lot from him, not only about research, technical writing, and other academic staff, but also about moral values, ethics, and in general, about life. I have always considered him as a role model both as a researcher and as a person, and I will definitely continue to do so. I would like to thank him for everything he has done for me.

I also want to thank Dr. Kamer Kaya, who has been a great colleague and a sincere friend. He was also like a second mentor to me during the time we worked together. Particularly, the first two works in this thesis were the results of our fruitful collaboration with Dr. Kaya.

I am also thankful to Dr. Serdar Pehlivanoğlu for his efforts and patience during our collaboration that resulted the last two works in this thesis.

Fortunately, I had the opportunity to gain a great deal of cryptography knowledge apart from the scope of this thesis, thanks to our invaluable discussions with my dear friend Dr. Turgut Hanoymak.

I feel blessed that I had the privilege to spend so many years in Bilkent. I think it is impossible to adequately express the huge effects of Bilkent University on my personal development thanks to so many brilliant and inspiring people around. So, I would like to thank every single person who contributed to the foundation and development of Bilkent University, especially the late Prof. İhsan Doğramacı.

No matter how much I write, I can never thank them enough, but still, I want to express my special thanks and blessings to my dear parents. Without their unending genuine love and continuous support, my life would be incomplete.

This work was supported in part by TÜBİTAK (The Scientific and Technological Research Council of Turkey) Grants 108E150 and 111E213.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the advances in new technologies, the field of cryptography evolved even faster in the last few decades. Before 70s, there was only symmetric key encryption where we have one sender and one receiver, who have to somehow know the exact same key before they can communicate securely. In the mid-70s, public key cryptography (PKC) came into the scene and earned a well-deserved reputation. Before PKC, it seemed unbelievable that two actors who had not even met before can communicate secretly. And in 90s and the last decade, many cryptographic primitives are introduced for situations where PKC and old symmetric cryptography fell short. Digital rights management (DRM) is an umbrella term for technologies that allow digital contents to be secured over insecure channels and it is exactly one of the situations where PKC is not enough and people did not have to wait long for schemes that fulfill its requirements. Although it was more genius and elegant, PKC was still dealing with one-to-one encryption by nature. That is, encryption was being made by one sender to be decrypted by one receiver. In early 90s, broadcast encryption is introduced in order to make efficient encrypted transmissions to groups of users at once. Shortly after that, traitor tracing and trace and revoke methods followed.

1.1 Broadcast Encryption

Since it will be the main cryptographic method in the center of this thesis, let us first explain broadcast encryption briefly. Broadcast encryption (BE) is a cryptographic primitive that enables secure transmission of data to a dynamically changing large set of users such that only an authorized subset can decrypt it.

The usage of BE ranges from protecting recordable digital content in multimedia applications such as pay-TV, secure audio/video streaming and Internet multicasting, to file system security. Basically, whenever access control needs to be imposed on a one-way communication channel, BE is good alternative to employ. This key role of BE makes it a useful tool in digital rights management (DRM) technology. Especially in the last two decades new application areas have emerged that greatly benefit from BE, such as content protection [1, 2], multicasting promotional material and low cost pay-per-view events [3], multi-certificate revocation/validation [4] and dynamic group key management [5, 6, 7, 8, 9].

The users of a BE system are given a set of pre-installed, long-term keys, typically in a set-top box. These keys are later used to encrypt the broadcast sessions such that only the set of authorized users, i.e., the users with the appropriate long-term keys, can decrypt the broadcast. The users who are authorized to receive a particular broadcast are called *privileged* (or *subscriber*) whereas the remaining non-authorized users are called *revoked* (or *non-subscriber*).

The particular design of a BE system varies according to the system characteristics, such as the size of the user domain, required security level, available bandwidth, and hardware capabilities. In the traditional setting, the amount of long-term storage is very limited as it has to be tamper resistant, the communication channel is one way, and the devices are stateless in the sense that no additional long-term storage is possible.

Note that for communication channels that are two-way, more effective solutions than embedding long-term keys are possible. However, most of the broadcasting communication channels are typically one way, such as satellite channels, and the user decoders are modeled as stateless, meaning that they have no typical long-term memory.

1.2 Related Work on Broadcast Encryption

The idea of BE is introduced by Berkovits [10] in 1991. However the model of Fiat and Naor [11] is celebrated to be the first formal model of BE. They introduced the *resiliency* concept for BE, and called a scheme *k-resilient* if it is secure against any k revoked users working together, so that they would not be able to decrypt the encrypted broadcast message. They also described a scheme that required every receiver to store $O(k \log k \log n)$ keys and the center to broadcast $O(k^2 \log^2 k \log n)$ messages where n is the total number of users. Later on, fully resilient schemes dominated the BE research and these schemes became obsolete.

In 1999, the logical key hierarchy (LKH) was proposed independently by Wallner et al. [5] and Wong et al. [6]. According to LKH, the receivers were being associated with the leaves of a tree, and a unique key is associated with each node of the tree. Then, each receiver is given the keys of the nodes on the path from the corresponding leaf to the root. Although being originally proposed for secure Internet multicast, LKH was quite useful for BE. Recognizing this fact, Abdalla et al. [3] used LKH to design a BE scheme and reduced key storage complexity to logarithmic scale in terms of the number of receivers, namely to $O(\log n)$ while achieving $O(n)$ transmission overhead.

In their seminal paper, [12], Naor et al. proposed the renowned subset difference (SD) scheme. The SD scheme decreased the transmission overhead to $O(r)$ while keeping the key storage $O(\log^2 n)$ by employing one-way functions.

Later two important variants of SD was proposed. The layered subset difference (LSD) scheme, which was proposed by Halevy and Shamir [13]. Their optimized LSD scheme has a transmission overhead of $O(\log n \log\log n)$ and a key storage of $O(r \log\log n)$. Goodrich, Sun and Tamassia [14] introduced the stratified subset difference (SSD) scheme, which has $O(r \log n/\log\log n)$ transmission overhead and $O(\log n)$ key storage complexity. Horwitz presented an analysis of [11, 13, 12] in his survey [15]. The SD scheme has recently gained popularity in applications as well and is included in the next-generation DVD standard [16].

Despite this popularity, the SD scheme did not long remain as the only most efficient scheme. In 2005, Jho et al. [17] proposed the Punctured Interval (PI) scheme. The PI scheme is also a subset cover framework scheme but with a different subset structure. The subsets are designed as intervals with possible skippings on a straight line on which users are thought to be placed virtually. Originally, in [18, 19, 17, 20], the PI scheme is employed alongside two other subset cover schemes called C-basic chain and cascade chain and they are treated as one combined scheme. Basic chain subsets are defined as all intervals with a length less than a bound. On top of the basic chain and PI subsets, these schemes also employ the cascading idea to bring extra transmission cost efficiency by grouping subsets from different layers together. This combined scheme outperforms the SD scheme in terms of transmission overhead but with a slightly larger key storage requirement.

On the other hand, a number of different approaches to the BE problem have been introduced in the public key setting. Typically, these schemes rather depend on number theoretic structures and there are no predefined subsets. They allow keys to be generated on-the-fly for any privileged user subset. In 2005, Boneh et al. [21] used bilinear maps and the bilinear decision Diffie-Hellman exponent problem to design a public key BE system. Their scheme has constant size private key and offers a trade-off between ciphertext and public key sizes, product of which can be linear in the number of receivers.

The BE problem is also investigated in the context of identity-based encryption where, briefly stated, public keys are the identities of user. Boneh and Hamburg provided a framework for ID-based BE schemes in [22]. As a result of the increasing interest around 2008, several identity based and public-key BE schemes have been proposed [23, 24, 25, 26, 27].

1.3 Performance of BE schemes and Improvement Methods

In different settings, some concerns such as user domain size, security, bandwidth, or hardware may be more important than others. However, usually, two concerns are inherent in almost all BE systems. First, the amount of key storage must be adequate because the long-term secure storage size at the receiver side is very limited since it has to be tamper resistant. Second, the amount of additional data sent along with the content through the communication channel, called the transmission overhead, must be adequate because of the limited nature of the bandwidth of communication channels.

1.3.1 Free riders

In all traditional BE schemes, by default, it is assumed that all unauthorized receivers must be revoked in an encrypted broadcast. However, Abdalla et al. [3] pointed out that this assumption could be relaxed for some applications, and the transmission overhead can be reduced significantly by allowing a limited amount of free riders. So, in certain cases, a number of non-subscribers can be allowed to decrypt the broadcast in order to reduce the overall cost of the system. Such users are called *free riders*. In this case, there needs to be a limit on the number of free riders allowed, and the question is how to optimally use this given free rider quota.

1.3.2 Profiles

User profiling is the concept of monitoring data on preferences and interests of the users in the system in order to serve them more effectively. It is broadly used in various areas such as web mining [28] and broadcasting and multicasting [29, 30, 31].

In the BE literature, traditionally, the users are assumed to be identical in the sense that they are taken to be equally likely to be interested in any particular broadcast. However, in practice every user has a certain type of interest, some being more interested in sport events, some in movies, some in entertainment, etc. If these user profiles are taken into account, they can provide some critical information to optimize the operations of a BE system.

1.4 Traitor Tracing and Trace & Revoke Systems

As we mentioned above, broadcast encryption (BE) schemes handle the task of encrypting content for groups of users. However, this might not be enough for certain DRM systems. Because when a malicious user forges a decoder that circumvents the access control used by the content distribution system, BE schemes can do nothing about this. Such a decoder created by an adversary is called a *pirate decoder*, the users that divulge their keys to the adversary are called *traitor*s, and the divulged keys are called *traitor key*s. The sender may want to restrict this type of behavior since such adversarial behavior introduces additional unauthorized receivers in the system. *Traitor tracing* is such a deterrence mechanism where an authority is capable of performing an analysis to any working pirate decoder and recovering at least one of the traitor keys that was used in its construction. Traitor tracing emerged first in the work of Chor, Fiat and Naor [32] as a solution to the problem that we mentioned above.

We categorize the traitor tracing mechanisms as *non-black-box* if it is possible to extract the keys from the decoder through reverse-engineering techniques. Such schemes that have been proposed in the literature include [33, 34]. However, in many settings, the non-black-box approach is inapplicable for many reasons, e.g., it may be expensive or deterred through obfuscation or the tracer may only have remote access to the decoder. We call black-box tracing if the tracing authority interacts with the pirate decoder in a black-box manner: querying the decoder with input and observing the response of the decoder. Majority of the works, [35, 36, 37, 32, 38, 39, 40, 41], in the traitor tracing literature supports black-box tracing.

Trace and Revoke Schemes: The ultimate goal in a content distribution system would be combining traitor tracing and broadcast encryption so that any receiver key found to be compromised in a tracing process would be revoked in the future transmissions. This is introduced by Naor and Pinkas in [42]. However, it is not possible to achieve this trivially, and a naive combination of both mechanisms would severely fail as discussed in the subsequent works [43, 44, 12]. The subset cover framework of [12] leads to a number of schemes [14, 13] which rely on combinatorial structures and support somewhat weak tracing in the symmetric setting (the tracing does not guarantee to identify a traitor but rather disables the pirate decoder). This weakness leads to a new type of attack called Pirate Evolution in [45]. The studies on trace and revoke schemes followed in the public key setting with notable examples of by Boneh et al. [43] and by Furukawa and Attrapadung [46]. We leave the discussion of traitor tracing and trace and revoke systems to Chapter 5.

Chapter 2

Preliminaries

In this chapter we will briefly present the preliminary knowledge needed and the definitions that we will adhere to throughout the thesis.

2.1 Broadcast Encryption Model

In order to represents users in a BE system, we will consider n recipients that we will denote with the set U. We suppose a broadcasting center that continually makes broadcasts to subsets of U. This subset possibly changes every time a new broadcast is made.

2.1.1 Structure of a Broadcast Encryption Scheme

Since we will explain these algorithms formally when needed in the following chapters, we briefly explain the general structure here. A broadcast encryption scheme can be defined in terms of three algorithms.

- A key distribution algorithm for creating keys and assigning them to receivers in the construction time of the BE system.

- An encryption algorithm that encrypts the message such that only intended user can decrypt.
- A decryption algorithm that will be run by receivers and succeed only if the receiver is in the intended set. Otherwise it must give no non-trivial information about the message.

2.1.2 Security Definitions

Security of a BE scheme is about the confidentiality of the data being sent. As in classical cryptography, it is defined in the form of an indistinguishability game. In the relevant chapters we will give this game in detail. Basically, a revoked user must be unable to distinguish the message being sent from an arbitrary message from the message space. That is, if a revoked user listens to a broadcast channel and obtains an encrypted message c, and if it is provided two messages m_1 and m_2 one being the message in c ($c = \texttt{BroadcastEncrypt}(m_b)$) the probability of guessing b correctly must be close to $1/2$. By "close" we mean that it must be different than $1/2$ by only a negligible amount in terms of the system parameters. We leave detailed explanation of security to the relevant chapters.

2.1.3 Evaluation Parameters

As in all computational systems, the speed of the key distribution, encryption and decryption algorithms are important parameters by default. The speed of the key distribution algorithm is the least sensitive one, however, because it will typically be run only once when the system is being prepared beforehand. The speed of the decryption algorithm is typically more important since the device that will run the decryption algorithm will be less powerful. For example, a broadcasting center's device is usually much more powerful than a set-top box in a house.

However, there are two evaluation parameters more specific to broadcast encryption schemes: key storage and transmission overhead. Key storage is basically the size of the key per receiver. This storage must be minimized since the keys must be tamper-resistant in order to prevent illegal actions by malicious actors like stealing the keys and forging pirate decoders. Transmission overhead is the extra cost of broadcast encryption compared to insecure broadcasts.

Key storage and transmission overhead has a trade-off. As more keys are stored, it is more likely that a broadcast encryption can be made with less transmission overhead. There are theoretical works that show this trade-off such as [47]. However, it is easy to see this trade-off intuitively: If we can store a unique key for every possible subset, we can use only one encryption for each broadcast because for every subset, we already have a key. On the other hand, if we were allowed to store only one key per user, we have to encrypt the content as many times the number of users in the intended recipient set. Both cases are infeasible in almost all broadcast encryption systems since the number of users is typically huge and we simply have neither that much key storage space in the user devices to carry out the first idea nor that much bandwidth capacity to carry out the second.

2.2 Traitor Tracing

Although BE schemes fulfill the encryption and decryption functionalities for DRM systems, there are still a few problems. One important problem is that a malicious party who obtains a number of user keys can forge a pirate decoder. Broadcasting centers need to be able to investigate these decoders and identify the keys that are used to forge it. Traitor tracing schemes are designed for this purpose.

Traitor tracing methods are defined on top of a broadcast encryption system in the form of a tracing algorithm. Tracing algorithm gets the pirate

decoder and its known properties such as its success ratios when transmissions are made to particular subsets. Usually, pirate decoders are modeled as a black box. That is, we assume that we cannot reverse engineer a pirate box and easily get the keys inside it. So, we are only allowed to make transmissions and observe the successes and failures of the pirate decoder. This is called black box tracing.

2.2.1 Tracing capability

The success of a tracing algorithm is measured by its tracing capability. This capability has two parameters:

- Number of users allowed to collude to make the pirate decoder
- The probability of the tracing algorithm to successfully identify at least one traitor key.

Trace and Revoke System: A trace and revoke (T&R) system can be thought as a BE scheme together with a tracing method. This is, in a sense, the ultimate goal in a DRM system and it is the best type of system we can achieve today.

Chapter 3

Broadcast Encryption with Client Profiles

In this chapter, we study the problem of achieving a more efficient BE system in the presence of provided user preference information. Our approach works by constructing the subset structure of a CS or SD system according to the given set of subscriber profiles. We first analyze the relationship between the transmission overhead of a BE scheme and the distribution of the user profiles. After proving several key results, we give two optimal algorithms for the CS scheme with one broadcast type. Then we generalize our approach by proposing a similarity metric for the CS and SD schemes with multiple broadcast types. Theoretical and experimental results show that the approach can significantly reduce the transmission overhead of the CS-based and SD-based BE schemes. This reduction can especially be remarkable when the proposed approach is used in conjunction with an optimal free rider assignment [48, 49].

3.1 User Profiles

User profiling has been used in a number of different applications. Recent works in broadcasting literature have made use of user profiles in order to increase broadcast efficiency in several aspects [50, 31, 30]. Similarly, web-user profiles have been heavily studied to serve individual users more effectively [28, 51]. User profiling was also used in multicast key management [52] where the key distribution tree is optimized according to the members' expected stay time in a session.

In a recent study that utilizes subscriber profiles for BE efficiency, D'Arco and De Santis [53] proposed a method for efficient key storage, the other important performance metric for a BE system besides the transmission overhead, in presence of non-uniform revocation probabilities. The authors assumed these probabilities to be given and used this information to give fewer keys to users with a higher probability of revocation.

The idea of allowing free riders in a broadcast to get better performance was introduced by Abdalla, Shavitt and Wool [3]. They investigated the usage of free riders and developed the basic intuitions about their effective assignment. Ramzan and Woodruff [49] recently proposed an algorithm to optimally choose the set of free riders in a CS scheme to minimize the transmission overhead. Ak, Kaya, and Selcuk [48] extended this work to the SD scheme.

To the best of our knowledge, user profiles have not been used in the BE literature to reduce the transmission overhead despite the fact that the subset cover framework is by its nature an excellent context for utilizing user profiles.

3.2 Subset Cover Framework and the CS and SD Schemes

A subset-cover BE scheme first generates a collection of subsets from the user set and associates a different long-term key with each subset. Then, every user in the system is installed with the long-term keys of the subsets he is included in.

To broadcast a message to a privileged user set P, the sender finds a cover C from the subset collection such that

$$P = \cup_{S \in C} S$$

and encrypts the message using the keys of the subsets in C. The number of subsets in C, i.e., $|C|$, is called the *transmission cost* which is one of the main performance parameters for a BE scheme.

Both the CS and SD schemes obtain the user subsets by organizing the users in a binary tree. These schemes differ in the way they define their subsets.

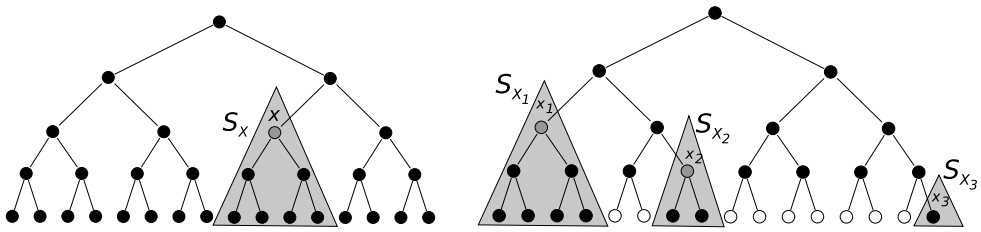
In the CS scheme, the leaves of the subtree rooted at a node $x \in T$ correspond to a subset in the system. That is, for every node x, a subset is defined as

$$S_x = \{v | v \text{ is a leaf of } T(x)\}$$

where $T(x)$ denotes the subtree rooted at node x. An example subset and an example cover are illustrated in Figure 3.1.

In the SD scheme, a subset is defined by two nodes x and y where y is a descendant of x in T. A subset $S_{x,y}$ is the set of leaves that are descendants of x but not descendants of y. More formally, for every non-leaf node x, and every descendant y of x, a subset is defined as

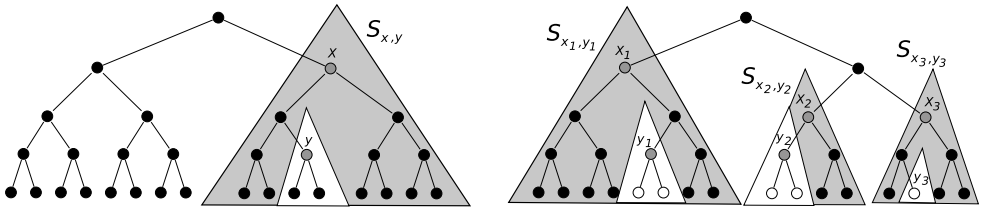
$$S_{x,y} = \{v | v \text{ is a leaf node, } v \in T(x) \text{ and } v \notin T(y)\}.$$



(a) A single CS subset (b) A CS cover

Figure 3.1: A simple subset and cover of the CS scheme. Revoked users are denoted by white leaves.

The total user set is also included as a subset in the SD scheme. An example subset and an example cover for the SD scheme are illustrated in Figure 3.2.



(a) An SD subset (b) An SD cover

Figure 3.2: A simple subset and cover of the SD scheme. Revoked users are shown with white leaves.

Note that every subset in the CS scheme is also a subset in the SD scheme. The SD scheme also has the advantage of covering the leaves of several subtrees at once by a single subset. The increased key storage complexity of the SD scheme is reduced by an intelligent key generation scheme employing a pseudo-random function [12].

3.3 Broadcast Encryption with User Profiles

As noted in Section 1, the original CS and SD schemes treat the users identically when organizing the key distribution tree. However, if we have information about the user preferences and interests, we can use this information

to group similar users together and make the BE scheme more efficient by constructing the subsets in a more clever way.

Consider a system supporting b different types of broadcasts where type j has a broadcast probability of q_j and $\sum_{j=1}^{b} q_j = 1$. Let $p_{u,j}$ denote the probability of user u subscribing to a broadcast of type j. We denote the *profile* of user u with the b-tuple $(p_{u,1}, p_{u,2}, \ldots, p_{u,b})$.

As described above, both CS and SD schemes use a binary tree T to organize the subsets and construct the cover. For a binary tree T, we will use r_T to denote its root and L_T to denote the set of its leaves. For a node $x \in T$, $par(x)$, $sib(x)$, $l(x)$ and $r(x)$ denote the parent, sibling, left child and right child of x in T, respectively. For a node x, let $p_{x,j}$ denote the probability of all users (leaves) in $T(x)$ subscribing to a type j broadcast, i.e.,

$$p_{x,j} = \prod_{u \in L_{T(x)}} p_{u,j}$$

where $L_{T(x)}$ is the set of leaves in the subtree with root x.

For clarity, we will investigate the cases $b = 1$ and $b \geq 1$ separately and we will use the terms *unitype* and *multitype* broadcast to refer to these cases, respectively.

3.3.1 Analysis of the CS Scheme with User Profiles

We will first investigate the unitype broadcast case. In this case, we will use p_u instead of $p_{u,1}$ to denote the probability of user u being a subscriber. Let $P(S_x)$ be the probability of a CS subset S_x being used in a cover.

Lemma 3.3.1 *In a CS tree, if x is a node other than the root, then*

$$P(S_x) = p_x - p_x p_{sib(x)} = p_x - p_{par(x)}.$$

If x is the root r_T, then $P(S_x) = p_{r_T} = \prod_{u \in L_T} p_u$.

Proof For a node x other than the root, if S_x is in the cover, all the users in $L_{T(x)}$ must be subscribers. Also, there must be at least one non-subscriber in $L_{T(sib(x))}$, because otherwise $S_{par(x)}$ would be in the cover instead of S_x.

Note that if x is the root, S_x will be in the cover if and only if each user in L_T is a subscriber, which happens with probability $\prod_{u \in L_T} p_u$.

Let $E_{CS}(T)$ denote the expected cover size for a CS tree T.

Theorem 3.3.2 *For a CS tree T,*

$$E_{CS}(T) = \sum_{x \in L_T} p_x - \sum_{x \notin L_T} p_x. \tag{3.1}$$

Proof The expected cover size for the CS scheme is equal to the sum of $P(S_x)$ over all $x \in T$. Hence,

$$E_{CS}(T) = \sum_{x \in T} P(S_x) = \sum_{x \in T,\ x \neq r_T} \left(p_x - p_{par(x)}\right) + p_{r_T}. \tag{3.2}$$

Note that since T is a binary tree, for each non-leaf x, p_x appears three times in the summation where one of them is positive and the other two are negative. And for a leaf x, the contribution to the summation is one p_x. Hence, (3.2) is equal to (3.1).

Theorem 3.3.2 can be extended to the multitype case where $b \geq 1$:

Theorem 3.3.3 *For a CS scheme with $b \geq 1$ broadcast types, the expected cover size is*

$$E_{CS}(T) = \sum_{j=1}^{b} q_j \left(\sum_{x \in L_T} p_{x,j} - \sum_{x \notin L_T} p_{x,j} \right) \tag{3.3}$$

Proof The expected cover size is the weighted average of the expected cover sizes for all broadcast types. Since each type j has probability q_j, $E_{CS}(T)$ is equal to (3.3).

3.3.2 Analysis of the SD Scheme with User Profiles

As in Section 3.3.1, we begin with an analysis for the unitype SD scheme: Let $P(S_{x,y})$ be the probability of an SD subset $S_{x,y}$ being used in a cover, and let

$$P(S_{*,y}) = \sum_{x \text{ is an ancestor of } y} P(S_{x,y}).$$

Lemma 3.3.4 *For a non-leaf, non-root node $y \in T$,*

$$P(S_{*,y}) = p_{sib(y)}(1 - p_{l(y)})(1 - p_{r(y)}), \tag{3.4}$$

and for a leaf $y \in L_T$

$$P(S_{*,y}) = p_{sib(y)}(1 - p_y). \tag{3.5}$$

Proof If $S_{x,y}$ is used in the cover, for a node y and one of its ancestors x, all the users in $L_{T(sib(y))}$ must be subscribers. Furthermore, if y is a non-leaf, non-root node, there must be at least one non-subscriber in both $L_{T(l(y))}$ and $L_{T(r(y))}$.

If y is a leaf node and $S_{x,y}$ is in the cover $sib(y)$ must be a subscriber and y cannot. Hence (3.4) and (3.5) follow.

Let $E_{SD}(T)$ denote the expected cover size for an SD tree.

Theorem 3.3.5 *In an SD tree T,*

$$E_{SD}(T) = \prod_{y \in L_T} p_y + \sum_{y \in L_T} \big(p_{sib(y)}(1 - p_y)\big) + \sum_{\substack{y \notin L_T \\ y \neq r_T}} \big(p_{sib(y)}(1 - p_{l(y)})(1 - p_{r(y)})\big). \tag{3.6}$$

Proof The expected cover size for the SD scheme, $E_{SD}(T)$, is equal to the sum of $P(S_{*,y})$ for all $y \in T$ except the root r_T. Besides, if all of the users subscribe to a broadcast, which happens with probability $\prod_{y \in L_T} p_y$, the cover size will be one. Hence,

$$E_{SD}(T) = \sum_{y \in T - \{r_T\}} P(S_{*,y}) + \prod_{y \in L_T} p_y.$$

By substituting (3.4) and (3.5) for $P(S_{*,y})$, (3.6) follows.

Theorem 3.3.5 can be extended to the multitype case:

Theorem 3.3.6 *For an SD scheme with $b \geq 1$ broadcast types, the expected cover size is*

$$E_{SD}(T) = \sum_{j=1}^{b} q_j E_{SD}(T, j) \tag{3.7}$$

where

$$E_{SD}(T, j) = \prod_{y \in L_T} p_{y,j} + \sum_{y \in L_T} \left(p_{sib(y),j}(1 - p_{y,j}) \right) + \sum_{\substack{y \notin L_T \\ y \neq r_T}} \left(p_{sib(y),j}(1 - p_{l(y),j})(1 - p_{r(y),j}) \right)$$

is the expected cover size for the broadcast type j with probability q_j.

Proof The expected cover size is the weighted average of the expected cover sizes for all broadcast types. Since each type j has probability q_j, $E_{SD}(T)$ is equal to (3.7).

3.4 Optimal CS Tree Construction

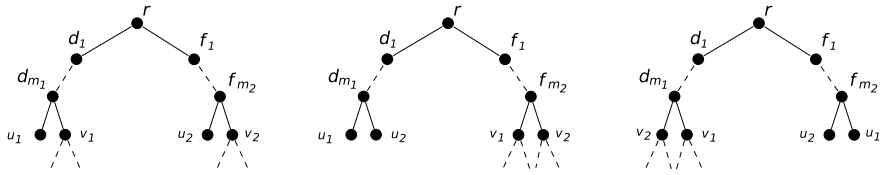
In this section, we will give two optimal tree construction algorithms for the unitype CS scheme. We will assume that for users $u_1, u_2, \cdots, u_n$, the subscription probabilities are $p_{u_1} \geq p_{u_2} \geq \cdots \geq p_{u_n}$; i.e., the users are indexed with

respect to their subscription probabilities in decreasing order. We say that a CS tree is *optimal* if it minimizes the expected cover size.

We will consider the optimal CS tree organization problem for two different settings: First, the CS tree has to be a balanced tree, and second, the CS tree is not necessarily balanced. We will refer the former as the balanced setting and the latter as the general setting. Lemma 3.4.1 below applies to both settings:

Lemma 3.4.1 *In a CS scheme with unitype broadcast, there exists an optimal tree where u_1 and u_2, the two users with the highest subscription probabilities, are siblings.*

Proof First recall that for any binary tree T, balanced or unbalanced, $E_{CS}(T) = \sum_{x \in L_T} p_x - \sum_{x \notin L_T} p_x$. Let T be an optimal tree with the minimum expected cover size. If u_1 and u_2 are siblings in T then we are done. Otherwise let v_1 and v_2 be the siblings of u_1 and u_2, respectively. Since we are investigating both settings, balanced and general, v_1 and v_2 may be internal nodes of T. Let r be the first common ancestor of u_1 and u_2 and let $path(r, u_1) = (r, d_1, d_2, \ldots, d_{m_1}, u_1)$ and $path(r, u_2) = (r, f_1, f_2, \ldots, f_{m_2}, u_2)$ be the paths from r to u_1 and u_2, respectively, as shown in Figure 3.3(a).



(a) Before swap (b) Swap u_1 and v_2 (c) Swap u_2 and v_1

Figure 3.3: Structure of $T(r)$ before and after the swap operations.

Note that $p_{u_1} p_{v_1}$ is a factor of each term in $\{p_{d_1}, p_{d_2}, \ldots, p_{d_{m_1}}\}$, and $p_{u_2} p_{v_2}$ is a factor of each term in $\{p_{f_1}, p_{f_2}, \ldots, p_{f_{m_2}}\}$. Let $D = \sum_{i=1}^{m_1} p_{d_i}/(p_{u_1} p_{v_1})$ and $F = \sum_{i=1}^{m_2} p_{f_i}/(p_{u_2} p_{v_2})$. Let $V(u_1, u_2)$ be the combined set of nodes on

$path(d_1, d_{m_1})$ and $path(f_1, f_{m_2})$. The expected cover size can be written as

$$\begin{aligned} E_{CS}(T) &= \sum_{x \in L_T} p_x - \sum_{x \notin L_T \cup V(u_1,u_2)} p_x - \sum_{x \in V(u_1,u_2)} p_x \\ &= \sum_{x \in L_T} p_x - \sum_{x \notin L_T \cup V(u_1,u_2)} p_x - (p_{u_1} p_{v_1} D + p_{u_2} p_{v_2} F) \end{aligned}$$

where the first two terms do not change if we swap u_1 and v_2, or u_2 and v_1, as shown in Figures 3.3(b) and 3.3(c), respectively. We have two cases:

1. $\mathbf{D < F}$: Let T' be the tree obtained by swapping u_1 and v_2 as in Figure 3.3(b). Since we have $p_{u_1} \geq p_{v_2}$ and $p_{u_2} \geq p_{v_1}$, the difference

$$\begin{aligned} E_{CS}(T) - E_{CS}(T') &= p_{v_1} p_{v_2} D + p_{u_1} p_{u_2} F - p_{u_1} p_{v_1} D - p_{u_2} p_{v_2} F \\ &= p_{u_2} F (p_{u_1} - p_{v_2}) - p_{v_1} D (p_{u_1} - p_{v_2}) \end{aligned}$$

   is non-negative. Given that T is optimal, we must have that $p_{u_1} = p_{v_2}$ and swapping u_1 and v_2 does not change the expected cost.

2. $\mathbf{D > F}$: Let T' be the tree obtained by swapping v_1 and u_2 as in the Figure 3.3(c). Since we have $p_{u_2} \geq p_{v_1}$ and $p_{u_1} \geq p_{v_2}$, the difference

$$\begin{aligned} E_{CS}(T) - E_{CS}(T') &= p_{u_1} p_{u_2} D + p_{v_1} p_{v_2} F - p_{u_1} p_{v_1} D - p_{u_2} p_{v_2} F \\ &= p_{u_1} D (p_{u_2} - p_{v_1}) - p_{v_2} F (p_{u_2} - p_{v_1}) \end{aligned}$$

   is non-negative. Given that T is optimal, we must have that $p_{u_2} = p_{v_1}$ and swapping u_2 and v_1 does not change the expected cost.

3. $\mathbf{D = F}$: Let T' be the tree obtained by swapping u_1 and v_2 as in Figure 3.3(b). (Note that we could choose to swap u_2 and v_1, as well.) Since we have $p_{u_1} \geq p_{v_2}$ and $p_{u_2} \geq p_{v_1}$, the difference

$$\begin{aligned} E_{CS}(T) - E_{CS}(T') &= p_{v_1} p_{v_2} D + p_{u_1} p_{u_2} F - p_{u_1} p_{v_1} D - p_{u_2} p_{v_2} F \\ &= p_{u_2} F (p_{u_1} - p_{v_2}) - p_{v_1} D (p_{u_1} - p_{v_2}) \end{aligned}$$

   is non-negative. Given that T is optimal, we must have that $p_{u_2}(p_{u_1} - p_{v_2}) - p_{v_1}(p_{u_1} - p_{v_2}) = 0$ which implies $(p_{u_2} - p_{v_1})(p_{u_1} - p_{v_2}) = 0$. (Here,

note that if we had chosen to swap u_2 and v_1 we would end up with this same equation, by symmetry.) Then, either $p_{u_2} = p_{v_1}$ or $p_{u_1} = p_{v_2}$. If $p_{u_2} = p_{v_1}$, swapping u_2 and v_1 does not change the expected cost. If $p_{u_1} = p_{v_2}$, in this case, swapping u_1 and v_2 does not change the expected cost. So in either case, we can come up with an optimal tree where u_1 and u_2 are siblings.

Hence, for all three cases we can say that the two nodes with maximum subscription probabilities can be paired in a tree that preserves the optimality.

3.4.1 Optimality for Balanced Trees

In this section we give the optimal CS tree construction algorithm with the balanced tree constraint. We assume that n is a power of 2 throughout the discussion in this section.

Lemma 3.4.2 *For a unitype CS scheme, there exists an optimal balanced CS tree where the pairs* $(u_1, u_2), (u_3, u_4), \cdots, (u_{n-1}, u_n)$ *are siblings of each other.*

Proof From Lemma 3.4.1, we know that there exists an optimal balanced tree T such that (u_1, u_2) are siblings. Similar to the proof of Lemma 3.4.1, starting with T, the other users can be paired as siblings by swapping operations by an iterative process that starts with (u_3, u_4). Note that u_3 and u_4 are the users with the two maximum subscription probabilities excluding u_1 and u_2; hence the optimality is preserved after the swap operations. Since the tree T is balanced at the beginning, each leaf T will have a leaf sibling at any time.

Now we are ready to prove the main result for the balanced case.

Theorem 3.4.3 *In a unitype CS scheme with the balanced tree constraint, sorting the users in the leaf level with respect to their subscription probabilities gives the minimum expected cover size.*

Proof Let $T^{(k)}$ denote an optimal balanced CS tree of depth k whose leaf nodes are grouped as stated in Lemma 3.4.2 as $(u_1, u_2), (u_3, u_4), \ldots, (u_{n-1}, u_n)$ for a given user set. Let $H^{(k)}$ denote the balanced tree of depth k on the same user set, obtained by ordering the leaves according to the sorted p_{u_i} values. We will use induction on the depth of the tree to prove that $E_{CS}(T^{(k)}) = E_{CS}(H^{(k)})$ for any k.

For the basic case, for any set of two nodes, obviously $E_{CS}(T^{(1)}) = E_{CS}(H^{(1)})$. Now assume that the claim is also true for all balanced trees with depth less than k. For the tree $T^{(k)}$ for a given user set, let T' denote the subtree of depth $k-1$ which has the paired nodes $u_{2i-1,2i}$ as its leaves, with probabilities $p_{u_{2i-1,2i}} = p_{u_{2i-1}} p_{u_{2i}}$, for $1 \leq i \leq n/2$. Let $H^{(k-1)}$ denote the balanced tree obtained by sorting the same set of nodes, $\{u_{1,2}, \ldots, u_{n-1,n}\}$. By induction, $E_{CS}(T') \geq E_{CS}(H^{(k-1)})$. Also from (3.1),

$$E_{CS}(T^{(k)}) = E_{CS}(T') + \sum_{i=1}^{n} p_{u_i} - 2\sum_{i=1}^{n/2} p_{u_{(2i-1)(2i)}}$$

$$E_{CS}(H^{(k)}) = E_{CS}(H^{(k-1)}) + \sum_{i=1}^{n} p_{u_i} - 2\sum_{i=1}^{n/2} p_{u_{(2i-1)(2i)}}.$$

Hence, $E_{CS}(T^{(k)}) \geq E_{CS}(H^{(k)})$; and since $T^{(k)}$ is optimal, $H^{(k)}$ is also optimal.

3.4.2 Optimality for the General Setting

The optimal construction for the general setting is also based on equation (3.1) and Lemma 3.4.1, which are true independent of the tree's being balanced.

Let T_i be a tree with one user node u_i. Let $T \circ T'$ denote the union of two trees constructed by adding a new root r and connecting T and T' to r as the

left and right subtrees. The UNI-GEN CLUSTER algorithm below takes the subscription probabilities as inputs and constructs a broadcast tree with the minimum expected cover size in a style similar to Huffman trees [54].

Algorithm 1 UNI-GEN CLUSTER

1: $\mathcal{T} \leftarrow \{T_1, T_2, \ldots, T_n\}$, , where T_i is the tree containing just one node u_i
2: **while** $|\mathcal{T}|$ is not equal to 1 **do**
3: Find the pair $T, T' \in \mathcal{T}$ with maximum p_{r_T} and $p_{r_{T'}}$
4: Construct the merged tree $T'' = T \circ T'$
5: $\mathcal{T} \leftarrow \mathcal{T} \setminus \{T, T'\}$
6: $\mathcal{T} \leftarrow \mathcal{T} \cup T''$
7: **return** $\mathcal{T}$

The algorithm works in a bottom-up fashion. At each iteration, two trees T and T' with the largest p_{r_T} and $p_{r_{T'}}$ are selected. These trees are extracted from the queue, and a new tree $T'' = T \circ T'$ with a new root $r_{T''}$ is inserted where $p_{r_{T''}} = p_{r_T} p_{r_{T'}}$. The optimality proof of the tree obtained by this algorithm is given in Theorem 3.4.4:

Theorem 3.4.4 *For a unitype CS scheme, the tree obtained by the* UNI-GEN CLUSTER *algorithm is optimal with the minimum expected cover size.*

Proof Let $T^{(k)}$ denote an optimal CS tree with k leaves where u_1 and u_2 are connected as siblings as stated in Lemma 3.4.1, for a given user set. Let $H^{(k)}$ denote the tree with the same k leaves constructed by the algorithm UNI-GEN CLUSTER. We will use induction on the number of leaves in the tree to prove that $E_{CS}(T^{(k)}) = E_{CS}(H^{(k)})$ for any k.

For the basic case, for any set of two nodes, obviously $E_{CS}(T^{(2)}) = E_{CS}(H^{(2)})$. Now assume that the claim is also true for all trees with $k-1$ or fewer leaves. For the tree $T^{(k)}$ for a given user set, let T' denote the tree with $k-1$ leaves obtained by merging u_1 and u_2 into a new node u_{12}, with probability $p_{u_{12}} = p_{u_1} p_{u_2}$. Let $H^{(k-1)}$ be the tree constructed by the UNI-GEN CLUSTER

algorithm from the same set of leaves. By induction, $E_{CS}(T') \geq E_{CS}(H^{(k-1)})$. Also from (3.1),

$$E_{CS}(T^{(k)}) = E_{CS}(T') + p_{u_1} + p_{u_2} - 2p_{u_{12}}$$
$$E_{CS}(H^{(k)}) = E_{CS}(H^{(k-1)}) + p_{u_1} + p_{u_2} - 2p_{u_{12}},$$

and it follows that $E_{CS}(T^{(k)}) \geq E_{CS}(H^{(k)})$. We know $T^{(k)}$ is optimal, therefore $H^{(k)}$ is optimal.

3.5 The Case of Multitype Broadcasts

In multitype BE schemes, we cannot simply group the users with respect to their subscription probabilities since there are b different subscription probabilities for each user. Nevertheless, if we place similar users closer in the tree, the number of subtrees containing them will increase, hence smaller covers can be obtained. We will first focus on the probability of two users being interested in a common broadcast. If two users' probabilities of being interested in the same broadcast are both high, we will say that these two users are *similar*. We define the similarity of two user profiles as the weighted sum of the products of their probabilities over different broadcast types:

$$\mathrm{Sim}(u, v) = \sum_{j=1}^{b} q_j p_{u,j} p_{v,j}.$$

Assuming that the user subscription decisions are independent, the similarity between two users is the probability of both subscribing to a common broadcast.

Extending the formulation for individual users to groups of users, we define the similarity of groups of users as follows: We call a set of users *similar* if the probability of all users being interested in the same broadcast is high. Let T and T' be two trees containing disjoint sets of users as their leaves. Then the

similarity of these trees are

$$\mathrm{Sim}(T, T') = \sum_{j=1}^{b} q_j p_{r_T,j} p_{r_{T'},j}$$

where

$$p_{r_T,j} = \prod_{u \in L_T} p_{u,j}.$$

3.5.1 The Balanced Tree Algorithm

The Multi-Bal Cluster algorithm below clusters the set of users according to the Sim metric and organizes them as the leaves of a balanced binary tree. It works by arranging the tree in levels. It starts with the bottom level by organizing the most similar users in pairs. Then, at every level, pairs of nodes/subsets are matched and clustered according to their similarities.

Algorithm 2 Multi-Bal Cluster

```
1: T ← {T_1, T_2, ..., T_n}, where T_i is the tree containing just one node u_i
2: S ← {}
3: while |T| is not equal to 1 do
4:    while T is not empty do
5:       Find the pair T, T' ∈ T with maximum Sim(T, T')
6:       Construct the merged tree T'' = T ∘ T'
7:       T ← T \ {T, T'}
8:       S ← S ∪ {T''}
9:    T ← S
10:   S ← {}
11: return T
```

The algorithm works in a bottom-up fashion; in the first iteration, it clusters the pairs of leaves starting with the most similar pair. The pairs in these clusters will be the siblings in the resulting tree. In the next iteration, these clusters are paired and this process continues until just one cluster remains and the tree is constructed. Note that the algorithm constructs a balanced binary tree since the list $\mathcal{T}$ always contains trees of the same depth. For

$b = 1$, the MULTI-BAL CLUSTER algorithm sorts the users with respect to their subscription probabilities, which we know to give the optimal CS tree for $b = 1$.

3.5.2 The General Algorithm

The similarity approach can also be used for the general setting where the CS and SD trees need not be balanced.

Algorithm 3 MULTI-GEN CLUSTER

1: $\mathcal{T} \leftarrow \{T_1, T_2, \ldots, T_n\}$, where T_i is the tree containing just one node u_i
2: **while** $|\mathcal{T}|$ is not equal to 1 **do**
3: Find the pair $T, T' \in \mathcal{T}$ with maximum $\mathrm{Sim}(T, T')$
4: Construct the merged tree $T'' = T \circ T'$
5: $\mathcal{T} \leftarrow \mathcal{T} \setminus \{T, T'\}$
6: $\mathcal{T} \leftarrow \mathcal{T} \cup \{T''\}$
7: **return** $\mathcal{T}$

As in the balanced setting, the MULTI-GEN CLUSTER algorithm constructs the tree in a bottom-up fashion. Similar to its unitype counterpart UNI-GEN CLUSTER, at each iteration the algorithm chooses and merges the most similar pair.

3.6 Experimental Results

We tested the performance of the proposed algorithms against the standard BE approach by running a large number of experiments on synthetically generated user profiles. The user profiles were carefully generated with various characteristics to be representatives of a wide variety of applications.

We experimented with a population of $n = 1024$ users. Each user profile contains b subscription probabilities for some $1 \leq b \leq 10$. For each broadcast

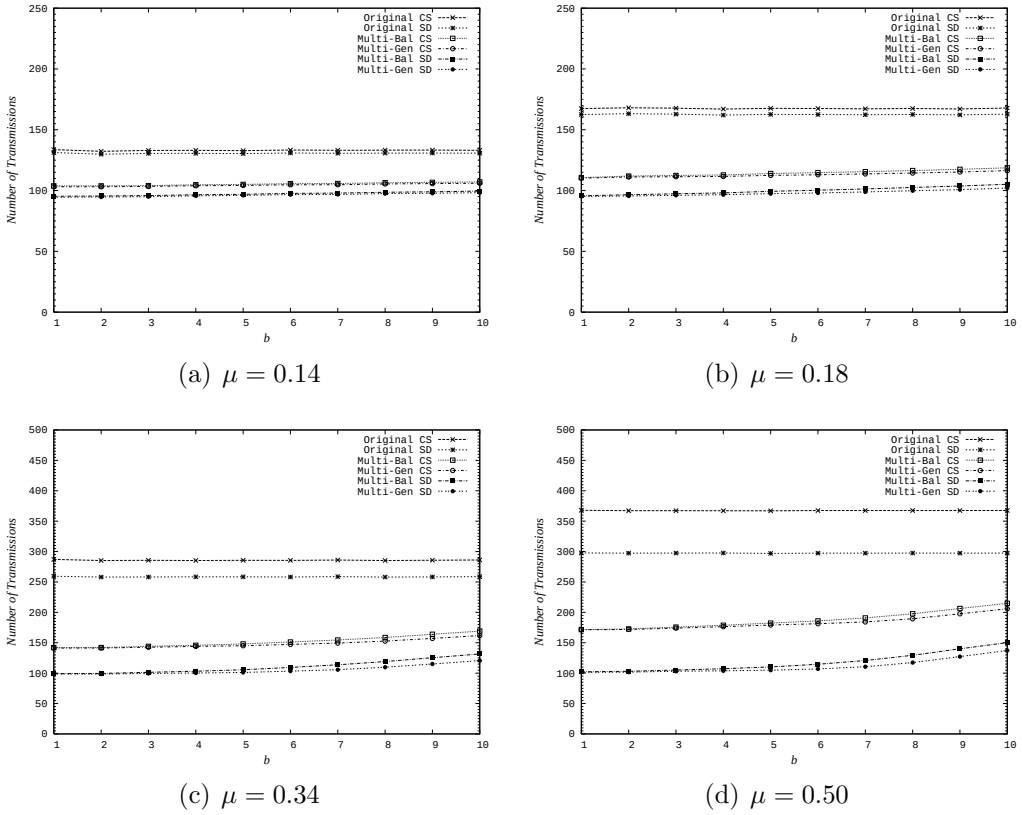
type j, the subscription probabilities $p_{i,j}$ are randomly generated by using a bimodal density function based on two uniform distributions with respective means of $\mu_1 = 0.9$ and $\mu_2 = 0.1$ to represent the interested and uninterested user populations, respectively. The overall population mean, μ, is determined according to the weight of the interested users in the population. For each set of experiments, we compared the average transmission costs of the basic CS and SD schemes with those obtained by subscriber profiling. In the experiments, the broadcast types are taken to be equally likely with a probability of $q_j = 1/b$ for each $1 \leq j \leq b$.

The experimental results are summarized in Figure 3.4 where the transmission costs of the basic and similarity-based CS and SD schemes are compared. The results show that utilizing the user profiles with the given similarity metric can reduce the transmission cost significantly. For the balanced-tree CS scheme, the reduction rate is about 20–45% for larger values of b and more than 20–50% for smaller values of b. The improvements are even more significant for the balanced-tree SD scheme, with 25–55% improvement for larger values of b and 25–65% for smaller b values. The cost reduction rates get higher with larger population means.

The improvement rates for the generalized (unbalanced) algorithm are only slightly better than those of the balanced tree algorithm for smaller values of b and the population mean; however as the value of b gets larger and the population mean increases, the generalized algorithm provides better improvement rates that allow up to an additional 5% reduction in the transmission costs.

3.7 Using Similarity Approach with Free Riders

Free riders are the users who are able to decrypt a broadcast session although they are not subscribed to it. Some free riders can be allowed in a BE system



(a) $\mu = 0.14$ (b) $\mu = 0.18$

(c) $\mu = 0.34$ (d) $\mu = 0.50$

Figure 3.4: Transmission costs of the CS and SD schemes in their basic form and with subscriber profiling. Four different plots are given for four different values of the interested user density, 5%, 10%, 30% and 50%, making the population mean $0.14, 0.18, 0.34$ and 0.5, respectively. The results indicate that significant reductions are possible over the basic CS and SD schemes by the proposed algorithms. On the other hand, there is only a slight difference between the balanced-tree algorithms and their generalized counterparts.

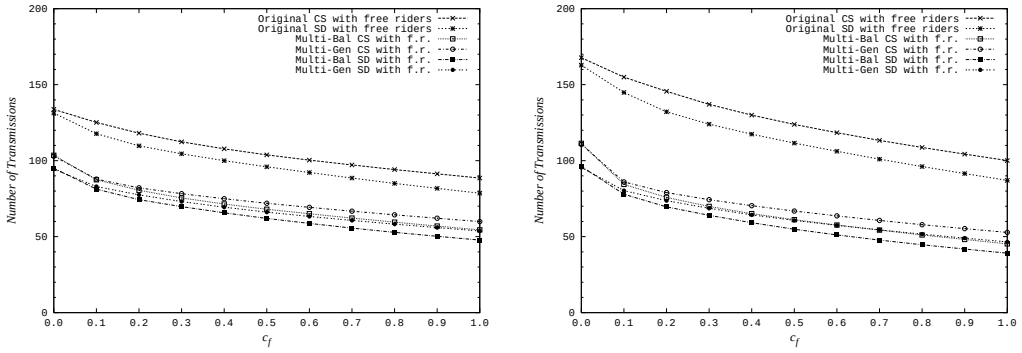
in order to lower the transmission cost by relaxing the restriction that the cover must exactly match the privileged user set. Free riders must be assigned carefully in order to reduce the cost effectively. Optimal free rider assignment algorithms for the CS and SD schemes have recently been given by Ramzan and Woodruff [49] and Ak et al. [48], respectively.

Our proposed similarity-based organization algorithms can be expected to be even more effective when a few free riders can be tolerated. Our approach aims to obtain large subsets by taking a set of consecutive users as subscribers. Hence, if a few remaining non-subscribers can be tolerated as free riders in such a sequence of subscribers, a larger and fully privileged subset can be obtained, leading to more compact covers.

Let f denote the number of free riders that can be allowed, and let c_f denote the *free rider ratio*, $f/(n-r)$, where n and r are the total number of users and the number of revoked users, respectively. We tested the performance of our algorithms with a given number of free riders by a large number of simulation experiments with $n = 1024$ and $0.1 \leq c_f \leq 1.0$, where the user profiles are generated with the same parameters used for the experiments with no free riders in Section 3.6.

Figures 3.5,3.6,3.7 and 3.8 show the results for the basic and the similarity-based CS and SD schemes with free riders for $b = 1, 2, 5, 10$ broadcast types. The plots demonstrate the improvements in the transmission cost according to the free rider ratio c_f. The results show that significant savings can be achieved by using the similarity approach and allowing a very limited number of free riders. A sharp decrease in the transmission cost can be obtained by using the similarity approach with a free rider ratio of just 10%, while the improvement rates of the basic CS and SD schemes appear to be linear with c_f.

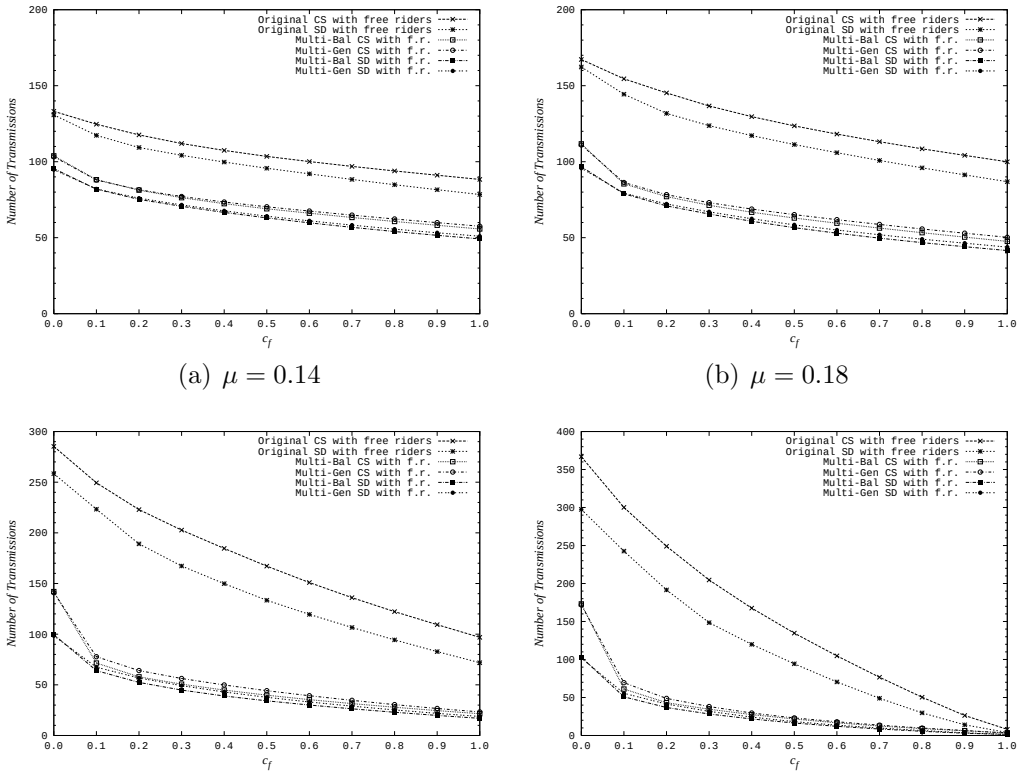
The experiments show that allowing a free rider ratio of 10% reduces the transmission cost of the similarity-based CS scheme by $40 - 70\%$ and the similarity-based SD scheme by $35 - 55\%$, whereas the transmission cost of the



(a) $\mu = 0.14$ (b) $\mu = 0.18$

(c) $\mu = 0.34$ (d) $\mu = 0.50$

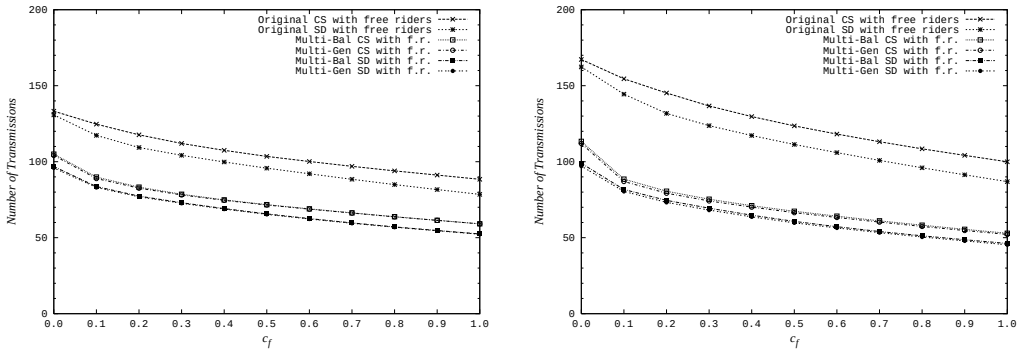
Figure 3.5: Transmission costs of the CS and SD schemes with free riders, in their basic form and with user profiling, where the number of broadcast types is $b = 1$. The results indicate that a sharp decrease in the transmission cost is possible by allowing a limited number of free riders, especially for higher values of μ.



(a) $\mu = 0.14$ (b) $\mu = 0.18$

(c) $\mu = 0.34$ (d) $\mu = 0.50$

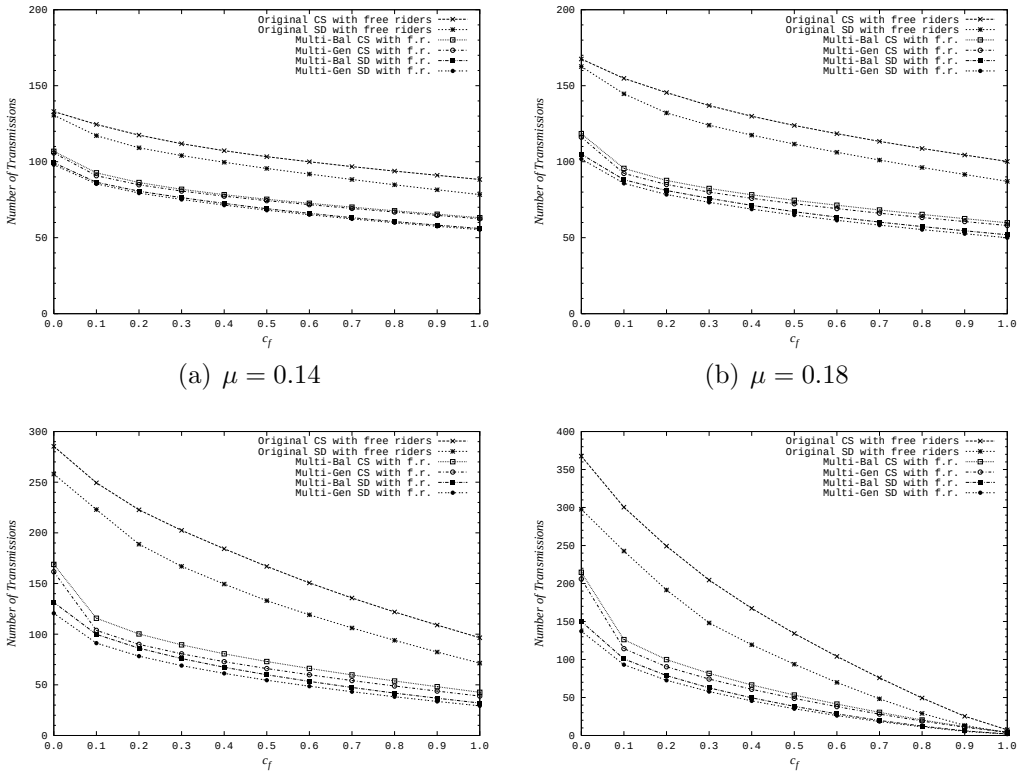
Figure 3.6: Transmission costs where $b = 2$.



(a) $\mu = 0.14$ (b) $\mu = 0.18$

(c) $\mu = 0.34$ (d) $\mu = 0.50$

Figure 3.7: Transmission costs where $b = 5$.



(a) $\mu = 0.14$ (b) $\mu = 0.18$

(c) $\mu = 0.34$ (d) $\mu = 0.50$

Figure 3.8: Transmission costs where $b = 10$.

original schemes are only reduced by 20%. As a result, the similarity-based CS scheme has $65-85\%$ lower cost than the original CS scheme and the similarity-based SD scheme has $60-80\%$ lower cost than the original SD scheme when a free rider ratio of 10% is allowed. The similarity approach becomes more effective at smaller values of b and at greater values of μ, which is consistent with the previous experiments with no free riders.

The balanced-tree and the generalized algorithms have similar transmission costs for a given number of free riders, while the generalized algorithms have a slight cost advantage over their balanced-tree counterparts.

3.8 Discussion

In this chapter, we analyzed the problem of reducing the transmission costs of subset-cover based BE schemes of CS and SD by utilizing information about user interests. We gave optimal algorithms for the CS scheme when only one type of broadcast exists. For the multitype case, we proposed a similarity approach which can be used in both CS and SD schemes. The simulation experiments showed that the proposed algorithms are effective and can provide significant reductions in the transmission complexity of a BE system. The gains obtained by the proposed algorithms turn out to be even more significant when a limited number of free riders can be tolerated in the system.

Chapter 4

Free Rider Optimization for Punctured Interval Broadcast Encryption Scheme

In this chapter, we study how to reduce the transmission cost of the Punctured Interval (PI) scheme [17] by effective use of *free riders*. In certain scenarios where allowing a limited number of non-privileged users (called free riders) to decrpyt the transmission, the center can shrink the size of the transmission significantly by making such a relaxation.

The idea of allowing free riders, which may be considered as a relaxation for the original BE problem, was introduced and investigated by Abdalla et al. [3]. Ramzan and Woodruff [49] proposed an algorithm to optimally choose the set of free riders to be allowed in the CS scheme [12]. Recently, Ak et al. [55] solved the same problem for the SD scheme [12].

In the following sections, we first give an algorithm for finding the the optimal placement of free riders for an instance specified by:

- a user set,

- the set of subscribers which is a subset of the user set,
- ratio of the number of free riders allowed to the number of revoked users.

We then propose a parametric heuristic that we call the top-down algorithm for the same problem which runs much faster than the optimal algorithm while decently reducing the transmission overhead. We also provide a hybrid solution which, in a sense, uses ideas from both the optimal algorithm and the top-down heuristic in order to obtain a trade-off between speed and reduction in transmission overhead.

4.1 Punctured Interval (PI) scheme

In this chapter, we will focus on the punctured interval (PI) scheme (a.k.a. skipping-chain scheme) of [20, 19, 17, 18]. We confine ourselves to the plain form of this scheme without combining it with the *C-basic chain* and *cascade chain* schemes. We assume that the PI scheme itself is used in a layered fashion, which we will describe in detail in Section 4.1.1.

The PI scheme is a subset cover scheme, and subsets are designed in the form of bounded-size *punctured intervals* on a virtual number line which can possibly have at most a certain number of skippings called *punctures*. So, it is parametrized with two parameters c and p which represent the bounds on the size of the punctured intervals and on the number of punctures, respectively. Specifically, subsets are designed as follows: First, users are thought on a line numbered from 1 to n. For every (i, j) with $1 \leq i \leq j \leq n$ and $j - i < c$, for every $\pi = \{x_1, x_2, \ldots, x_p\}$ with $i < x_k < j$ and $x_k < x_{k+1}$ for all $k \in \{1, \ldots, p\}$, $S_{i,j;\pi}$ is the set of users $\{u_i, \ldots, u_j\} \backslash \{u_{x_1}, u_{x_2}, \ldots, u_{x_p}\}$. For example, $S_{3,9;\{5,8\}}$ consists of users $3, 4, 6, 7, 9$ as shown in Figure 4.1.

The PI scheme works like any other scheme in the subset cover framework. For every subset, i.e., punctured interval, a key is made available to the users

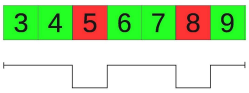


Figure 4.1: Sample subset $S_{3,9;\{5,8\}}$.

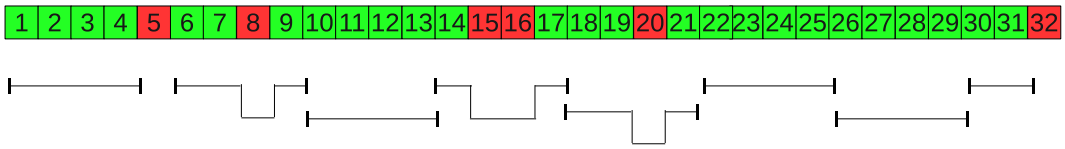


Figure 4.2: Sample cover with $c = 4$ and $p = 2$ for basic PI scheme. Red (dark) cells indicate revoked users.

in that subset and when a broadcast is to be made, it is encrypted with a set of keys, subsets of which covers the privileged user set. One may note that if we were to store one key per each subset in the form $S_{i,j;\pi}$, we would need to store too many keys in the receiver boxes. Particularly, $O(c^{p+2})$ keys would be needed per user. However, using one-way functions, the number of keys to store is reduced to $O(c^{p+1})$ in [17]. Note that this reduction becomes quite significant especially for large c and small p. Since key storage is not a concern in our work, we refer the readers who are interested in key storage cost to [17, 20, 19, 18].

When a broadcast is to be made, having all the subsets defined as above and keys distributed accordingly, what remains is to find the best cover for the privileged set, consisting of the predefined subsets. It is easy to see that the best cover can be found by going from the beginning to the end (1 to n) and including the next longest punctured interval successively [17]. A simple cover with such subsets is illustrated in Figure 4.2.

4.1.1 Layered PI scheme

In the basic PI scheme, when most of the users are privileged, there would be many consecutive full subsets in the cover. Therefore, [17] further adds layers to their scheme so that long intervals of users can be included to the

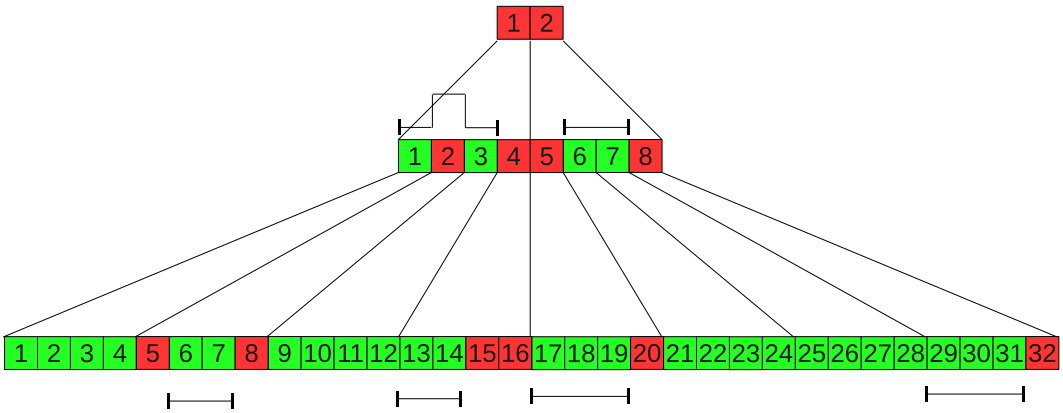


Figure 4.3: Sample cover with $c = 4$ and $p = 2$ for layered PI scheme. Dark cells indicate revoked users. Note how the number of subsets reduced when layers are introduced.

cover by fewer subsets. In the original form of the scheme [20, 19, 17, 18] layers consist of non-punctured long intervals for covering long sequences of consecutive privileged users. Here, we consider a generalized form of this approach for layering: We allow punctured intervals in the upper layers as well. More precisely, the idea is to group c consecutive users together and treat them as one user in the layer above. At the end, there will be $\lfloor \log_c n \rfloor + 1$ layers each having c times more users than the one above itself. The subsets in the upper layers are defined in the same way as the basic PI scheme but their keys are prepared independently.

Note that the key storage does not increase significantly: Since each layer will have $1/c$ times fewer users, the number of keys required by each layer decreases exponentially. Therefore, the total number of keys required increases only logarithmically. Throughout this work, we focus on this modified version of the layered PI scheme.

In the layered scheme, sets are also parametrized by their layers, and denoted by $S_{\ell;i,j;\pi}$ meaning the subset at layer ℓ, from node i to node j having punctures π between i and j.

In the layered PI scheme, privileged users are covered in a greedy manner

starting from the highest layer, covering as many fully privileged nodes as possible, and proceeding with the next layer. An example cover is shown in Figure 4.3.

Jho et al. [17] further employ cascading on top of the layered PI scheme to improve the scheme further. However, the performance improvement it brings becomes visible only if the ratio of revoked users is less than 0.1% as already shown emprically in [17]. Since we concentrate on relatively larger revoked user ratios, we confine ourselves to the layered PI scheme as we defined.

4.2 Problem Statement

One of the critical evaluation parameters regarding BE schemes is the transmission overhead. This cost is measured as the number of encryptions per a single broadcast. BE schemes in the subset cover framework perform one encryption per subset in the cover. Therefore, the transmission overhead is simply the number of subsets in the cover. Since the PI scheme is such a BE scheme and subsets are designed as punctured intervals, the transmission overhead that we focus on is the number of punctured intervals in a cover.

Note that by allowing a certain number of free riders and relaxing the constraint that privileged sets must be covered exactly, one may reduce the transmission overhead of the broadcast. The question is, given a fractional limit on the number of free riders, how should one use this free rider quota best in order to decrease the transmission overhead as much as possible?

Notation: We denote the set of all users with U, while denoting its two partitions, the privileged and revoked user sets in a particular broadcast, by P and R, respectively, where $n = |U|$, $\rho = |P|$, $r = |R|$. Throughout the chapter, we will assume that users are organized in a B-tree-like data structure, $\mathcal{T}$, to represent users as in Figure 4.3. In particular, $\mathcal{T}$ is in the form of a double array of nodes. We denote the ℓth layer by $\mathcal{T}[\ell]$ and its ith node by $\mathcal{T}[\ell][i]$.

Each node has the following information:

- number of users beneath the user i at layer ℓ, denoted by $\#usr(\mathcal{T}[\ell][i])$,
- number of privileged users beneath it, denoted by $\#pri(\mathcal{T}[\ell][i])$,
- number of revoked users beneath it, denoted by $\#rev(\mathcal{T}[\ell][i])$,
- index of the left and right children in the previous layer, denoted by $\mathcal{T}[\ell][i].left$ and $\mathcal{T}[\ell][i].right$, respectively.

Nodes in the first layer of $\mathcal{T}$ (that is, $\mathcal{T}[0]$) correspond to individual users. Nodes in the upper layers correspond to intervals of users arranged as explained in Section 4.1.1. We also denote the index of the top layer by ℓ_{max}. Note that $\ell_{max} = \lfloor \log_c n \rfloor + 1$. Thus, $\mathcal{T}[\ell_{max}]$ includes a single node, $\mathcal{T}[\ell_{max}][0]$, covering all users.

The allowed free riders to revoked users ratio is denoted by c_f and the number of free riders allowed becomes

$$fr = c_f \cdot r$$

The set of all subsets, denoted by $\mathcal{S}$, is defined according to the Layered PI scheme as explained in Section 4.1.1. Intervals on the layered structure is defined via three parameters, and denoted by $I_{\ell;i,j}$, where ℓ denotes the layer number whereas i and j are the start and end nodes.

Problem: Having stated all the needed notation, the problem is to find a cover $\mathcal{C} \subseteq \mathcal{S}$ with the smallest cardinality $|\mathcal{C}|$ where

$$P \subseteq \bigcup_{S_{\ell;x,y;\pi} \in \mathcal{C}} S_{\ell;x,y;\pi}$$

with

$$\left| \bigcup_{S_{\ell;x,y;\pi} \in \mathcal{C}} S_{\ell;x,y;\pi} \setminus P \right| \leq fr = c_f \cdot r$$

Throughout the chapter, we will assume that instances of the problem are already organized into a data structure, $\mathcal{T}$, that represents the users, layer nodes, child-parent relationships and revoked and privileged counts of the nodes.

4.3 Optimal Algorithm

In this section, we describe a dynamic programming solution for the problem stated in Section 4.2.

We define a subproblem $(I_{\ell;i,j}, f)$ as follows: What is the minimum cost to cover an interval $I_{\ell;i,j}$ with f free riders? For a particular subproblem instance, we record the solutions to subproblems in a four-dimensional table *cost* where each cell corresponds to the solution of a subproblem. In particular, $cost[\ell, i, j, f]$ denotes the minimal cost of covering the users beneath the interval $I_{\ell;i,j}$ by using exactly f free riders. The table is filled for all ℓ ranging from 0 to ℓ_{max}, i and j with $i \leq j$ ranging from 0 to $|\mathcal{T}[\ell]|$, and f ranging from 0 to fr.

4.3.1 The Algorithm

In this section, we will present OptimalPICover$_{c,p}$ algorithm that, for a given $\mathcal{T}$ instance and ratio of free riders, finds the minimum transmission cost possible by the free rider relaxation idea.

In the OptimalPICover$_{c,p}$ algorithm, only iterations through the subproblems are performed. Filling a single cell of the table *cost* is delegated to the FindCost procedure. Iterations are done in such an order that the costs for all subproblems are solved before the ones that use them. Algorithm goes from the lowest layer to the highest one and at each layer, subproblems are solved from right to left. The rationale behind this order in which the table

is filled will become more clear when we describe the recurrence relation in Section 4.3.2.

Algorithm 4 OPTIMALPICOVER$_{c,p}(\mathcal{T}, c_f)$

```
1: if c_f ≥ 1 then
2:     return 1
3: fr ← ⌊c_f · r⌋
4: for ℓ from 0 to ℓ_max do
5:     for j from |T[ℓ]| down to 0 do
6:         for i from j down to 0 do
7:             for f from 0 to fr do
8:                 cost[ℓ, i, j, f] ← FINDCOST(ℓ, i, j, f)
9: return cost[ℓ_max, 0, 0, fr]
```

The cost of a particular interval with a given number of free riders is calculated by the FINDCOST procedure (Algorithm 5). This procedure first checks the base cases:

- If the number of free riders allowed is more than the number of revoked users in the interval, it returns ∞ as the cost, because we do not want to allocate more than enough free riders.
- If there are no privileged users in the interval, it returns 0, because we obviously do not need to cover this interval.
- If the interval is a one-node interval:
  - If the number of free riders allowed is just enough for the revoked users under that node, it returns 1 meaning that the node is covered by itself as an interval.
  - If we do not have enough free riders, then it takes the cost from the layer below, which must be set already because of the direction of execution.

After handling all these basic cases, the procedure finally makes its core calculation in the final line, which we will explain separately. The FINDCOST

procedure is given in Algorithm 5.

Algorithm 5 FINDCOST(ℓ, i, j, f)

```
1: if #rev(I_{ℓ;i,j}) < f then
2:    return ∞
3: if #pri(I_{ℓ;i,j}) = 0 and f = 0 then
4:    return 0
5: if i = j then
6:    if f = #rev(T[ℓ][i]) then
7:       return 1
8:    if f < #rev(T[ℓ][i]) then
9:       if ℓ > 0 then
10:         ch_l ← T[ℓ][i].left
11:         ch_r ← T[ℓ][i].right
12:         return cost[ℓ − 1, ch_l, ch_r, fr]
13: return CompMinCost(ℓ, i, j, fr)
```

4.3.2 CompMinCost procedure

The last line of the FINDCOST procedure consists of a single call to the COMPMINCOST procedure. The aim of this minimization operation is to find the minimum cost to cover $I_{\ell;i,j}$ with f free riders, by using the costs of other subproblems which have been calculated and recorded already. This procedure basically implements the recurrence relation of the dynamic programming being performed. To describe this recurrence relation, we introduce the following notation: An *arrangement* of a subproblem $(I_{\ell;i,j}, f)$ is defined by a five-tuple (s, PI, f_l, f_d, f_r).

- The first two components (s, PI) describe the *pivot*, which is the candidate first punctured interval of a cover of the subproblem $(I_{\ell;i,j}, f)$ from the left. The node $i \leq s \leq j$ is the starting index of the pivot and PI stands for the punctured interval combination used. PI varies over all possible punctured interval combinations, which we denote by $\mathcal{PI}$, specified by c and p. Of course, there are boundary constraints, $i \leq s$ and $s + |PI| \leq j$, so that the pivot completely lies in $I_{\ell;i,j}$.

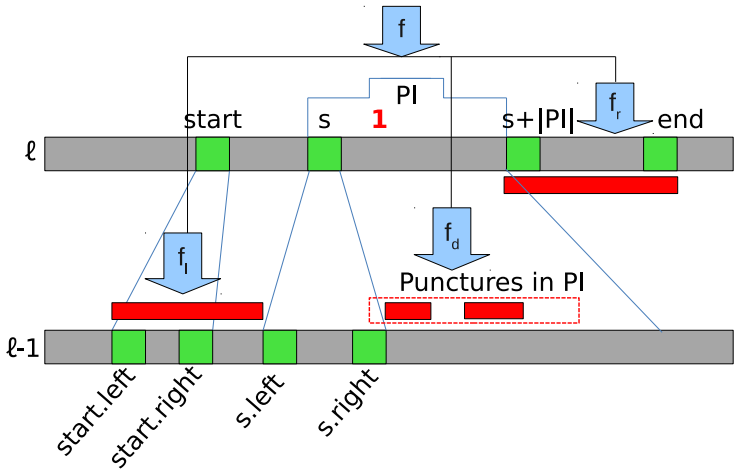


Figure 4.4: Illustration of an arrangement (s, PI, f_l, f_d, f_r) with a pivot. In the picture, the subproblem being examined is $(I_{\ell;start,end}), f$ and this is an arrangement that possibly can lead to an optimal solution for this particular subproblem. For each arrangement, there are four components that must be added up: a single cost for the pivot PI itself, the cost of covering left handside at the layer below, the cost of covering right handside at the same layer, and the cost of covering the punctured places at the layer below.

- The last three components, (f_l, f_d, f_r), describe a partitioning of the free rider quota into three parts: intervals on the left, underneath, and right of the pivot, respectively. For these values, we have the constraint that $f_l + f_d + f_r = f - \#rev(s, PI)$, where $\#rev(s, PI)$ is the number of revoked users underneath the pivot. Because the pivot is assumed to be taken into the cover, the revoked users under it are taken as free riders.

All possible arrangements are considered and the one that leads to the least cost is taken. One additional possibility is that there are no punctured intervals at layer ℓ in the optimal solution, which we will consider as another arrangement. I.e., we must also check whether delegating the whole subproblem to the lower layer would be better: whether $cost[\ell - 1, i.left, j.right, f]$ is better than all arrangements *with* a pivot at layer ℓ. Minimization is performed over all possible arrangements together with this additional one. For an illustration of an arrangement with a pivot, see Figure 4.4.

In order to calculate the cost of an arrangement (s, PI, f_l, f_d, f_r), first, we allocate $\#rev(s, PI)$ free riders for taking the pivot to the cover. The rest of the free rider quota is partitioned into three parts:

- left of the pivot in $I_{\ell;i,j}$ (f_l)
- punctures of the pivot (f_d)
- right of the pivot in $I_{\ell;i,j}$ (f_r)

All possible distributions of the $f - \#rev(s, PI)$ free riders as $f_l + f_d + f_r$ are considered and the cost associated with an arrangement (s, PI, f_l, f_d, f_r) is then calculated as the sum of four components:

1: A cost of 1 for the pivot itself.

c_l: Cost of covering left of the pivot borrowed from the lower layers with f_l free riders. This cost is already calculated.

c_r: Cost of covering right of the pivot at the same layer with f_r free riders. This cost is also already calculated.

c_d: Cost of covering the punctures of the pivot with f_d free riders borrowed from the lower layer. Unlike the previous three, this cost will not be readily available when there are more than one punctures of PI. In such cases, a last nested minimization function must be performed over all possible distributions of the f_d free riders to the punctures of PI.

The values of c_l and c_r can be directly retrieved from the dynamic programming table because according to the iteration order, they are already calculated and recorded before. Calculation of c_d requires one additional minimization over all possible distributions of the free rider quota f_d to the punctures of PI. Before presenting the pseudocode, we first describe the output of the CompMinCost procedure as a mathematical formula below. Here, $\mathcal{PI}$ denotes the set of all possible punctured interval combinations.

$$\textsc{CompMinCost}(\ell, i, j, f) = \min_{\substack{i \le s \le j \\ PI \in \mathcal{PI} \\ f_l + f_r + f_d = fr \\ f_l, f_r, f_d \ge 0}} \{1 + c_l + c_r + c_d\}$$

where

$$c_l = cost[\ell - 1, i.left, s.left - 1, f_l]$$

$$c_r = cost[\ell, s + |PI|, j, f_r]$$

$$\kappa = \#(\text{punctures of } PI)$$

$$\pi_x = x\text{th puncture of } PI$$

$$c_d = \min_{\sum_{x=1}^{\kappa} f_x = f_d} \left\{ \sum_{x=1}^{\kappa} cost[\ell - 1, \pi_x.start.left, \pi_x.end.right, f_x] \right\}$$

Recall that the CompMinCost procedure is called in the last line of the FindCost procedure. By doing so, we end up checking all possible pivots (in the form of a valid punctured interval) within the examined interval and also the additional arrangement without a pivot. We take the minimum of these costs and return it to Algorithm 4 so that it can be recorded in the dynamic programming table.

In its current, basic form, the OptimalPICover$_{c,p}$ algorithm has a running time of $O(\log_c n \cdot \binom{n}{3} \cdot c \cdot p^2 \cdot \binom{c}{p} \cdot \binom{fr}{p+2})$.

4.3.3 Performance Improvement

As the reader might have already noticed, the efficiency of the optimal algorithm is decreased mostly because of the fact that we need to calculate the cost of covering all intervals at all layers. Therefore, it would be a good idea to avoid any calculations that are unnecessary whenever we can detect them.

Algorithm 6 CompMinCost(ℓ, i, j, f)

$min \leftarrow \infty$
for s from i to j **do**
  for $PI \in \mathcal{PI}$ **do**
    $piv \leftarrow S_{\ell,s,PI}$
    $\kappa \leftarrow \#$(punctures of (piv))
    $\#rev(piv) \leftarrow \#$(revoked users under piv)
    for f_l from 0 to $f - \#rev(piv)$ **do**
      for f_d from 0 to $f - \#rev(piv) - f_l$ **do**
        $f_r \leftarrow f - \#rev(piv) - f_l - f_d$
        // Investigating cost of the arrangement (piv, f_l, f_d, f_r):
        $cost \leftarrow 1$ // Cost of the pivot
        $cost \leftarrow cost + cost[\ell - 1, i.left, s.left - 1, f_l]$
        $cost \leftarrow cost + cost[\ell, s + |PI|, j, f_r]$
        // We also need to add the cost for covering punctures:
        $mincost_d \leftarrow \infty$
        for each free rider partition $f_d^{(1)} + f_d^{(2)} + \ldots + f_d^{(\kappa)} = f_d$ **do**
          $cost_d \leftarrow 0$
          for x from 0 to κ **do**
            $\pi_x \leftarrow x$th puncture of piv
            $cost_d \leftarrow cost_d + cost[\ell - 1, \pi_x.left, \pi_x.right, f_d^{(x)}]$
          if $cost_d < mincost_d$ **then**
            $mincost_d \leftarrow cost_d$
        $cost \leftarrow cost + mincost_d$
        if $cost < min$ **then**
          $min \leftarrow cost$
if $cost[\ell - 1, i.left, j.right, f] < min$ **then**
  $min \leftarrow cost[\ell - 1, i.left, j.right, f]$
return min

Theorem 4.3.1 *Suppose that two consecutive nodes x and y at layer ℓ are fully privileged. Then, an optimal cover exists where the users under x and y are covered by a punctured interval at layer ℓ or above.*

Proof Since the nodes x and y at layer ℓ are fully privileged, so are the $2c$ nodes $x.left$ through $y.right$ at layer $\ell-1$, the $2c^2$ nodes $x.left.left$ through $y.right.right$ at layer $\ell-2$, and so on until the lowest layer. Note that covering users beneath x and y at layer $\ell-2$ or below certainly brings much more cost compared to covering them at layer $\ell-1$ because approximately c times more subsets are needed for every lower layer.

Now, suppose that there exists an optimal cover $\mathcal{C}$ that covers users beneath x and y with subsets at layer $\ell-1$. Note that since $\mathcal{C}$ is an optimal cover, the subsets of $\mathcal{C}$ that covers the users beneath x and y must be in the following format: $S_{\ell-1;i,i+c-1}$, $S_{\ell-1;i+c,i+2c-1}$ and $S_{\ell-1;i+2c,j}$ where $i < x.left$ and $y.right < j$ as shown in Figure 4.5. This is because:

- if $i = x.left$ and $\mathcal{C}$ was to include the subsets $S_{\ell-1;x.left,x.right}$ and $S_{\ell-1;y.left,y.right}$, we could replace both with only one subset $S_{\ell;x,y}$ and obtain a strictly better cover,
- and if $i < x.left$ and $\mathcal{C}$ was to include the subsets $S_{\ell-1;i,i+c-1}$, $S_{\ell-1;i+c,i+2c-1}$ and $S_{\ell-1;i+2c,y.right}$, we could replace these three subsets with only two subsets $S_{\ell;x,y}$ and $S_{\ell-1;i,x.left-1}$ and obtain a strictly better cover again.

If $\mathcal{C}$ is indeed an optimal cover and therefore includes subsets $S_{\ell-1;i,i+c-1}$, $S_{\ell-1;i+c,i+2c-1}$ and $S_{\ell-1;i+2c,j}$ where $i < x.left$ and $y.right < j$, in this case we can construct another optimal cover by replacing these three subsets with $S_{\ell-1;i,x.left-1}$, $S_{\ell;x,y}$ and $S_{\ell-1;y.right+1,j}$. Hence, if x and y are fully privileged, we can certainly obtain an optimal cover that covers users beneath x and y with subsets at layer ℓ or above and this concludes the proof.

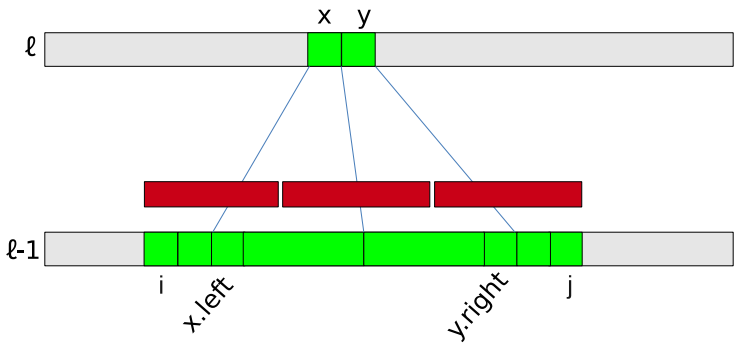


Figure 4.5: Shows cover $\mathcal{C}$ covering users beneath x and y. If $\mathcal{C}$ is optimal, another optimal cover that employs $S_{\ell;x,y}$ can be found. Therefore, we can avoid calculating the costs of the intervals that contains the interval $x.left$ through $y.right$.

Now, having proved this result, if two consecutive nodes x and y at layer ℓ is fully privileged, we can safely avoid calculating costs for intervals at layer $\ell - 1$ that start at $x.left$ or before and end at $y.right$ or after. As a result of this observation, we can reduce the running time of the optimal algorithm significantly over populations where privileged users are dense.

4.4 Heuristic Approaches

In this section, we will provide two heuristics that give near-optimal results while running significantly faster than the optimal algorithm. The top-down heuristic basically traverses the user tree in a top-down fashion and greedily takes punctured intervals conforming to the allowed free rider ratio into the cover. The hybrid heuristic offers a trade-off between the accuracy of the optimal algorithm and the speed of the top-down heuristic.

4.4.1 Top-Down: A Greedy Heuristic

In this section, we will describe a greedy heuristic which runs much faster than the optimal algorithm while reducing the transmission cost quite decently. The basic idea of the algorithm is to start from the top layer, try to cover as much as possible with the CoverLayer procedure conforming to a bound on the ratio of free riders allowed and proceed to the lower layer. The top-down heuristic is given in Algorithm 7.

Algorithm 7 TopDown$_{c,p}(\mathcal{T}, c_f)$

1: $fr \leftarrow \lfloor c_f \cdot r \rfloor$
2: $d_f \leftarrow fr/\rho$
3: $\mathcal{C} \leftarrow \{\}$
4: **for** ℓ from ℓ_{max} down to 0 **do**
5: CoverLayer(ℓ, d_f)

Algorithm 8, the CoverLayer procedure, works at a particular layer as follows: From left to right, each node is considered as a potential starting point of a punctured interval to be taken to the cover. Then, each possible punctured interval starting at this node is considered one by one. As soon as the procedure encounters a punctured interval which, if taken, will conform to the allowed free rider ratio, it takes that punctured interval into the cover. Then it continues considering punctured intervals starting from the node immediately after the last one that had been taken to the cover. When it comes to the end of the layer, it proceeds to the layer below.

We take the ratio fr/ρ where ρ is the number of all privileged users as defined before. Now we have a ratio that *can* be used in a greedy choice. Specifically, we look at the candidate punctured interval being investigated, S_{cand}, and take the ratio $\#rev(S_{cand})/\#pri(S_{cand})$. If this ratio is less than or equal to fr/ρ, this means that we can take this candidate punctured interval into the cover. If we use the same criterion with all candidate intervals, note that the resulting ratio will be less than or equal to fr/ρ.

Algorithm 8 CoverLayer(ℓ, d_f)

```
1: offset ← 0
2: f_cur ← 0
3: while offset ≤ |T[ℓ]| do
4:   for PI ∈ PI do
5:     S_cand ← S_{ℓ,offset,PI} // Rename the candidate punctured interval
6:     if #rev(S_cand)/#pri(S_cand) ≤ d_f then
7:       C ← C ∪ S_cand
8:       f_cur ← f_cur + #rev(S_cand)
9:       Mark users under S_cand as covered
10:      offset ← offset + |PI| − 1
11:      break for loop
12:  offset ← offset + 1
```

4.4.2 Introducing Tolerance for Performance Improvement

Although the top-down heuristic works quite well, there are some subtle issues with it. Note that since we work on populations where the number of revoked users are rather small, the ratio fr/ρ induced by fr/r turns out to be very small. Unfortunately, this makes the top down heuristic rather ineffective because at the higher layers, this small ratio is almost never met and everything is left to the lowest layers where the transmission cost reduction will be limited. This is against the idea of having layers in the first place.

In order to overcome this problem, we have another parameter c_{tol} denoting a tolerance factor, typically $c_{tol} \geq 1$. While making the greedy decision of having a particular punctured interval into the cover or not, now we compare the free rider ratio in that interval to $c_{tol} \cdot fr/\rho$ instead of fr/ρ alone. However, at the same time, we do not forget the total number of free riders that can be allowed, and we check whether having that interval in the cover will make the total free rider count exceed the bound. So, line 6 of Algorithm 8 will become

6 : **if** $\#rev(S_{cand})/\#pri(S_{cand}) \leq c_{tol} \cdot d_f$ **and** $\#rev(S_{cand}) + f_{cur} \leq fr$ **then**

in the top-down algorithm parametrized with c_{tol}.

Empirical evidence suggests that this modification can indeed be used to significantly increase the extent of reduction gained by the top-down heuristic. This improvement is further investigated in Section 4.5 with experimental results.

4.4.3 Hybrid Approach

In this section we will describe a mixed strategy which is a synthesis of the optimal algorithm with the top-down heuristic. Recall how we improved the performance of the optimal algorithm by first finding out the intervals which must obviously be in an optimal cover anyway and avoiding the computations for intervals that line beneath these nodes. Here, we slightly relax the selection of such intervals by requiring a certain level of fullness instead of requiring them to be full in terms of privileged users. Furthermore, we do not check individual nodes but rather punctured intervals themselves as in the top-down heuristic, instead. By doing so, the optimality of the result is compromised in return for much higher speed.

In order to realize this, we first run the top-down heuristic once excluding the lowest layer, but without adding any intervals into the cover. We only mark the free riders that should have been marked in a top-down run. Next, the dynamic programming phase is performed by treating the free riders marked by the top-down run as privileged.

The HybridCover procedure is given in Algorithm 9.

Algorithm 9 HybridCover$_{c,p}(\mathcal{T}, c_f)$

1: $fr \leftarrow \lfloor c_f \cdot r \rfloor$
2: Run TopDown$_{c,p}(\mathcal{T}, c_f)$ excluding the lowest layer
3: Update fr
4: Run OptimalPICover$_{c,p}(\mathcal{T}, c_f)$

As a result, we can say that the basic idea of the hybrid approach is to avoid

the computational cost of placing all free riders with dynamic programming by employing a top-down pass first, placing a portion of the free rider quota, and then performing the dynamic programming with the remaining free rider quota optimally, which will take much less time since most of the free riders will be placed in the top-down phase. Experimental results of this approach are given in Section 4.5.

4.5 Experiments

In this section, we will first investigate the problem of choosing the tolerance value that makes the top-down pass as effective as possible. Then we will present our experimental results and discuss the performance of our methods.

4.5.1 Choosing the Tolerance Level

In Section 4.4.2, we explained the idea of tolerance that we use to improve the effectiveness of the top-down pass. However, it was not clear how this tolerance ratio should be chosen. Obviously, if the tolerance is too small, it will have no effect. If it is too large, free riders will be spent wastefully in the upper layers, and improvement will be limited. So, we want this tolerance to be:

- large enough to allow the top-down heuristic to be as effective as possible,
- not too large to prevent the top-down pass to spend the free rider quota wastefully.

Our experimental results (See Figures 4.6 and 4.7) suggest that the top-down heuristic reaches its best performance when c_{tol} is around $1/c_f$, and for larger values of c_{tol} almost no improvement is observed. This observation can be explained as follows: Note that $c_{tol} \cdot d_f = c_{tol} \cdot fr/\rho \approx r/\rho$ gives

$c_{tol} \approx r/fr = 1/c_f$. Therefore, when $c_{tol} > 1/c_f$, this means that we allow subsets with a privileged density even less than the overall privileged density of all users. Obviously, assuming that a good cover consists of such low density subsets is a bad idea. However, by allowing c_{tol} value to be up to $1/c_f$ we only allow subsets that have a higher privileged density than the overall population. This is a decent decision because such high-density subsets are indeed likely to be in a good cover.

4.5.2 Transmission Complexity Experiments

In this section we will present our experimental results regarding both transmission cost and execution time. Both types of plots are produced by the same data randomly created for a population of 512 users. Note that such a moderate population size is preferable because in subset framework BE systems, a larger population it is typically partitioned into moderate-size populations and each partition is handled separately [3]. Several combinations of the following parameters are tested:

- $c = 16, 32$
- $p = 1, 2$
- $c_f = 0.1, 0.3, 0.5$
- $c_{tol} = 1.5, 2.0, 3.3, 10.0$

The parameters c and p were chosen according to [17] and also our population size of 512. The c_f values are selected as $0.1, 0.3$ and 0.5 because one would allow only a small portion of the revoked users to be free rider and keep majority of them as revoked. The c_{tol} values are chosen according to c_f values and the results of Section 4.5.1.

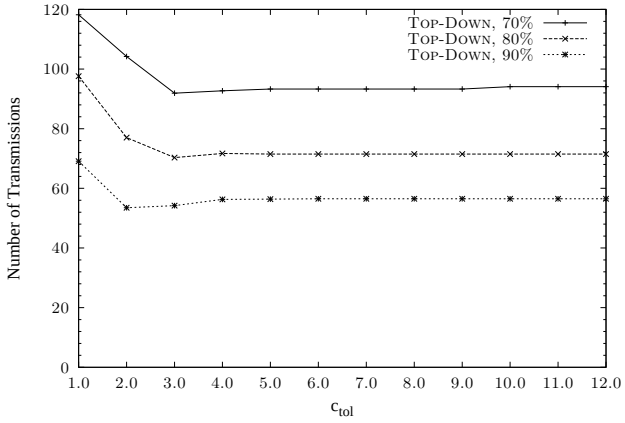
Results are given as a plot of transmission cost and execution time with respect to the ratio of privileged users in the population. Since we are interested in the mostly-privileged populations, we give the results for populations with a privileged ratio above 75%.

Figures 4.8 through 4.10 show the improvement in the transmission cost. It is observed that the cost can be reduced quite dramatically depending on c_f.

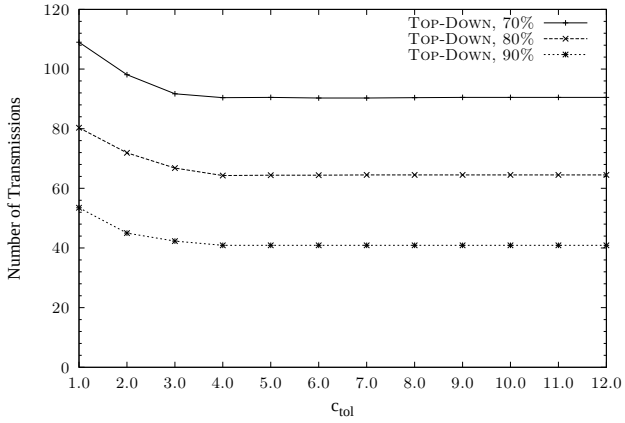
Figures 4.11 through 4.13 show the average execution times of our methods. Note that, considering both the transmission cost and execution time plots, it can be argued that the hybrid method reduces the transmission cost quite decently while running much faster than the optimal algorithm. This trade-off appears best when c_f is about 0.3.

4.6 Discussion

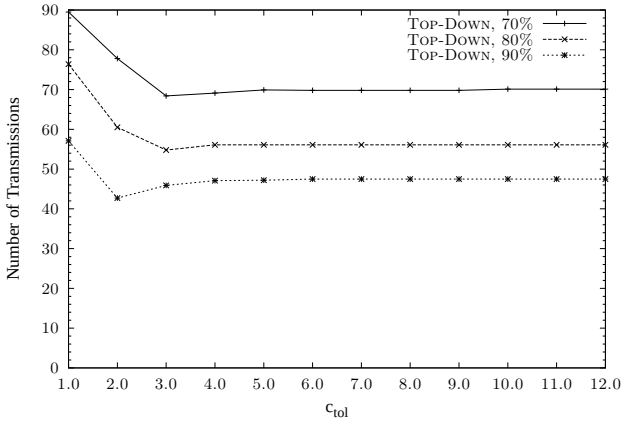
The PI scheme is currently one of the most efficient BE schemes. In this chapter, we examined the question of how the performance of the PI scheme can be further improved by allowing a limited number of free riders. We gave a dynamic programming optimization algorithm as well as a top-down heuristic which decently decreases transmission overhead while running extremely fast. We parametrized our heuristic with a tolerance parameter to use it more effectively. We also proposed a hybrid approach that first performs a top-down pass and then runs the same dynamic programming algorithm to offer a trade-off between speed and optimality. We conducted experiments to analyze the transmission costs obtained by different methods. Experiments showed that it is possible to obtain much reduced transmission complexities with the PI scheme by tolerating a limited number of free riders.



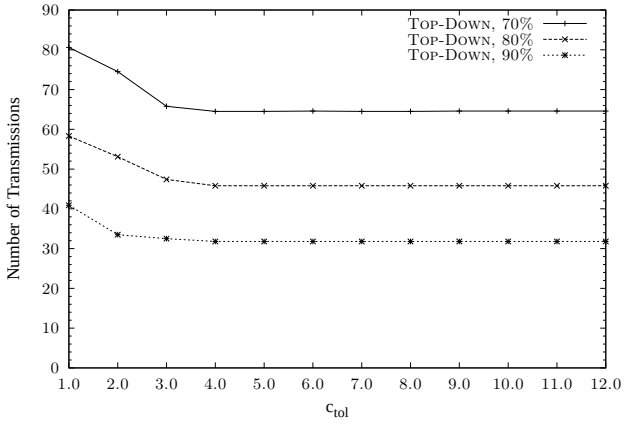
(a) $c = 16, p = 1$



(b) $c = 32, p = 1$

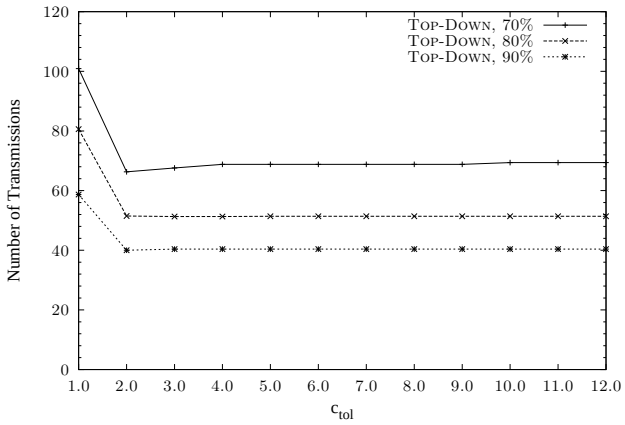


(c) $c = 16, p = 2$

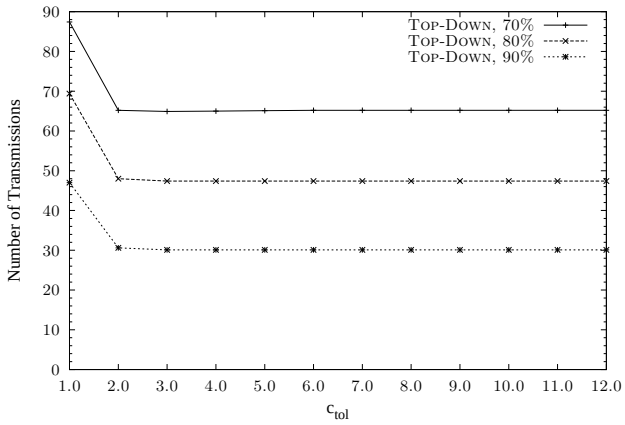


(d) $c = 32, p = 2$

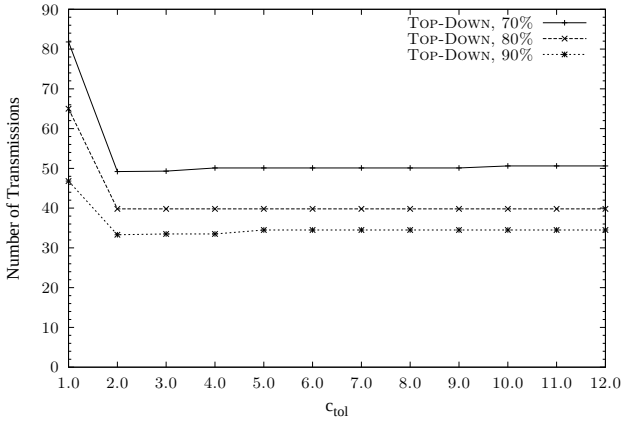
Figure 4.6: Results of the experiments for the optimal tolerance value, for $c_f = 0.3$. Note that regardless of the privileged user ratio, that are chosen as 70%, 80% and 90%, improvement stops after $c_{tol} = 1/c_f = 3.3$.



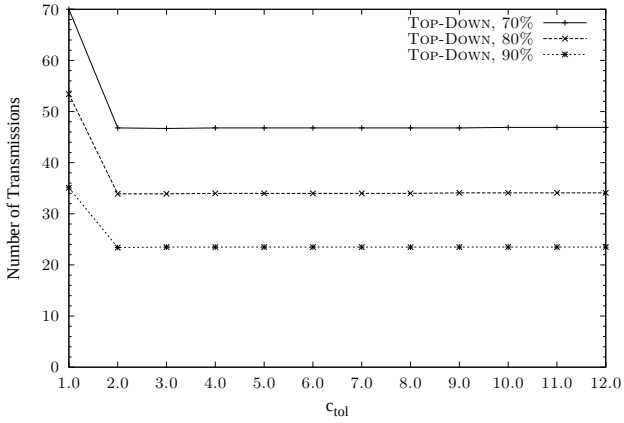
(a) $c = 16, p = 1$



(b) $c = 32, p = 1$

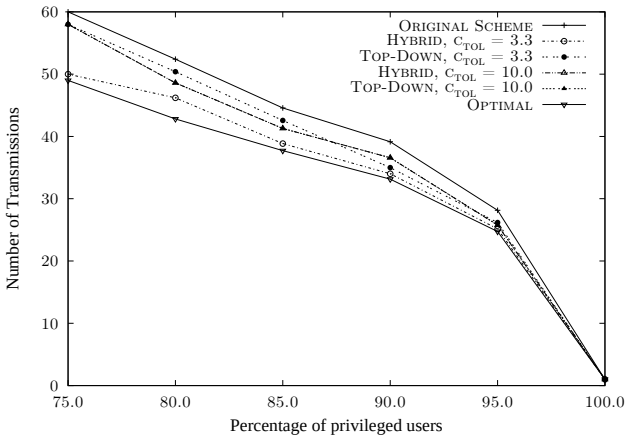


(c) $c = 16, p = 2$

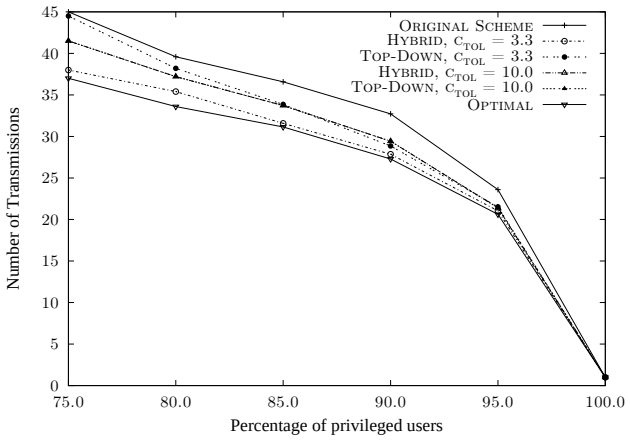


(d) $c = 32, p = 2$

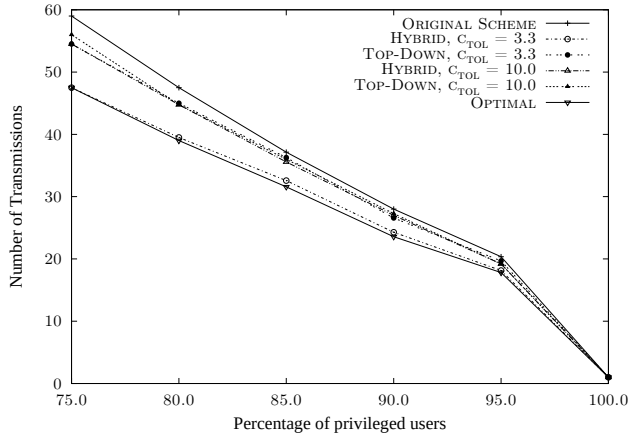
Figure 4.7: Results of the experiments for the optimal tolerance value, for $c_f = 0.5$. Note that the improvement stops after $c_{tol} = 1/c_f = 2.0$ this time.



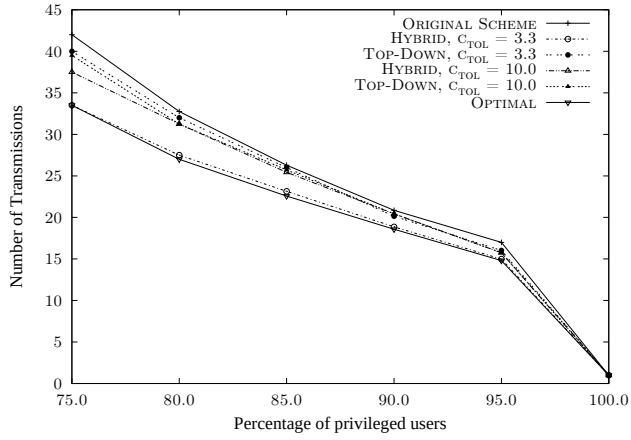
(a) $c = 16, p = 1$



(b) $c = 16, p = 2$

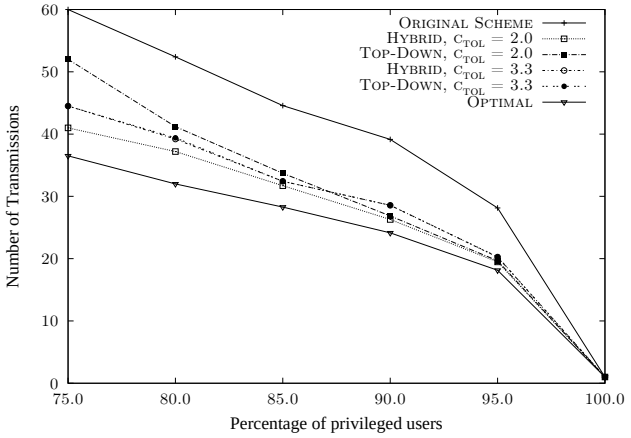


(c) $c = 32, p = 1$

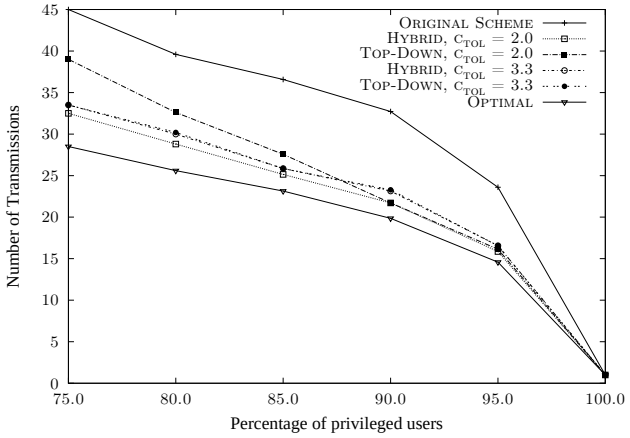


(d) $c = 32, p = 2$

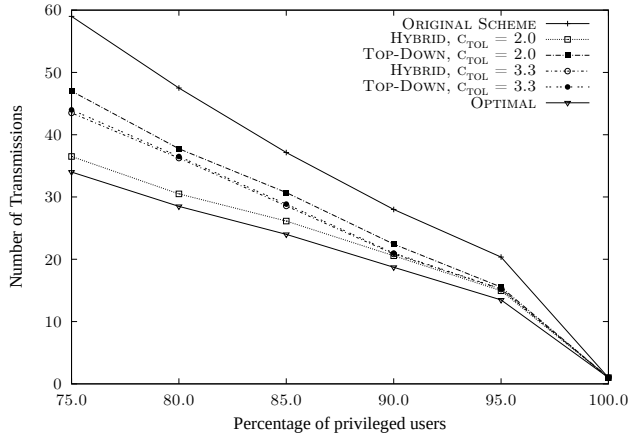
Figure 4.8: Average number of transmissions of the algorithms where $f/r = 0.1$



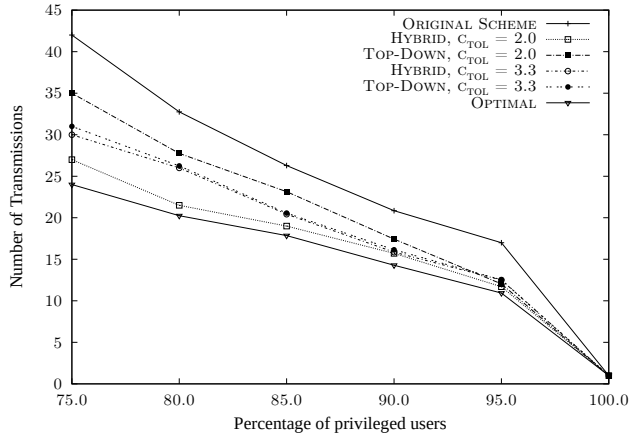
(a) $c = 16, p = 1$



(b) $c = 16, p = 2$

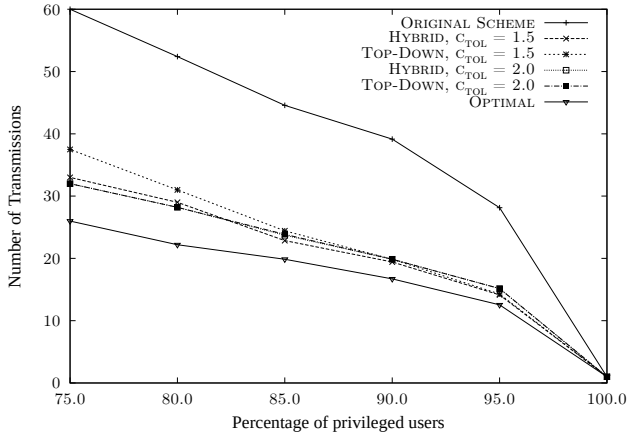


(c) $c = 32, p = 1$

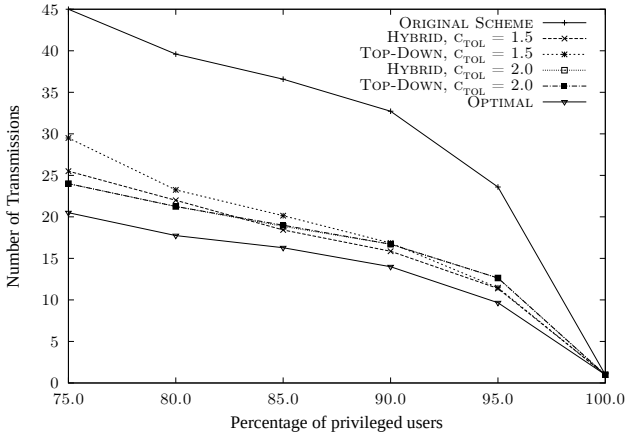


(d) $c = 32, p = 2$

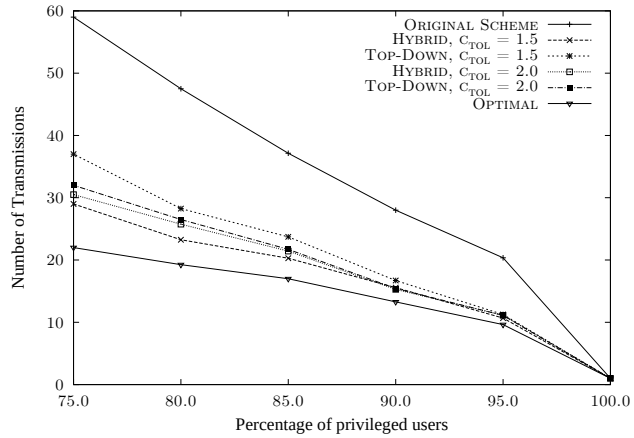
Figure 4.9: Average number of transmissions of the algorithms where $f/r = 0.3$



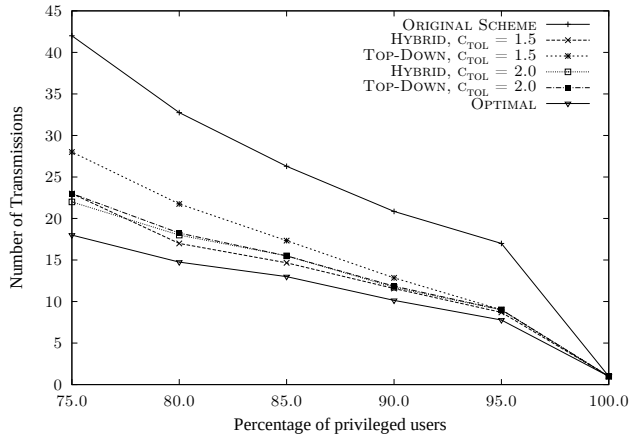
(a) $c = 16, p = 1$



(b) $c = 16, p = 2$

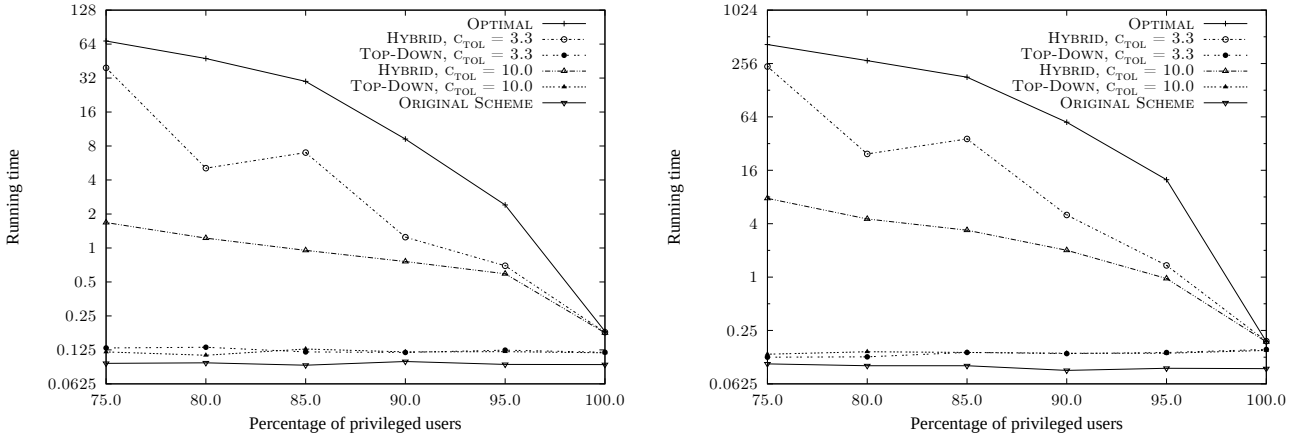


(c) $c = 32, p = 1$



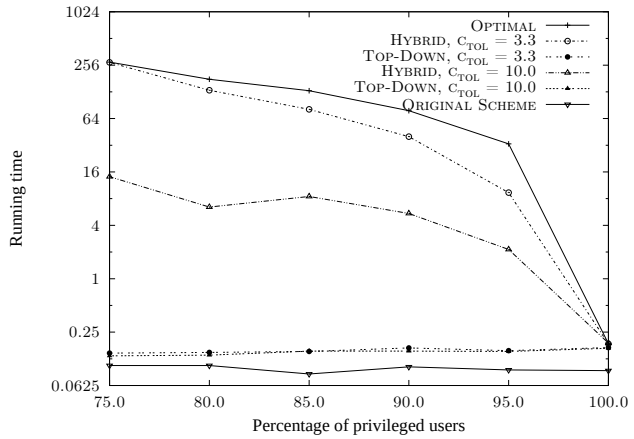
(d) $c = 32, p = 2$

Figure 4.10: Average number of transmissions of the algorithms where $f/r = 0.5$

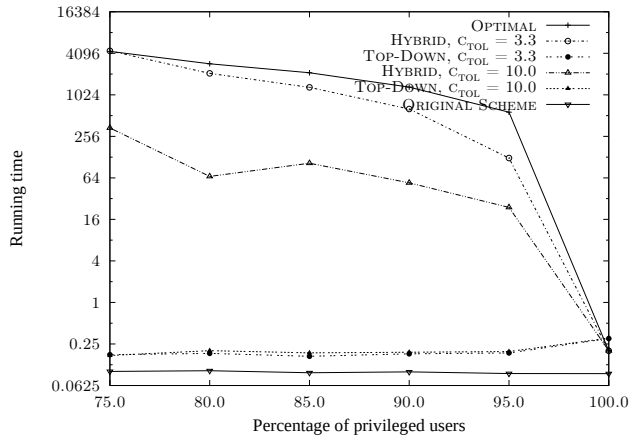


(a) $c = 16, p = 1$

(b) $c = 16, p = 2$

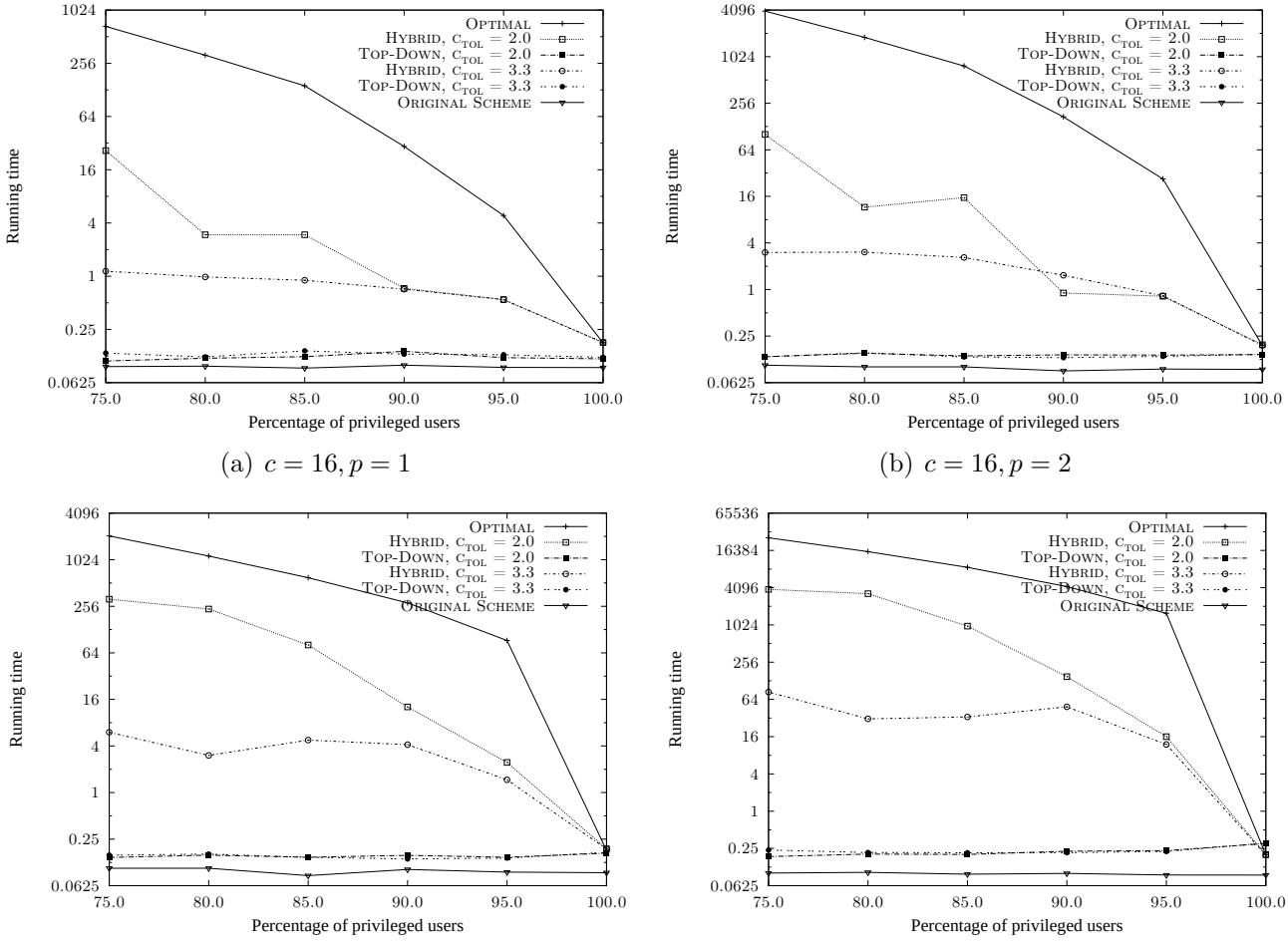


(c) $c = 32, p = 1$



(d) $c = 32, p = 2$

Figure 4.11: Average time complexity of the algorithms in terms of seconds where $f/r = 0.1$, in logarithmic scale.



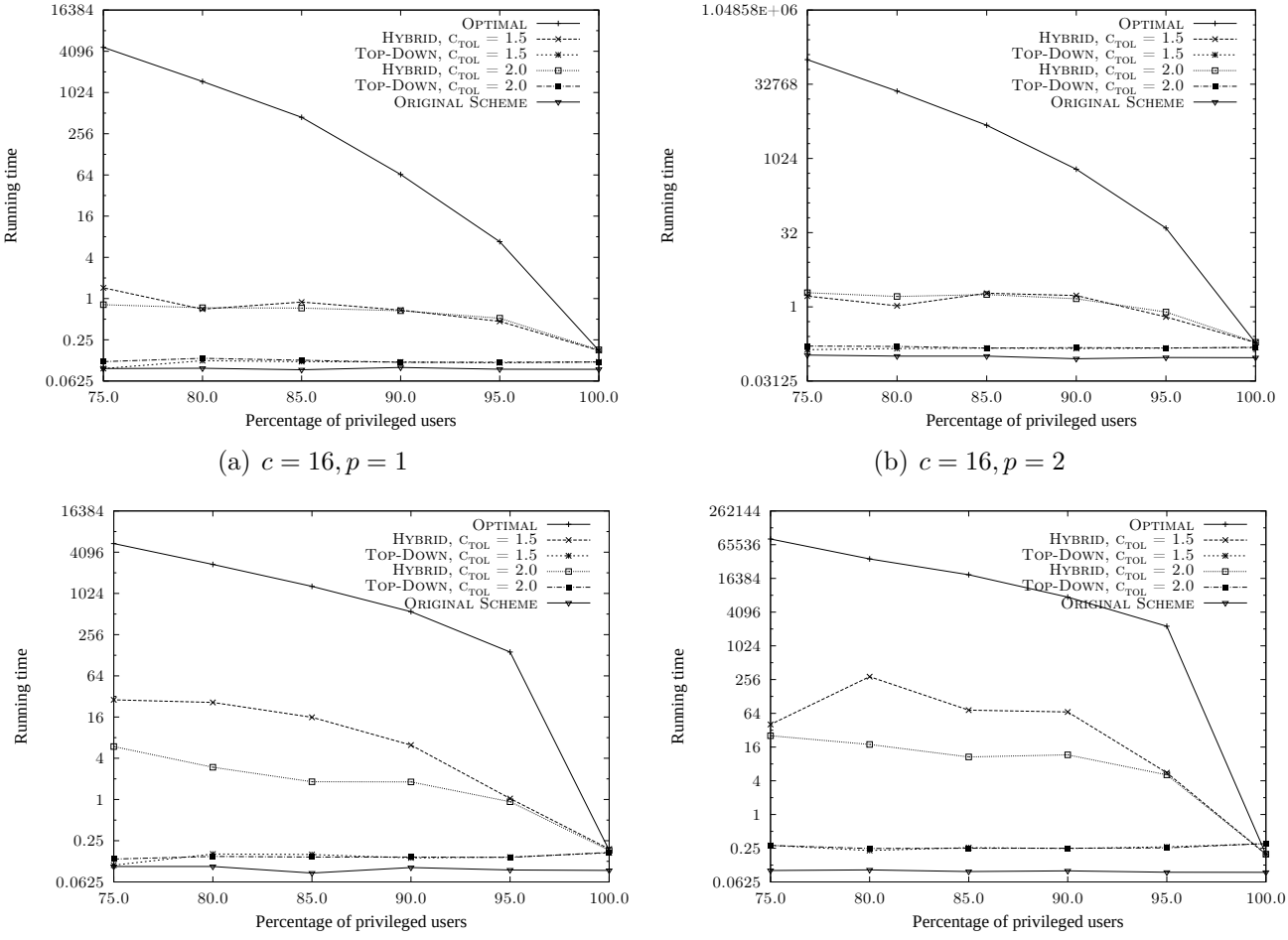
(a) $c = 16, p = 1$

(b) $c = 16, p = 2$

(c) $c = 32, p = 1$

(d) $c = 32, p = 2$

Figure 4.12: Average time complexity of the algorithms in terms of seconds where $f/r = 0.3$, in logarithmic scale.



(a) $c = 16, p = 1$

(b) $c = 16, p = 2$

(c) $c = 32, p = 1$

(d) $c = 32, p = 2$

Figure 4.13: Average time complexity of the algorithms in terms of seconds where $f/r = 0.5$, in logarithmic scale.

Chapter 5

Generic Trace and Revoke using Broadcast Encryption

Recall that in a digital content distribution setting, the content is encrypted such that the intended authorized users, having access to the decryption keys, are capable of receiving the transmission. This task is handled by broadcast encryption (BE) schemes as we mentioned in the previous chapters. However, this might not be sufficient to achieve an adequate access control.

A shortcoming of BE schemes in general is the possibility of the illegal redistribution of the content by the authorized receivers to others. This can be possible by issuing a malicious decoder that circumvents the access control used by the content distribution system. Following the standard terminology, the decoder created by an adversary is called a *pirate decoder*, the users that divulge their keys to the adversary are called *traitor*s and such keys are called *traitor key*s. The sender may want to restrict this type of behavior since such adversarial behavior introduces additional unauthorized receivers in the system. Traitor tracing is such a deterrence mechanism where an authority is capable of performing an analysis to any working pirate decoder and recovering at least one of the traitor keys that were used in its construction. Traitor tracing emerged first in the work of Chor, Fiat and Naor [32] as a solution.

Recall that we will consider black-box tracing where the tracing authority interacts with the pirate decoder in a black-box manner: querying the decoder with inputs and observing the responses of the decoder. Majority of the works, [35, 36, 37, 32, 38, 39, 40, 41], in the traitor tracing literature supports black-box tracing.

Trace and Revoke Schemes: As we have discussed above, traitor tracing and broadcast encryption has their own functionalities. However, for most of the digital content distribution systems, the ultimate goal is to combine both, so that any receiver key found to be compromised in a tracing process can be revoked in future transmissions. This is introduced by Naor and Pinkas in [42]. However, it is not possible to achieve this trivially, and a naive combination of both mechanisms would severely fail as discussed in the subsequent works [43, 44, 12]. The subset cover framework of [12] leads to a number of schemes [14, 13] which rely on combinatorial structures and support somewhat weak tracing in the symmetric setting (the tracing does not guarantee to identify a traitor but rather disables the pirate decoder). This weakness leads to a new type of attack called Pirate Evolution [45]. The studies on trace and revoke schemes followed in the public key setting [43, 46].

Tracing and Revoking Pirate Rebroadcasts: In fact, any content distribution system is vulnerable to much more serious attack of rebroadcasting: in a pirate rebroadcast the pirate instead of issuing a malicious decoder it simply publishes the content. Evidently, this defeats any mechanism that requires an interaction with the pirate decoder with some specially designed ciphertexts like the above mechanisms we discussed so far. Pirate rebroadcasting is introduced as an attack concept by Fiat and Tassa [56] and further studied in [57]. Needless to say, merely tracing pirate rebroadcasts is of little use and one should be able to revoke the involved traitor keys.

A trace and revoke scheme that is able to guard against pirate rebroadcasts is implemented as part of the AACS standard [16]. The scheme is presented and its security and performance is analyzed in Jin and Lotspiech [58] with

further analysis in [44] by Kiayias and Pehlivanoglu that revealed some limitations of that construction. In [44], tracing and revoking pirate rebroadcasting was formally modeled and a scheme for tracing and revoking an unlimited number of users was introduced. This is the only efficient trace and revoke scheme available, but restricted to the symmetric case with an underlying combinatorial key-distribution method based on subset cover framework.

Public Traceability: In Eurocrypt 2005, Chabanne, Phan and Pointcheval [59] introduced the notion of public traceability where tracing requires no secrets. A two user solution was presented in [59] and further improved to the multiuser setting with short transmissions in [38] and [60] by employing fingerprinting codes. However, the public key and the private key sizes are all linear in length of the fingerprinting code employed for key distribution. The trace and revoke scheme of [43] is also publicly traceable with shorter key sizes, i.e. $O(\sqrt{n})$ many, but requires higher bandwidth, i.e. it has a ciphertext length of $O(\sqrt{n})$.

5.1 Technical Background for Traitor Tracing

The majority of the black-box traitor tracing schemes share the same tracing strategy that is called 'hybrid coloring' in [39] or 'linear tracing' in [40] and is inherent almost in all black-box traitor tracing mechanisms. This strategy can be summarized in the following fashion: The pirate decoder is queried with a sequence of special tracing ciphertexts that are gradually randomizing the way receivers decrypt. In this sequence, while the first special ciphertext is decryptable by all receivers, the last one is decryptable by none. In between, a 'walking procedure' is processed where the i-th type of tracing ciphertext disables the first i receivers in decrypting the transmission. This is repeated many times to approximate the success rates of the decoder in decrypting each type of tracing ciphertext. Finally, the traitor key used in the construction of the pirate decoder is inferred by an analysis of the success rates.

The technique above yields a trivial traitor tracing system with each user having a unique decryption key. The ciphertext size would be very high (as much as $n/2$) in average and n in the worst case. For better trade-offs, the same technique can trivially be applied over more flexible key-distribution methods like the schemes based on fingerprinting codes [32, 39] or combinatorial structures [14, 13, 12]. In the public key setting, a number of tracing schemes (e.g. [37, 43, 46, 41]) also build their tracing strategies on 'linear tracing' technique: the pirate decoder is queried with specially crafted tracing ciphertexts that allows the walking procedure implicitly. The difficulty of designing such a scheme can be shown in the example of [41] which is broken independently by [61] and [62].

In the case of tracing and revoking, the same technique is found to be useful but the underlying multiuser encryption scheme is needed to be designed with much more demanding property. Boneh et al. [43] fulfills this by introducing an Augmented Broadcast Encryption scheme which supports revocation of any subset, i.e. the scheme can be used purely as broadcast encryption scheme, and further allows walking procedure within any enabled subset. This will eventually lead to the application of the basic linear tracing strategy and hence to the identification of a traitor. The traitor then can be easily revoked as the scheme supports any further revocation.

Fingerprinting codes [63, 32, 64] are one of the basic mathematical tools in the design of tracing mechanisms. The fingerprinting codes, in the context of tracing, have been used (in almost all of the schemes they are employed including but not limited to [35, 36, 32, 38, 58, 39, 60, 57]) to shape the key-distribution so that each receiver gets a unique set of keys.

In this long sequence of works, there are various trade-offs between a number of important efficiency characteristics of traitor tracing schemes: (i) the ciphertext size, (ii) public key size: more specifically the length of the encryption key and (iii) private key size: the key-storage required in the subscriber side. These quantities are typically expressed as a function of the number of users n, some error probability and an upper bound on the number of

corrupted users. Of particular interest, tracing scheme of [37] has constant private key size and $O(\sqrt{n})$ ciphertext size and public key size while the trace and revoke scheme of [43] has all parameters in $O(\sqrt{n})$. The application of the standard tracing strategy in a trace and revoke scheme increases dramatically the private key size from constant to quadratic in number of users.

Recently, new applications of fingerprinting codes have been introduced in [44, 40], where the code is imposed on the interaction of the tracer and the pirate decoder to observe the way the decoder responds back. This is a quite different approach compared to the conventional use of fingerprinting codes for individualizing each receiver (as in the case of virtually all earlier works we cited above) through key-distribution. This new application of fingerprinting codes leads to strong results: [44] introduces the first trace and revoke system against pirate rebroadcasts with unlimited number of traitors and revocations and [40] introduces a faster tracing strategy that can be used to replace linear tracing strategy. Following a similar approach, our goal is to transform a broadcast encryption scheme into a trace and revoke scheme.

5.2 Our Contributions

The present work has the following major contributions:

1. We present a generic transformation of a broadcast encryption scheme into a trace and revoke scheme. The transformation preserves the public and private key sizes of the underlying scheme while factors the ciphertext length with some q value that is related to the traitor coalition size the scheme will be resistant to.

As it is evident in the following Table 5.2 where we give three instantiations of our generic transformation applied to the BE schemes of [21, 65, 66] with the use of open Chor-Fiat-Naor code of [32], our results outperform the existing trace and revoke schemes of [43, 46]. In particular, we obtain the first trace

and revoke scheme with constant private key size in the standard model. The scheme of [46] can be proven in generic group model. Some weaknesses in that group model has been discussed in [67] which are similar to the ones in the random oracle model.

The schemes of [68] and [69] supports a weaker traceability (they do not guarantee to identify a traitor but rather disables the pirate decoder) similar to the subset cover framework based tracing and revoking [12]. Hence, we did not include these works in the table for comparison.

Trace-only Schemes	Public Key Size	Private Key Size	Ciphertext Size	Security & Type
BSW[37]	$O(\sqrt{n})$	$O(1)$	$O(\sqrt{n})$	Static
ADMNPS[35]	$O(n^2 \log \frac{n}{\epsilon})$	$O(1)$	$O(n^2 \log \frac{n}{\epsilon})$	Ad/ID-based
Trace&Revoke	**Public Key Size**	**Private Key Size**	**Ciphertext Size**	**Security & Type**
BW[43]	$O(\sqrt{n})$	$O(\sqrt{n})$	$O(\sqrt{n})$	Adaptive
FA[46]	$O(n)$	$O(1)$	$O(\sqrt{n})$	Ad/Generic GM
Our Results				
T&R-BGW1[21]	$O(n)$	$O(1)$	$O(1)$	Static
T&R-BGW2[21]	$O(\sqrt{n})$	$O(1)$	$O(\sqrt{n})$	Static
T&R-Del1[65]	$O(n)$	$O(1)$	$O(1)$	Static/ID-based
T&R-Del2[65]	$O(\sqrt{n})$	$O(1)$	$O(\sqrt{n})$	Static/ID-based
T&R-GW1[66]	$O(m)$	$O(1)$	$O(1)$	Static
T&R-GW2[66]	$O(n)$	$O(1)$	$O(1)$	Ad/ROM/ID-based
T&R-GW3[66]	$O(\sqrt{n})$	$O(1)$	$O(\sqrt{n})$	Ad/ID-based

Table 5.1: Our construction applied to the BE schemes of [21, 65, 66] are compared with the T&R scheme of [43] and traitor tracing-only schemes of [35] and [37]. m in GW1 is the maximum number of recipients in a single broadcast and ϵ in ADMNPS is the probability of tracing failure.

2. Of particular interest, the generic construction instantiated by [65] and [66] yields the first identity based trace and revoke scheme against both static and adaptive adversary. Recall again that the ID-based scheme of [69] supports a weaker traceability, hence we do not consider it for a comparison in here.

3. We define, for the first time, the concept of the public samplability of a

fingerprinting code which was crucial in the design of our construction. We also highlight an advantage of open fingerprinting codes over secret codes despite the fact that the secrete codes like [63, 64] are shorter. It is left open for future studies, to elaborate on the best design of a publicly samplable fingerprinting codes which can be useful in the type of transformation we have designed in this work.

4. The publicly traceable schemes of [38] and [60] suffers from long public and private keys that are typically $O(n^2)$. The trace and revoke scheme of [43] also supports public tracing but still the private key size and the ciphertext length is a function of the number of users. Our generic construction does not require any tracing secret key, hence supports fully public traceability as well as revocation. This gives us the first publicly traceable schemes with constant private key sizes while achieving short transmissions that is a function of the number of traitors only.

5. In [44], tracing and revoking pirate rebroadcasting was formally modeled and a scheme for tracing and revoking an unlimited number of users was shown. This is the only available and efficient trace and revoke scheme, but restricted to the symmetric case with an underlying combinatorial key-distribution method based on subset cover framework.

Our generic construction, by adapting the way the ciphertext is prepared, fulfills the need for tracing and revoking pirate rebroadcasts in the public key setting. The instantiations provided in the table presented above would work smoothly leading to a number of schemes against pirate rebroadcast with several different efficiency parameters and security types.

6. In this work, we only have generic construction based on a new tracing strategy and we didn't look for any advantage of a specialized underlying multiuser encryption scheme. We open a new direction of designing traitor tracing schemes: a multiuser encryption scheme that supports the tracing idea given in Section 5.5 may lead traitor tracing schemes that outperform the existing schemes. This is left open for further studies.

5.3 Preliminaries and Definitions

This section summarizes the notions of broadcast encryption and trace and revoke systems, and fingerprinting codes while explaining the notation that we will use throughout the chapter. We also give security definitions, which we will adhere to, to prove the security of our scheme. But before proceeding with the definitions, we first refer to Chernoff bounds, to be employed in our analysis later in Section 5.5.4:

Theorem 5.3.1 (Chernoff Bound) *Let $X_1, \ldots, X_n$ be independent Bernoulli trials such that $Pr[X_i] = p_i$. Let $X = \sum_n^{i=1} X_i$ and $\mu = E[X]$. Then,*

$$\forall 0 < \delta < 1, Pr[X \geq (1+\delta)\mu] \leq e^{-\mu\delta^2/3},$$

$$\forall 0 < \delta < 1, Pr[X \leq (1-\delta)\mu] \leq e^{-\mu\delta^2/2}.$$

It immediately follows from Theorem 5.3.1 that

$$\forall 0 < \delta < 1, Pr[|X - \mu| \geq \delta\mu] \leq 2e^{-\mu\delta^2/3} = 2e^{-(\mu\delta)^2/3\mu}$$

5.3.1 Broadcast Encryption in KEM structure

As we know from the previous sections, a broadcast encryption (BE) scheme is a method for encrypting messages in a way that only an intended recipient set, which we call the *privileged* users, will be able to decrypt it, and even if all other users, which we call the *revoked* users, collude, they cannot get any information about the message. The broadcast encryption schemes in the previous chapters were in the subset cover framework, and their working principle was depending on combinatorial structures defined on the user set. The rest was making several symmetric encryptions for each subset. Not all BE schemes is in this form. Also, the previous sections were not about security and their concern was rather performance. Therefore, we have not investigated the

inside of a BE scheme yet. In this section, we will see the the algorithms that form a BE scheme and security definitions that they are required to satisfy. Basically, there are two requirements for a BE scheme:

1. Correctness: Any non-revoked user must be able to decrypt the message correctly.

2. Security: Any coalition of revoked users must be unable to get any non-trivial information about the message.

In a content distribution setting with revocation, the actual data is usually encrypted with a standard, symmetric message encryption scheme while the symmetric key used in that encryption is transmitted with the BE scheme. The reason is that the data that needs to be encrypted is typically too long and encrypting the whole data directly with the BE scheme is too expensive. This hybrid approach will still be successful in disabling the revoked users from decrypting the content as the symmetric key will not be available to them. In the literature, the terminology of key encapsulation mechanism (KEM) and data encapsulation mechanism (DEM) has been used (see [70] where these notions are discussed in the context of public key cryptography). In our setting, the DEM part consists of only symmetric encryption of the transmitted data whereas the KEM part is the broadcast encryption component that encapsulates the symmetric encryption key which is used in the DEM part. Our definition for broadcast encryption will be in the KEM structure.

We will consider BE schemes in the form of three algorithms: ($\texttt{KeyDist}(1^n)$, $\texttt{Encrypt}(PK, S)$, $\texttt{Decrypt}(PK, sk_i, hdr)$). Throughout the chapter we will denote the set of all users $\{1, 2, \ldots, n\}$ by $[n]$. The functionalities of these algorithms are given below:

- $\texttt{KeyDist}(1^n)$ algorithm generates private keys for users $i \in [n]$ denoted by sk_i and a public key denoted by PK.

- `Encrypt`(PK, S) algorithm prepares a header hdr and a key K for a receiver set $S \subseteq [n]$ using public key PK. K is the symmetric message encryption key which will be used later in the DEM phase. In a broadcast, hdr is transmitted so that non-revoked users can use it to recover K.

- `Decrypt`(PK, sk_i, hdr) algorithm takes a private key sk_i and some header hdr and returns a key from the key-space of the symmetric encryption key which will be used later in the DEM phase.

As we will argue in coming sections, we require correctness and security for an encryption (hdr, K) that is output by `Encrypt`(PK, S). Informally speaking, by calling the `Decrypt`(PK, sk_i, hdr) function we should have (i) correctness: the i-th receiver having the private key sk_i must succeed to recover K if $i \in S$, and (ii) security: the output of the algorithm must reveal no nontrivial information about K if if $i \notin S$.

5.3.1.1 Correctness

A BE scheme in the KEM model satisfies correctness property if a user in the intended recipient set can decrypt and recover the symmetric key K using the header hdr and the private key sk_i. Formally stated, a BE scheme is correct if $\forall PK, \forall S \subseteq [n], \forall i \in S$, `Decrypt`$(PK, sk_i, hdr) = K$, whenever $(PK, sk_1, \ldots, sk_n) \leftarrow$ `KeyDist`(1^n) and $(hdr, K) \leftarrow$`Encrypt`(PK, S).

5.3.1.2 Security

Semantic security against chosen plaintext and ciphertext attacks for BE schemes is defined via a game between an attacker $\mathcal{A}$ and a challenger $\mathcal{C}$. However, schemes proposed so far do not agree on one particular security definition, rather each scheme has its own. This is inevitable due to the nature of BE schemes because unlike classical public-key encryption schemes, there

are many recipients, thus private keys, and there is a wider range of attacks. Recall that in public-key schemes, there is only one private key being attacked and therefore in attack models, only decryption oracles are employed to give power to the attacker. In BE schemes, however, the attacker may also be allowed to capture the private keys of some of the users and the attack is performed against another group of users disjoint from the ones private keys of which are captured. Also, there are differences depending on whether the scheme is a public-key one or identity-based. First of all, the definition of the user set differs significantly depending on this distinction. Since we will give instantiations of our generic method on both cases, we will discuss both security definitions.

In ID-based BE schemes, there is no pre-defined user set, and the attacker usually chooses the set of IDs it wishes to attack at the beginning. Also, in public-key BE schemes, the user set can be fixed at the beginning, and the attacker can choose the set of users it wishes to attack again at the very beginning, even before seeing the public key. Indeed, this is what most of the previously proposed schemes (particularly [21] and [65]) considered their security game models. However, in [66], Gentry and Waters gave more power to the attacker by allowing it to capture private keys adaptively and then choose the target set. In this model, unlike [21] and [65], the attacker captures private keys and selects target set *after* seeing the public key.

One similarity of these schemes is that they are all semantically secure. Since the attack games of [21] and [65] is almost the same, we will adopt this version of the BE attack game. In this game, the attacker first chooses the target set and gets the private keys of other users along with the public key. Then decryption query phase 1 is followed by the challenge and the decryption query phase 2 and then attacker makes its guess on the challenge.

Game 1 (Broadcast KEM-IND Game) *Both the attacker* $\mathcal{A}$ *and the challenger* $\mathcal{C}$ *are given the number of users,* n*.*

- ***Initialize.*** *$\mathcal{A}$ chooses a victim subset $S^* \subset [n]$ to attack.*
- ***Setup.*** *The challenger $\mathcal{C}$ runs* `KeyDist`(1^n) *to obtain private keys $sk_1, \ldots, sk_n$ and the public key PK. Then, $\mathcal{C}$ sends all private keys of receivers that are not in S^* (i.e., all sk_i such that $i \notin S^*$) and the public key PK to $\mathcal{A}$.*
- ***Decryption Query (phase 1).*** *(only in CCA1 and CCA2 attacks) $\mathcal{A}$ adaptively makes queries of the form (i, S, hdr) where $i \in S \subseteq [n] \backslash S^*$. Upon receiving a query, $\mathcal{C}$ responds with the result of the corresponding decryption operation* `Decrypt`(PK, sk_i, hdr).
- ***Challenge.*** *$\mathcal{C}$ runs algorithm* `Encrypt`(PK, S^*) *and obtain (hdr^*, K^*). Then it picks a random bit b and sets $K_0 = K^*$ and K_1 is randomly selected from the symmetric key space $\mathcal{K}_{\mathcal{SYM}}$. It forms the challenge string as (hdr, K_b) and sends the challenge to $\mathcal{A}$.*
- ***Decryption Query (phase 2).*** *(only in CCA2 attacks) $\mathcal{A}$ again adaptively makes queries that are formed the same way as in the Decryption Query (phase 1) with the only difference that $\mathcal{A}$ cannot use the header in the challenge string in these queries. $\mathcal{C}$ responds as in phase 1.*
- ***Guess.*** *The attacker $\mathcal{A}$ guesses b' for b and wins if $b' = b$.*

Having this definition, if we allow both decryption query phases, we obtain the KEM-IND-CCA2 game for KEM-type BE schemes. The semantic security of a BE scheme against CCA2 attacks in the KEM setting is then defined as follows:

Definition A KEM-type BE scheme B is KEM-IND-CCA2 secure if for any polynomial time attacker $\mathcal{A}$ in the KEM-IND-CCA2 game 1:

$$Adv_{\mathcal{A}} = |Pr[\mathcal{A} \text{ wins}] - 1/2| = |Pr[b' = b] - 1/2| \leq \epsilon$$

where $Adv_{\mathcal{A}}$ denotes the advantage of the attacker $\mathcal{A}$ for winning the security game and ϵ is negligible in terms of the security parameter of the system.

If we allow only decryption query phase 1, we obtain the KEM-IND-CCA1 game for KEM-type BE schemes. Therefore, we say that a KEM-type BE scheme is KEM-IND-CCA1 secure if it satisfies the same definition without the second query phase.

Similarly, if we allow none of the decryption query phases, we obtain the KEM-IND-CPA game for KEM-type BE schemes. Then, A KEM-type BE scheme is KEM-IND-CPA secure if it satisfies Definition 5.3.1.2 without the query phases.

Note that in Game 1, the adversary chooses the set it will attack at the very beginning, specifically before the keys are created by the challenger. This is called a static attack. Again, recall that Gentry and Waters [66] achieves adaptive security by allowing the attacker to choose the users to be corrupted, adaptively, after getting the public key. After retrieving enough keys, the remaining users form the target set to attack.

5.3.2 Trace and Revoke Systems

A trace and revoke (T&R) system is a system which has tracing and revoking capabilities. In this context, *tracing* is about detecting users who leak their decryption keys, and *revoking* is about invalidating the keys of such users so that the key leakage that took is neutralized. A trace and revoke system can be obtained by adding a tracing algorithm on top of a BE scheme. The tracing algorithm ensures the detection of the traitor receivers whose keys might have been used for creating the pirate decryption box being traced. Afterwards, they may simply be added to a black list so that whenever a new broadcast is to be made, the system can make sure that these receivers are in the revoked set.

In order for the tracing algorithm to identify a traitor we need to make a necessary assumption that the pirate decoder succeeds in decrypting ciphertexts intended for at least one subset with a non-negligible probability.

Otherwise, it is theoretically impossible to assert any tracing capability since it is trivial to construct such a decoder without any decryption keys. Therefore, throughout the chapter, we say a decryption box is a (σ, S)-pirate if its rate of correctly decrypting broadcasts to set S is at least σ. We denote such a pirate decoder by $\mathcal{D}_S^\sigma$. Upon encountering a decoder, we will assume that S is known to the tracer. This is a reasonable assumption and holds for almost all existing trace and revoke schemes in the literature like in [43, 68, 46]. A working pirate decoder eventually will also reveal its σ value which can be approximated by the tracer. Hence, from now on we will assume that both S and σ can be extracted from the description of $\mathcal{D}_S^\sigma$:

The formal definition of a T&R system which consists of four algorithms, is given below.

- `Setup`(1^n) is a probabilistic algorithm run by the broadcaster. It takes the number of users 1^n and generate keys $(PK, sk_1, \ldots, sk_n, TK)$. PK is the public key while sk_i is the private key of i^{th} user. TK is the tracing key, which is used for tracing and it may possibly be empty depending on the design of the system.

- `Transmit`(PK, S) is a probabilistic algorithm run by the broadcaster. It takes a subset $S \subseteq [n]$, public key PK, and outputs a header Hdr and a symmetric key K.

- `Receive`(PK, sk_i, c) is run by the user i. It takes a ciphertext c that is produced for a set S and the private key sk_i of the user and successfully decrypts and recovers K if and only if user i is in S.

- `Trace`$(S, \mathcal{D}_S^\sigma, PK, TK)$ algorithm is run by the broadcaster. It takes a set S together with a pirate decryption box for this set, $\mathcal{D}_S^\sigma$ (i.e. a (σ, S)-pirate decoder), the public key PK, and the tracing key TK, and it outputs a set A of accused traitors whose key(s) must have contributed in the construction of $\mathcal{D}_S^\sigma$.

We again call the pair $\langle S, hdr \rangle$ full header. The trace and revoke schemes of [43, 46], that we included in Table 5.2 for comparison, transmit the full header as a broadcast. Hence, the receivers will be able to access the information of S to run the decryption algorithm.

Black box tracing: In practice, it may not be possible for the tracer to reverse-engineer the pirate decoder and directly find out the keys used in the construction of the decoder, e.g. such a case is when the tracer has only remote-access to a pirate decoder or when the code of the decoder is obfuscated. The tracer can be modeled as making queries to the pirate decryption box and observing whether it decrypts or not. The tracing process with such kind of restrictions placed on the tracer is called black-box tracing as the accepted interaction between the tracer and the decoder consists of only input/output communication. In the literature, almost all of the positive results in designing traitor tracing schemes (including the schemes that we compare to our constructions) are based further on two assumptions: (i) resettability, i.e., the decoder does not maintain state during the tracing process, and (ii) availability, i.e., the pirate decoder remains available as long as the tracing process wishes to experiment with it. In this chapter, we consider black-box tracing against resettable and available pirate decoders.

5.3.2.1 Correctness

The correctness property that a T&R system must satisfy is the same as that of a BE scheme. Formally stated, $\forall S \subseteq [n]$, $\forall i \in S$, and $\forall Hdr$, it holds that $\texttt{Receive}(PK, sk_i, Hdr) = K$ provided that $(PK, sk_1, \ldots, sk_n, TK) \leftarrow \texttt{Setup}(1^n)$ and $(Hdr, K) \leftarrow \texttt{Transmit}(PK, S)$.

5.3.2.2 Security

There are two kinds of security goals in a T&R system. The first one is about the confidentiality of the message. The second one is about the tracing ability

of the system. We will define these security goals via two games, namely *message confidentiality game* for the former and *tracing game* for the latter.

The *message confidentiality game* is very similar to the Broadcast KEM-IND-CCA Game defined in the context of BE security in Section 6.1.2. However, to avoid any confusion we define it here as well, with the algorithm names defined for T&R systems.

Game 2 (Message Confidentiality Game) *Both the attacker $\mathcal{A}$ and the challenger $\mathcal{C}$ are given the number of receivers, n.*

- ***Initialize.*** *$\mathcal{A}$ chooses a victim subset $S^* \subset [n]$ to get their private keys from $\mathcal{C}$.*
- ***Setup.*** *The challenger $\mathcal{C}$ runs* $\texttt{Setup}(1^n)$ *to obtain private keys $sk_1, \ldots, sk_n$, tracing key TK, and public key PK. Then, $\mathcal{C}$ sends all private keys of receivers that are not in S^* (i.e., all sk_i such that $i \notin S^*$) and the public key PK to $\mathcal{A}$.*
- ***Decryption Query (phase 1).*** *(only in CCA1 and CCA2 attacks) $\mathcal{A}$ adaptively makes queries of the form (i, S, Hdr) where $i \in S$ and Hdr is a header. Upon receiving a query, $\mathcal{C}$ responds with the result of the corresponding decrpytion operation* $\texttt{Receive}(PK, sk_i, S, Hdr)$.
- ***Request challenge.*** *After completing its queries, $\mathcal{A}$ sends a challenge request message to the challenger $\mathcal{C}$.*
- ***Challenge.*** *$\mathcal{C}$ first runs algorithm* $\texttt{Transmit}(PK, S^*)$ *and obtains (Hdr^*, K^*). Then it picks a random bit b and generates a random symmetric key K_r. It sets $K^+ = K^*$ if $b = 1$ and $K^+ = K_r$ otherwise. Finally, it sends (Hdr^*, K^+) to $\mathcal{A}$ as the challenge.*
- ***Decryption Query (phase 2).*** *(only in CCA2 attacks) $\mathcal{A}$ adaptively makes queries of the same form as the Decryption Query (phase 1) with the difference that $\mathcal{A}$ cannot use the challenge header Hdr^* in these queries. $\mathcal{C}$ responds as in phase 1.*

- ***Guess.*** *The attacker $\mathcal{A}$ guesses b' for b and wins if $b' = b$.*

As in BE confidentiality, we define semantic security as follows:

Definition A T&R system T is IND-CCA2 secure if for any polynomial time attacker $\mathcal{A}$ allowed to make polynomially many queries both phases in the IND-CCA2 Game 2:

$$Adv_{\mathcal{A}} = |Pr[\mathcal{A} \text{ wins}] - 1/2| = |Pr[b' = b] - 1/2| \leq \epsilon$$

where $Adv_{\mathcal{A}}$ denotes the advantage of the attacker $\mathcal{A}$ for winning the security game and ϵ is negligible in terms of the security parameter of the system.

If we allow only decryption query phase 1, we obtain the IND-CCA1 game. Thus we say that a T&R system is IND-CCA1 or IND-CPA secure if it satisfies the security definition above without the second query phase or none of the query phases, respectively.

Correctness and confidentiality definitions for T&R schemes are the same as their BE counterparts. So we skip them here. There is one additional property for T&R systems, though, which is traceability. Traceability is defined via the following game between an attacker $\mathcal{A}$ and a challenger $\mathcal{C}$:

Game 3 (Tracing Game) *Both $\mathcal{A}$ and $\mathcal{C}$ are given the number of users, n, and the upper bound t on traitor coalition size.*

- ***Request.*** *$\mathcal{A}$ chooses a traitor subset T of size at most t and requests their private keys from $\mathcal{C}$.*
- ***Provide.*** *$\mathcal{C}$ runs* `Setup`(1^n) *to obtain the keys. Then, $\mathcal{C}$ sends all sk_i such that $i \in T$ and the public key PK to $\mathcal{A}$. It keeps the tracing key TK.*

- ***Forge decoder.*** *$\mathcal{A}$ chooses a set S, and creates a (σ, S)-pirate decoder box $\mathcal{D}_S^\sigma$ which, by definition, correctly decrypts the broadcasts to set S with probability at least σ. It outputs $\mathcal{D}_S^\sigma$.*

- ***Trace.*** *The challenger $\mathcal{C}$ runs* `Trace`$(S, \mathcal{D}_S^\sigma, PK, TK)$ *to obtain an accused traitor set $A \subseteq S$.*

Attacker $\mathcal{A}$ wins the game if the set A is empty or it is not a subset of T.

Definition We say that `T`= (`Setup`, `Transmit`, `Receive`, `Trace`) is a T&R scheme with tracing success probability α against t-coalition σ-pirates if no polynomial time attacker $\mathcal{A}$, forging a σ-decoder by corrupting a traitor coalition of size t, can win the game described above with probability more than $1 - \alpha$.

5.4 Fingerprinting Codes

Fingerprinting codes were originally proposed for marking sold copies of easily reproducible digital contents in order to prevent piracy. The idea is that each copy is marked in a different way, in such a way that when one or more malicious users form a coalition and reproduce a pirate copy, upon recovery of this copy, the seller would be able to identify at least one of the users in the coalition.

Fingerprinting codes are mathematically formulated as matrices where each row of the code matrix, i.e. each codeword is associated with a copy to be sold, and used to mark this copy in a unique way. This marking can be thought as changing certain bits in a digital content according to the associated codeword, but in such a way that these changes will not affect the quality of the digital content recognizable to the human eye/ear. Later, when a coalition with different copies with different marks come together, although they may find

out some of these marks, and play with them, they will not be able to form a new copy that will lead the seller to an innocent user.

In order to perform the functionalities explained above, fingerprinting codes are defined by two algorithms, `CodeGen` and `Identify`. `CodeGen`(1^n) outputs a pair $(\mathcal{C}, tk)$ where $\mathcal{C}$ is an (ℓ, n, q)-code with alphabet Q such that $|Q| = q$, and tk is a key for identifying purposes which can possibly be empty. Here, ℓ and n are the length and number of codewords (rows) of the code (matrix). `Identify`$(\mathcal{C}, tk, c)$ outputs either $\perp$ or a codeword index t which is supposed to be the index of a user in the coalition that must have forged a copy marked with codeword c.

In the context of fingerprinting codes, adversaries can be modeled as an algorithm `Forge` run by a coalition C and forges a pirate codeword. Regarding the forgery operation, the set $desc(\mathcal{C}_T) = \{x \in Q^\ell : x[i] \in \{a[i] : a \in \mathcal{C}_T\}, 1 \leq i \leq \ell\}$ is called the descendant set of $\mathcal{C}_T \subseteq \mathcal{C}$ where $x[i], a[i]$ are the i-th symbols of the related vectors. Intuitively, it is the set of codewords, letters of which must be equal to at least one of the letters of the coalition codewords at each position. For example, if two codewords in the coalition are $\{1, 0, 1, 1\}$ and $\{0, 0, 0, 1\}$ in a binary code, the descendant set of this coalition would include the following four possible pirate codewords: $\{0, 0, 0, 1\}$, $\{0, 0, 1, 1\}$, $\{1, 0, 0, 1\}$, $\{1, 0, 1, 1\}$. So, piracy inside an (ℓ, n, q)-code $\mathcal{C}$ is equivalent to producing a valid pirate codeword $p \in desc(\mathcal{C}_T)$ out of the codewords available to a traitor coalition T. Such restriction on the pirate codeword production is called '*marking assumption*' and it holds in any reasonable piracy setting (including Boneh-Shaw codes).

The performance of a fingerprinting code against a `Forge` algorithm is evaluated according to its capability of identifying traitor codewords forged by that particular `Forge` algorithm.

Definition We say that an (ℓ, n, q)-fingerprinting code (`CodeGen`, `Identify`) is an (α, w)-identifier if the following holds: Given $(tk, C) \leftarrow$ `CodeGen`(1^n),

and a `Forge` algorithm satisfying marking assumption,

$$\forall T \subseteq U \ s.t. \ |T| \leq w$$

$$Pr\left[\emptyset \subsetneq \texttt{Identify}(\mathcal{C}, tk, p) \subseteq T\right] \geq 1 - \alpha$$

where $p \leftarrow \texttt{Forge}(C)$.

When the failure probability $\alpha = 0$, we say it is a *w-identifier* fingerprinting code. If we also have $w = n$, we call it a fully collusion resistant fingerprinting code. The generated code $\mathcal{C}$ is not kept hidden from the `Forge` algorithm. This does not contradict with the marking assumption since the piracy is made possible through the marks available to the pirate. Such fingerprinting code is called open fingerprinting code. If the `Forge` algorithm is restricted to the information of $C_T = \{c_j | j \in T\}$ with $\mathcal{C} = \{c_1, \ldots, c_n\}$, then we call the fingerprinting code secret code. While the fingerprinting code of [32] is an open code, the binary fingerprinting codes of Boneh-Shaw [63] and Tardos [64] codes are secret codes.

When forging a pirate codeword, the adversary may choose some bits to replace with some symbol not equal to zero or one which will be denoted by '?'. Although the positions where the bits are deleted are known to the identifier, this significantly complicates the identification. This is quite reasonable adversarial behavior in the context of traitor tracing where the pirate may deny to decrypt some type of transmissions. In producing a pirate codeword, such pirate's anti-tracing strategy corresponds to choosing '?' marks in some of the codeword positions.

Following the model of [36]: the j-th position, for $1 \leq j \leq \ell$, in an $(\ell, n, 2)$ binary fingerprinting code is called undetectable for set T if the j-th bits of traitor codewords $\mathcal{C}_T$ all coincide; and detectable otherwise. A binary fingerprinting code is δ-robust (α, w)-identifier if the Definition 5.4 holds for any forging algorithm (again satisfies marking assumption) that can produce $p \in \{0, 1, ?\}^\ell$, but the number of positions with $p_j =?$ is not larger than $\delta \cdot \ell$.

Boneh and Naor, in [36] further presented a construction for δ-robust fingerprinting codes. Later, an improved robust fingerprinting code is introduced in [71] that has an optimal length.

5.5 Generic T&R scheme

In this section we will give a method to convert any BE scheme into a T&R scheme: the revocation is handled by the direct revocation functionality of the underlying BE scheme while the tracing is made possible through a special design of tracing ciphertexts. Informally stated, we do the following: Whenever a BE transmission is to be made to a set A, instead of encrypting directly for A, we first partition A into two subsets A_1 and A_2 and encrypt for both separately (we make the choice of the partition clear later). The revocation security of this simple variant is based on the broadcast confidentiality of the underlying BE scheme.

We design the tracing algorithm as follows: it interacts with the pirate decoder and queries some form of special ciphertexts (that we call tracing ciphertexts); based on the responses we get from the pirate decoder we infer some partial information on the traitor keys available to the pirate decoder. The tracing ciphertexts will simply look like the regular transmissions (to ensure the structural indistinguishability of the regular transmissions and the tracing ciphertexts) with the exception that a random message is encrypted to A_2 instead of the real message. If the pirate box turns out to be successful in decrypting such tracing ciphertext, we infer that a pirate key used to forge that pirate box belongs to a traitor located in set A_1. If on the other hand, it does not decrypt to the correct message then we infer that a pirate key used to forge that pirate box belongs to a traitor located in set A_2. Over a sequence of tracing transmissions for different choices of A_1 and A_2, the tracer collects information on the traitor locations. Here, the fingerprinting codes come into the scene: the tracing partition will be based on a binary fingerprinting code

so that it is possible to locate a traitor identity after tracing ciphertexts as many as the length of the code.

In this direction, the center sets up an $(\ell, n, 2)$-fingerprinting code $\mathcal{C} = \{c_1, \ldots, c_n\}$ for the tracing purposes: the code defines the sets $S_j = \{i : c_i[j] = 1\}$, for bit positions $j = 1, \ldots, \ell$. As part of the tracing process, the pirate decoder will be given a pair of BE encryptions, for each $j \in [\ell]$ intended for sets $S \cap S_j$ and $S \setminus S_j$ where the latter set of receivers are actually given a random message. If the pirate box decrypts to the actual message, we infer that the pirate key used to forge that pirate box has a 1 (resp. 0) in that position, i.e. a traitor is contained in set S_j (resp. $[n] \setminus S_j$). We construct a pirate codeword by marking 1 (resp. 0) in the j-th position of a codeword of length ℓ. When we go through all bit positions, we end up with a pirate codeword of length ℓ. That codeword will enable us to identify a traitor by calling the identifying algorithm of the fingerprinting code.

We next comment on the choice of the partition in the regular transmission. A trivial attempt would be splitting the subset A into two according to the same fingerprinting code that we use in tracing. This will ensure the structural indistinguishability of the regular transmission from the tracing transmission (preparing a random encryption for the subset A_2 is a controlled deviation from the indistinguishability as the response of the decoder will provide the tracer an information on the traitor identities). The downside of this approach is that it requires the generated fingerprinting code to be part of the public key. However, the efficient binary fingerprinting codes are mostly secret codes, i.e. the generated code should not be published: as a result we can not employ them in the public-key broadcast encryption schemes.

Our solution is to prepare the regular transmission through a a sampling algorithm that simulates the code and partitions the set of enabled receivers in such a way that is indistinguishable from the partition based on the fingerprinting code that will be used for the tracing purpose. Towards this quest, we define, for the first time, the concept of the public samplability of a fingerprinting code and we next argue, later in the section, that the two well- known

and studied fingerprinting codes (Tardos code [64] and Boneh-Shaw code [63]) are both publicly samplable. We formally define:

Definition Let $\mathtt{F} = (\ \mathtt{CodeGen}, \mathtt{Identify}\ )$ be a binary fingerprinting code. We consider a sampling algorithm $\mathtt{Z}$ that, on input n and some auxiliary information aux, samples a distribution for subset $U \subseteq [n]$.

We say $\mathtt{F}$ is *publicly samplable* by $\mathtt{Z}(1^n, aux)$ with ϵ probability of failure, if the distribution for S is statistically indistinguishable from the distribution of

$$S^* = \{i \in [n] : c_i[j] = 1\}$$

with probability at least $1 - \epsilon$ where S^* is defined over the choice of (i) an $(\ell, n, 2)$ code $\mathcal{C} = \{c_1, \ldots, c_n\}$ generated by $\mathtt{CodeGen}(1^n)$ and (ii) the column-index $j \in [\ell]$.

Below, we explain how our system works in more detail: we formally describe our construction in Section 5.5.1 provide examples of publicly samplable fingerprinting codes in Section 5.5.2. The subsequent subsections discusses the security issues of the construction.

5.5.1 Formal description of the generic construction

We suppose that we have our broadcast system in the form of a key encapsulation mechanism as defined in Section 5.3.1.2. So let $\mathtt{B}$ be a BE scheme consisting of three algorithms $\mathtt{BKeyDist}(1^n)$, $\mathtt{BEncrypt}(PK_{\mathtt{B}}, S)$, and $\mathtt{BDecrypt}(PK_{\mathtt{B}}, sk_i, hdr)$ and suppose that it is secure in the sense given in Section 5.3.1.2. Also suppose that we have a symmetric encryption scheme $\mathtt{Sym} = (\mathtt{SymEnc}_K(m), \mathtt{SymDec}_K(c))$.

In the literature, a pirate decoder with $\sigma = 1$ probability is called perfect decoder as it correctly decrypts all well-formed ciphertexts. In reality, the pirate may be content with a decoder that works only a fraction of time that is

formulated in our definition as σ-pirate with $\sigma < 1$. Such decoders are called imperfect decoders in the work of [36] and require δ-robust binary fingerprinting codes to be employed as part of the tracing process. For the simplicity of the presentation, we first construct our T&R system $\mathtt{T}$ for $\sigma = 1$ case and later discuss traceability of imperfect decoders in Section 5.5.4:

- $\mathtt{TRKeyDist}(1^n)$ The key distribution algorithm $\mathtt{TRKeyDist}(1^n)$ of $\mathtt{T}$ runs the key distribution algorithm $\mathtt{BKeyDist}(1^n)$ of $\mathtt{B}$.

  This will produce a public key $PK_{\mathtt{B}}$ and a set of private keys sk_i, $i \in [n]$, which will be distributed to the receivers. It chooses description of a binary fingerprinting code $\mathtt{F} = (\mathtt{CodeGen}, \mathtt{Identify})$ that is publicly samplable by $\mathtt{Z}$. Here, we note that the actual codewords are not generated at this moment. Hence we do not require any tracing key while the algorithm $\mathtt{Z}$ and some auxiliary information will be published as part of the public key $PK_{\mathtt{T}} =< 1^n, PK_{\mathtt{B}}, \mathtt{Z}, aux >$.

- $\mathtt{Transmit}(PK_{\mathtt{T}}, S)$ The algorithm first choses a random key K to be transmitted and a subset $U \subseteq [n]$ is sampled by the algorithm $\mathtt{Z}(1^n, aux)$. It sets $S_1 = U \cap S$ and $S_2 = S \setminus (S \cap U)$ that partitions S into two (i.e., $S = S_1 \cup S_2$ and $S_1 \cap S_2 = \emptyset$). The transmission algorithm then runs the encryption algorithm of the BE scheme for both subsets and broadcasts the message $c = (c_1||c_2)$ which is obtained as

$$(hdr_1, K_1) \leftarrow \mathtt{BEncrypt}(PK_{\mathtt{B}}, S_1), \quad (hdr_2, K_2) \leftarrow \mathtt{BEncrypt}(PK_{\mathtt{B}}, S_2)$$

$$c_1 \leftarrow hdr_1||SymEnc_{K_1}(K), \quad c_2 \leftarrow hdr_2||SymEnc_{K_2}(K)$$

- $\mathtt{Receive}(PK_{\mathtt{T}}, sk_i, c)$ The i-th user parses the public key $PK_{\mathtt{B}}$ from $PK_{\mathtt{T}}$ and extracts $hdr_1||e_1$ and $hdr_2||e_2$ from $(c = c_1||c_2)$. Then it uses the decryption function of $\mathtt{B}$ to decrypt hdr_1 and hdr_2 and retrieves

$$K_1^* = \mathtt{BDecrypt}(PK_{\mathtt{B}}, sk_i, hdr_1)$$

$$K_2^* = \mathtt{BDecrypt}(PK_{\mathtt{B}}, sk_i, hdr_2)$$

K_1^* and K_2^* will be used to decrypt to the ciphertexts e_1 and e_2 respectively. The resulting pair will be the output of the `Receive` algorithm.

- `Trace`$(S, \mathcal{D}_S^1, PK_{\mathtt{T}}, TK)$ As part of the tracing process we will query the pirate decoder $\mathcal{D}_S^1$ (in this section we consider perfect decoders with $\sigma = 1$ only, Section 5.5.4 deals with imperfect decoders) with specially designed tracing ciphertexts for a randomly chosen message K (from the key-space compatible with the KEM-DEM design) and observe the responses of the pirate decoder. The tracing ciphertexts are different from regular ciphertexts in two ways: (i) Instead of using the partition based on the sampler `Z`$(1^n, aux)$, we generate a binary code of length ℓ by running $\mathcal{F}$.`CodeGen`(1^n) and use the partition based on the j-th column(we will do this for every $j \in [\ell]$) of the generated code (ii) Rather than encrypting the message K for both subsets in the partition, we encrypt a random message to one of them.

  More specifically, the sets $S_{j,1}$ and $S_{j,2}$ for the partition are chosen based on the j-th column of the binary code generated. The sub-ciphertext for the set $S_{j,1}$ decrypts to the actual message K and the sub-ciphertext for the set $S_{j,2}$ decrypts to the random message. The tracing algorithm forms a pirate codeword whose j-th position is recorded as 1 (which implies the existence of a traitor in $S_{j,1}$) if the pirate decoder succeeds to decrypt to the actual message K, and is recorded as 0 otherwise. The $\mathcal{F}$.`Identify()` algorithm will output a user index that is responsible for the acts of the pirate decoder. The complete and detailed description of the tracing procedure is given in the following algorithm 5.5.1:

Correctness: If the user possessing the private key sk_i is in one of the sets S_1 or S_2 for which the hdr_1 and hdr_2 are constructed, then the keys output by the `Receive` algorithm contains the key K.

Algorithm 10 $\texttt{Trace}(S, \mathcal{D}_S^1, PK_{\mathtt{T}}, TK)$

Produce $\{W(j)\}_{j\in[n]} \leftarrow \mathcal{F}.\texttt{CodeGen}(1^n)$.
Initialize a pirate codeword $w \leftarrow 0^\ell$
Parse $PK_{\mathtt{B}}$ from $PK_{\mathtt{T}}$
for $j \leftarrow 1$ to ℓ **do**
    $S_j = \{i : W(i)[j] = 1\}$
    $S_{j,1} \leftarrow S \cap S_j$
    $S_{j,2} \leftarrow S \setminus S_j$
    $(hdr_1, K_1) \leftarrow \texttt{BEncrypt}(PK_{\mathtt{B}}, S_{j,1})$
    $(hdr_2, K_2) \leftarrow \texttt{BEncrypt}(PK_{\mathtt{B}}, S_{j,2})$
    Sample K and R from the message space.
    $c_1 \leftarrow hdr_1 || SymEnc_{K_1}(K)$
    $c_2 \leftarrow hdr_2 || SymEnc_{K_2}(R)$
    $c \leftarrow c_1 || c_2$
    $K' \leftarrow \mathcal{D}_S^1(c)$
    if $K' = K$ **then**
        $w_j \leftarrow 1$
    else
        $w_j \leftarrow 0$
$t \leftarrow \mathcal{F}.\texttt{Identify}(w)$
Accuse u_t for being a traitor

5.5.1.1 Traceability of the Construction

In order to prove the traceability of our generic construction, we have to prove that no polynomial time attacker A that forges a perfect decoder can win the tracing game (Game 5) with some non-negligible probability. More specifically, we will bound the winning probability of such an attacker A by a function of the security bounds of the underlying primitives.

Theorem 5.5.1 *[Traceability of a Perfect Decoder] Consider the generic T&R scheme* `T` *that is constructed as above by employing a BE scheme* `B`*, a symmetric encryption scheme* `SYM` *and an open fingerprinting code* `F`*.*

Let `B` *be KEM-IND-CCA secure with probability* ϵ_b*, and* `SYM` *be IND-CCA secure with probability* ϵ_s *and* `F` *be an* (ϵ_f, t)*-identifier* q*-ary fingerprinting code that is publicly samplable by sampler* `Z` *with failure probability* ϵ_z *in the sense of Definition 5.5.*

`T` *is a trace and revoke scheme with success probability* $1 - \epsilon_f - \ell\epsilon$ *against* t*-coalition* 1*-pirate's if it holds that*

$$4q(\epsilon_s + \epsilon_b) + 2\epsilon_z + \frac{1}{|M|} \leq 1$$

where M *is the message space.*

Proof We consider a resettable pirate decoder $\mathcal{D}_S^1$ constructed for a subset $S \in [n]$ by coalitions of at most t traitors. The tracing process can be considered as three stages: (1) Approximating the success probability of the decoder for each tracing ciphertext of type $(j, v) \in [\ell] \times [q]$, (2) Producing the pirate codeword w and finally (3) Identifying a traitor index.

(1) *Approximation:* We define $\mu_{j,v}$ as the expected number of times the decoder succeeds in experiments of type (j, v) and $\rho_{j,v}$ as the actual number of successes during the approximation process where each experiment is repeated λ times. We would like to bound the approximation difference $|\rho_{j,b} - \mu_{j,b}|$.

Choosing $\lambda = \frac{3\ln(8/\epsilon)}{\Delta^2}$, we claim that $|\rho_{j,b} - \mu_{j,b}| \geq \lambda \cdot \Delta$ with probability at most $\epsilon/4$.

Due to the allowed resettability of the decoder after each tracing query we can consider the experiments performed by the tracer are independent. By applying a two-tailed form of the Chernoff bound we will have:

$$Pr[|\rho_{j,b} - \mu_{j,b}| \geq \alpha] \leq 2e^{-\frac{\alpha^2}{3\mu_{j,b}}} \leq 2e^{-\frac{\alpha^2}{3\lambda}}$$

Substituting $\alpha = \lambda \cdot \Delta$ and $\lambda = \frac{3\ln(8/\epsilon)}{\Delta^2}$ we obtain:

$$\begin{aligned} 2e^{-\alpha^2/3\lambda} &= 2e^{-\frac{\lambda^2\Delta^2}{3\lambda}} &\leq 2e^{-\Delta^2\lambda/3} \\ &\leq 2e^{-\ln(8/\epsilon)} &\leq \epsilon/4 \end{aligned}$$

The above analysis conclude the fact that for any $j \in \{1, \ldots, \ell\}$ and $v \in \{1, \ldots, q\}$, we have $|\rho_{j,v} - \mu_{j,v}| \leq \lambda \cdot \Delta$ with probability at least $1 - \epsilon/4$. This fact is equivalent of saying $|p_{j,v} - \sigma_{j,v}| \leq \Delta$ with probability at least $1 - \epsilon/4$ for which $\mu_{j,v} = \lambda \cdot \sigma_{j,v}$ and $\rho_{j,v} = \lambda \cdot p_{j,v}$ holds.

(2) *Pirate Codeword Generation:* The tracer sets $w_j = s$ for the smallest $s \in [q]$ that satisfies $|\rho_{j,s-1} - \rho_{j,s}| \geq \lambda\theta$ and we choose θ to be equal to $\frac{1-2\epsilon_z-1/|M|}{q}$.

We next argue that the pirate codeword w is in the descendant set of the traitor coalition T. This is equivalent of claiming that if $w_j = s$ then $S_{j,s} \cap T \neq \emptyset$ holds for $j = 1, \ldots, \ell$.

By applying the triangular inequality for the equations $|\mu_{j,s-1} - \rho_{j,s-1}| \leq \lambda \cdot \Delta$ and $|\mu_{j,s} - \rho_{j,s}| \leq \lambda \cdot \Delta$, we obtain:

$$|\mu_{j,s-1} - \mu_{j,s}| \geq |\rho_{j,s-1} - \rho_{j,s}| - 2\lambda\Delta$$

with probability at least $(1 - \epsilon/4)^2 \geq 1 - \epsilon/2$.

It follows that if the tracer returns the value s, i.e., $|\rho_{j,s-1} - \rho_{j,s}| \geq \frac{\lambda(1-2\epsilon_z-1/|M|)}{q}$ for the choice of $\Delta = \frac{1-2\epsilon_z-1/|M|}{4q}$, it will happen with probability at least $1-\epsilon/2$ that

$$\begin{aligned} |\mu_{j,s-1} - \mu_{j,s}| &\geq \tfrac{\lambda(1-2\epsilon_z-1/|M|)}{q} - 2\tfrac{\lambda(1-2\epsilon_z-1/|M|)}{4q} \\ |p_{j,s-1} - p_{j,s}| &\geq \tfrac{1-2\epsilon_z-1/|M|}{2q} \end{aligned}$$

The above suggest that if a value $s \in [q]$ is returned by the tracer, it holds that the probability difference $|p_{j,s-1} - p_{j,s}|$ exceeds the threshold of $2(\epsilon_s + \epsilon_b)$ with probability at least $1-\epsilon/2$, as we know from the statement of the theorem that $4q(\epsilon_s + \epsilon_b) + 2\epsilon_z + \frac{1}{|M|} \leq 1$. In such case, we claim that $S_{j,s} \cap T \neq \emptyset$. We proceed with proof by contradiction, i.e. assume the converse of the statement $|p_{j,s-1} - p_{j,s}| > 2(\epsilon_s + \epsilon_b)$ and there exists no traitor in set $S_{j,s}$. This contradicts with the security claims of the underlying symmetric encryption scheme `SYM` and broadcast encryption scheme `B`. Indeed, if there is no traitor in set $S_{j,s}$, the pirate decoder can distinguish between the tracing ciphertext of type $(j, s-1)$ and of type (j, s) by only breaking the underlying encryption schemes. Hence, the distinguishing probability is bounded by $2(\epsilon_s + \epsilon_b)$.

On the other hand, we claim that $p_{j,0} \geq 1 - 2\epsilon_z$: this is because a tracing ciphertexts of type $(j, 0)$ for $j = 1, \ldots, \ell$ is different from a regular transmission in the way partition of the subset S is chosen. Recall that `F` is an (ϵ_f, t)-identifier fingerprinting code that is publicly samplable by sampler `Z` with failure probability ϵ_z in the sense of Definition 5.5. Hence, the pirate decoder $\mathcal{D}_S^1$ would decrypt the tracing ciphertexts of type $(j, 0)$, for all $j \in [\ell]$, with probability at least $1 - 2\epsilon_z$. Otherwise, the decoder can be used to distinguish the way sampler and the fingerprinting code works.

We also know that $p_{j,q} \leq \frac{1}{|M|}$ since a tracing ciphertext of type (j, q) totally hides the information on the message transmitted. Hence the triangular inequality implies that there exists at least one $0 < v \leq n$ such that $|p_{v-1} - p_v| \geq (1 - 2\epsilon_z - 1/|M|)/q$. With an identical argumentation as above we show that when the tracer reaches the v-th interval it will output v with

probability $1-\epsilon/2$. This suggests that the tracer will indeed output a user and not reach the end of all experiments without discovering any candidate corrupted user. Combining the above two results we conclude the pirate codeword generation phase with success probability $1-\epsilon$.

(3) *Traitor Identification* We argue above that the pirate codeword is in the descendant set of the traitor coalition. In our application of fingerprinting code, the partition in a tracing transmission does not hide the user codewords, i.e. the code is open to the adversary. Hence, `Identify(`w`)` returns a traitor index with probability at least $1-\epsilon_f$ as long as the fingerprinting code is open (not secret as in the cases of Tardos or Boneh-Shaw codes). This completes the proof of the traceability. The overall failure probability of accusing an innocent user is bounded by $\epsilon_f + \ell\epsilon$ (for the failures in identification, and in approximations, respectively) for the given parameters.

5.5.2 Samplable Fingerprinting Codes

As we argued above, the traceability of our generic construction relies on the existence of publicly samplable open fingerprinting code. Fortunately, the Chor-Fiat-Naor fingerprinting code [32] is such a code.

Theorem 5.5.2 *There exists a sampling algorithm $\mathtt{Z}_{CFN}$ with auxiliary information w (the size of the traitor coalition) such that Chor-Fiat-Naor fingerprinting code resistant against a traitor coalition of size w is publicly samplable by $\mathtt{Z}_{CFN}$ in the sense of Definition 5.5. The sampler $\mathtt{Z}_T$ requires computation time linear in number of codewords.*

Proof Due to lack of space, we omit the description of Chor-Fiat-Naor code here. Very briefly, it generates a code $\mathcal{C} = \{c_1, \ldots, c_n\} \subset Q^\ell$ randomly over an alphabet Q of size $q = 2w^2$: more specifically, for all choices of $i \in [n]$ and $j \in [\ell]$, we set $c_i[j] = k$ with probability $1/q$ for any $k \in Q$. If the length

of the code is $4w^2 \log n/\epsilon$, then the code becomes a w-traceability code with probability $1-\epsilon$, hence becomes resistant against a traitor coalition of size w.

$\mathtt{Z}(1^n, w)$ will follow the same randomized method to construct a partition $V = \{V_1, \ldots, V_q\}$: for each $i \in [n]$, randomly selects an element from the alphabet Q, say $k \in Q$ and places i in set V_k. The proof of the theorem is now straight-forward as the columns of the generated code are independently sampled and the sampler $\mathtt{Z}$ constructs the partition in exact same way. Note also, the computation time of the sampler is linear in number of codewords n.

5.5.3 Broadcast Confidentiality

We next prove the security of our construction regarding the confidentiality of the broadcast messages.

Theorem 5.5.3 (Confidentiality) *Let $\mathtt{T}$ be a trace and revoke scheme that is constructed through our generic transformation from a BE scheme $\mathtt{B}$, a symmetric encryption scheme $\mathtt{S}$ and a q-ary fingerprinting code. $\mathtt{T}$ would satisfy the KEM-IND-CCA security for any polynomial time attacker A_T such that*

$$Adv_{A_T} \leq 2q \cdot \epsilon_b + 2q \cdot \epsilon_s$$

holds where $\mathtt{B}$ is ϵ_b-secure in the KEM-IND-CCA model and $\mathtt{S}$ is ϵ_s-secure in the IND-CPA model. It further holds that if the underlying scheme $\mathtt{B}$ supports adaptive security then so does the scheme $\mathtt{T}$.

Proof We will use a game hopping approach to prove the theorem: we will start with the basic confidentiality game for trace and revoke scheme. We next modify the basic game gradually to reach a final game which provides the adversary no advantage. This is a standard proof technique that bounds the advantage of the adversary in the original game by the differences in the subsequent games.

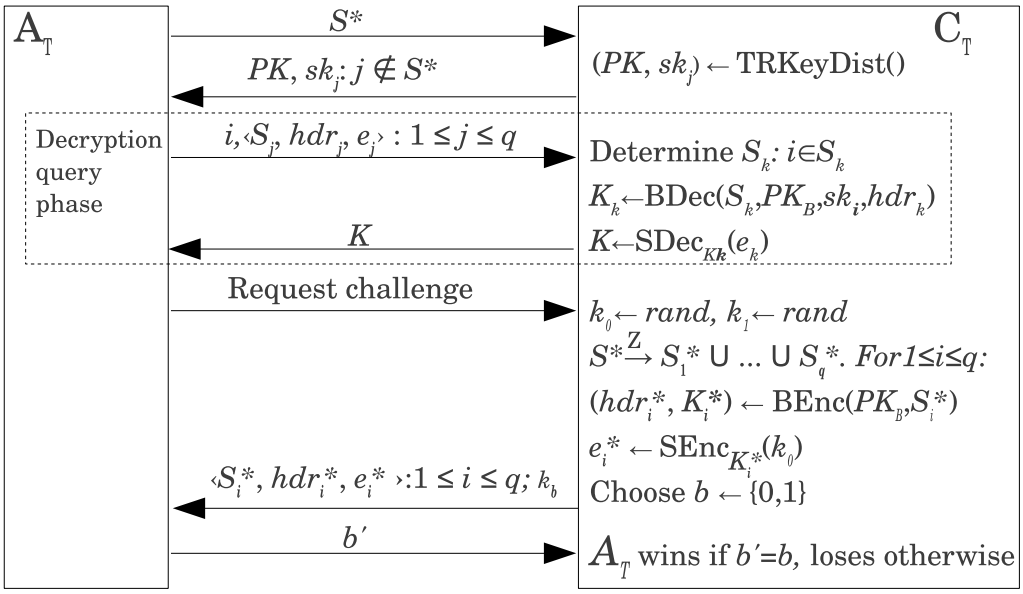


Figure 5.1: Game G_0: the actual KEM-IND-CCA game.

Let G_0 be the KEM-IND-CCA message confidentiality game for trace and revoke schemes. We had passed over the full description of the game in Section 6.1.3 with a quick reference to the Game 1. The details of this game is depicted in Figure 6.3. We denote an arbitrary adversary playing the game G_0 against the challenger C_T of the trace and revoke scheme by A_T. In the figure, we considered a static attack model where the adversary commits to the set S^* it wants to attack. The challenger publishes the public key afterwards. In contrast, it is also possible that the adversary commits to the set S^* after it observes the public key. The latter, denoted by adaptive attack, is a stronger attack model as the public key may let the adversary have some non-trivial information that is useful for the choice of the target set. The order of the commitment of the target set and the publication of the public key will not affect the validity of our proof arguments below. The choice of the order is propagated in our transformation smoothly. Hence we will consider the security for static attack model, the adaptive case follows in a similar way.

We proceed with description of the subsequent games. Let W_j denote the event that the adversary A_T wins the j-th game G_j:

Game 0: The first game, depicted in Figure 6.3 is identical to the KEM security game for trace and revoke schemes. Thus,

$$|Pr[W_0] - \frac{1}{2}| = Adv_{A_T}$$

In this game, the challenger prepares a valid ciphertext. The partition $\{S_i^*\}_{i\in[q]}$ is chosen based on the sampler $\mathtt{Z}$ and constructs the headers

$$(hdr_i^*, K_i^*) \leftarrow \mathtt{BEnc}(PK_{\mathtt{B}}, S_i^*), \quad e_i^* \leftarrow \mathtt{SEnc}_{K_i^*}(k_0)$$

for all $1 \leq i \leq q$ where k_0 and k_1 are randomly chosen keys compatible with the symmetric encryption algorithm $\mathtt{SEnc}$. Along with the full headers $\langle S_i^*, hdr_i^*, e_i^*\rangle_{i\in[q]}$, the challenger transmits k_b for a randomly selected bit b. We say the adversary wins the game if it guesses b correctly.

Game 1 through Game q: This sequence of games is identical to the first Game G_0 except the way the challenger prepares the encryption for e_i^*. In Game G_j, the encryption e_i^* for $i \leq j$ is made under a randomly chosen key K_i^+ instead of K_i^*:

$$e_i^* \leftarrow \mathtt{SEnc}_{K_i^+}(k_0)$$

Such modification breaks the relation between the header hdr_i^* and e_i^*. We next claim that there exists a broadcast encryption adversary A_B whose running time is about the same as A_T such that:

$$|Pr[W_{j-1}] - Pr[W_j]| = 2Adv_{A_B}$$

holds for $j = 1, \ldots, q$. We next argue the construction of the adversary A_B, depicted in Figure 6.4 which intends to break the KEM-IND-CCA security of the broadcast encryption $\mathtt{B}$. The adversary A_B will simulate the challenger C_T of the trace and revoke security game. The simulator will embed the challenge it receives from the broadcast challenger C_B to the challenge requested by the adversary A_T. After receiving the set S^*, the simulator will create the partition $S^* = \{S_1^*, \ldots, S_q^*\}$ immediately and forwards the j-th subset to the broadcast encryption challenger C_B. This is a crucial step to be able to

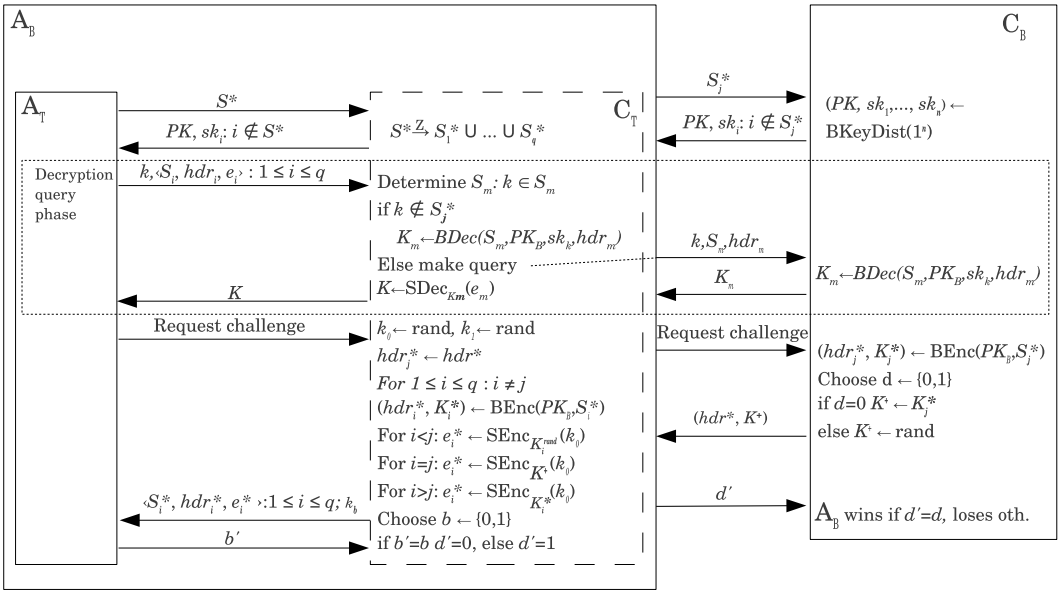


Figure 5.2: Constructing a broadcast encryption adversary A_B that simulates the challenger of the trace and revoke adversary A_T. Its advantage is reduced to the A_T's ability of distinguishing its views among games G_{j-1} and G_j.

simulate the secret keys of the scheme $\mathtt{T}$ whose keys are basically the keys of the underlying broadcast encryption scheme. The simulator will be able to respond the decryption queries $k, \langle S_i^*, hdr_i^*, e_i^* \rangle_{i \in [q]}$ of the adversary A_T as long as the secret key of the intended user k is available to the adversary A_B. Otherwise, the adversary forwards the decryption query $k, \langle S_j^*, hdr_j^* \rangle$ to the challenger C_B and retrieves the key to decrypt the symmetric encryption e_i^*.

After requesting the challenge from A_T, the adversary A_B simulates the challenger C_T as follows: All the headers $\langle S_i^*, hdr_i^*, e_i^* \rangle$ for $i \neq j$ will be prepared as in the Game G_{j-1}. The challenge received from C_B will base the $j-th$ header of the trace and revoke challenge. Upon receiving (hdr^*, K^+) from the challenger C_B we set $hdr_j^* = hdr^*$ and $e_j^* = \mathtt{SEnc}_{K^+}(k_0)$

Observe, now, that if the challenge of C_B is a valid broadcast ciphertext (this corresponds to the case $d = 0$ in Figure 6.4), the adversary A_T plays in Game G_{j-1}. In contrast, A_T plays in Game G_j if the challenge is not valid ($d = 1$ in Figure 6.4).

Let us compute the winning probability of A_B: (i) if $d = 0$ A_B wins the game if A_T wins the game, hence bounded by $Pr[W_{j-1}]$; (ii) if $d = 1$ A_B wins the game if A_T loses the game, hence bounded by $1 - Pr[W_j]$. This completes our claim that $|Pr[W_{j-1}] - Pr[W_j]| = 2Adv_{A_B}$ which is then upper-bounded by $2\epsilon_b$

At this point, we have reached to game G_q where all BE keys K_i^* are distorted. We continue with q more games gradually replacing the key k_0 with k_i^+'s.

Game q+1 through Game 2q: This sequence of games is identical to the Game G_q except the way the challenger prepares the encryption for e_i^*. In Game G_{q+j}, we set:

$$e_i^* \leftarrow \mathtt{SEnc}_{K_i^+}(k_i^+)$$

for $i \leq j$ where k_i^+'s are randomly chosen. Such modification in Game G_{q+j} hides totally the information of k_b in the first j headers. We next claim that there exists a symmetric encryption adversary A_S whose running time is about the same as A_T such that:

$$|Pr[W_{q+j-1}] - Pr[W_{q+j}]| = 2Adv_{A_S}$$

holds for $j = 1, \ldots, q$. We construct the adversary A_S in a similar way we have constructed the adversary A_B. We omit the details of the simulation due to simplicity and similarity. Hence the probability differences above are upper-bounded by $2\epsilon_s$.

Note that the last game G_{2q} gives absolutely no information about k_b thus the probability $Pr[W_{2q}]$ of the adversary winning the game G_{2q} is $\frac{1}{2}$. Applying the triangular inequalities over the probability differences above we obtain:

$$\begin{aligned} 2q \cdot \epsilon_b + 2q \cdot \epsilon_s &\geq \textstyle\sum_{i=1}^{2q} |Pr[W_i] - Pr[W_{i-1}]| \\ &\geq |Pr[W_{2q}] - Pr[W_0]| \\ &\geq |\tfrac{1}{2} - Pr[W_0]| \\ &\geq Adv_{A_T} \end{aligned}$$

which completes the security proof of our generic transformation.

Remark. In the definitions of the games, the order of attacker's choice of target set and receiving the public key can be changed. This does not affect the validity of the above argument neither. Therefore, if the primitive broadcast encryption scheme satisfies adaptive security rather than static, then so does the resulting trace and revoke scheme generated by our construction.

5.5.4 Tracing Imperfect Decoders

We proved the traceability of our generic construction against a perfect decoder in Section 5.5.1. However, as discussed in [36], any scheme whose traceability is due to a fingerprinting code can fail to identify a traitor key if the decoder chooses not to decrypt some transmissions. Such decoder is called an imperfect decoder and its behavior may lead to some gaps in producing a pirate codeword which will result to a failure in identification algorithm. The solution against such behavior is to use a δ-robust fingerprinting code to transform the tracing capability of a scheme to apply to imperfect decoders. An analysis of such a transformation is provided in [36]. We can apply the same transformation in a similar way to our case to obtain traceability against imperfect decoders.

We detail this transformation in this section to give a complete analysis: we consider an imperfect decoder $\mathcal{D}_S^{\sigma}$ that is (σ, S)-pirate where $0 < \sigma < 1$. The imperfect decoder will be queried with the same type of tracing ciphertexts of Section 5.5.1.

Let us recall the notation: we say the tracing ciphertext to be of type $(j, 0)$ if the partition is based on the j-th column of the fingerprinting code. If $S_{j,1}, S_{j,2}$ are the subsets partitioning the set S for the j-th column of the code, then the tracing ciphertext of type $(j, 0)$ will be concatenation of $c \leftarrow c_1||c_2$ where:

$$(hdr_1, K_1) \leftarrow \texttt{BEncrypt}(PK, S_{j,1}), \qquad c_1 \leftarrow hdr_1||SymEnc_{K_1}(K)$$

$$(hdr_2, K_2) \leftarrow \texttt{BEncrypt}(PK, S_{j,2}), \qquad c_2 \leftarrow hdr_2 || SymEnc_{K_2}(K)$$

The above tracing ciphertext of type $(j, 0)$ is a well-formed ciphertext with a small modification: Instead of using the partition based on the sampler $\texttt{Z}(1^n, aux)$, we generate a binary code of length ℓ by running $\mathcal{F}.\texttt{CodeGen}(1^n)$ and use the partition based on the j-th column(we will do this for every $j \in [\ell]$) of the generated code. We call the tracing ciphertext to be of type $(j, 1)$ if c_2 is constructed to encrypt a random message R in replace of K:

$$(hdr_1, K_1) \leftarrow \texttt{BEncrypt}(PK, S_{j,1}), \qquad c_1 \leftarrow hdr_1 || SymEnc_{K_1}(K)$$
$$(hdr_2, K_2) \leftarrow \texttt{BEncrypt}(PK, S_{j,2}), \qquad c_2 \leftarrow hdr_2 || SymEnc_{K_2}(R)$$

The tracing algorithm of Section 5.5.1, by only querying the ciphertexts of type $(j, 1)$ will not succeed against imperfect decoder: indeed, we can not simply infer that a traitor exists in S_{j_2} if the decoder does not decrypt the tracing ciphertext of type j. This is because the decoder is imperfect, i.e. works with σ probability, and may choose not to decrypt for such partition of (S_{j_1}, S_{j_2}). More generally speaking, the success probability of decrypting a tracing ciphertext of type $(j, 0)$ is not guaranteed to be related in a predictable manner to the overall success probability σ.

To overcome the above issues, we will, first, approximate the success probabilities of the decoder in decrypting tracing transmissions of each type. We next construct the pirate codeword based on the values of the approximated success probabilities. The complete tracing process is shown in Algorithm 5.5.4.

Let us discuss a quick overview of the tracing process of Algorithm 5.5.4. It has 3 stages: (i) Approximation, (ii) Pirate Codeword Generation and (iii) Traitor identification.

(i) As part of the approximation, we compute the success probability of the σ-pirate decoder in decrypting tracing ciphertexts of both types $(j, 0)$

Algorithm 11 $\texttt{Trace}(S, \mathcal{D}_S^\sigma, PK, TK)$ with parameters $1 > \epsilon, \Delta, \epsilon' > 0$

Produce $\{W(j)\}_{j\in[n]} \leftarrow \mathcal{F}.\texttt{CodeGen}(1^n)$.
Set $\lambda = \frac{3\ln(2/\epsilon)}{\Delta^2}$.
Initialize a pirate codeword $w \leftarrow 0^\ell$
//Approximation of the success probabilities
for $j \leftarrow 1$ to ℓ **do**
  $S_j = \{i : W(i)[j] = 1\}$
  $S_{j,1} \leftarrow S \cap S_j$
  $S_{j,2} \leftarrow S \setminus (S \cap S_j)$
  Approximate the success probability for tracing ciphertexts of type $(j, 0)$ $q_j = App(\mathcal{D}_S^\sigma, \lambda, S_{j,1}, S_{j,2}, 0)$ (Run the Algorithm 5.5.4)
  Approximate the success probability for tracing ciphertexts of type $(j, 1)$ $p_j = App(\mathcal{D}_S^\sigma, \lambda, S_{j,1}, S_{j,2}, 1)$ (Run the Algorithm 5.5.4)
//Pirate Codeword Generation
for $j \leftarrow 1$ to ℓ **do**
  if $p_j > 2\epsilon' + \Delta$ **then**
    $w_j \leftarrow 1$ // a traitor in subset $S_{j,2}$
  else
    if $q_j > 3\epsilon' + 3\Delta$ **then**
      $w_j \leftarrow 0$ // a traitor in subset $S_{j,1}$
    else
      $w_j \leftarrow ?$ // decoder does not respond
// Traitor Identification
$k \leftarrow \mathcal{F}.\texttt{Identify}(w)$
Accuse u_k for being a traitor

and $(j,1)$. Let us denote q_j and p_j respectively be the approximations. The Algorithm 5.5.4 is employed in this stage of the tracing process which basically queries the decoder with λ number of ciphertexts and returns the rate of the decryption success. Here, λ, as a function of the parameters $\epsilon, \Delta > 0$, will affect the accuracy of the approximation. More specifically, we say our approximation is Δ-close if the distance of the approximated rate from the actual success probability is less than Δ and ϵ is the probability of failure in having a Δ-close approximation.

Algorithm 12 Approximate the success probability of an imperfect decoder $App(\mathcal{D}, \lambda, S_{j,1}, S_{j,2}, b)$

$\rho \leftarrow 0$
for $k \leftarrow 1$ to λ **do**
  $(hdr_1, K_1) \leftarrow$`BEncrypt(`$PK, S_{j,1}$`)`
  $(hdr_2, K_2) \leftarrow$`BEncrypt(`$PK, S_{j,2}$`)`
  Sample K and R from the message space.
  $c_1 \leftarrow hdr_1 || SymEnc_{K_1}(K)$
  // We create a tracing ciphertext of type $(j,1)$ if $b = 1$
  if $b = 1$ **then**
    set $c_2 \leftarrow hdr_2 || SymEnc_{K_2}(R)$
    // We create a tracing ciphertext of type $(j,0)$ otherwise
  else
    set $c_2 \leftarrow hdr_2 || SymEnc_{K_2}(K)$
  $c \leftarrow c_1 || c_2$
  $K' \leftarrow \mathcal{D}(c)$
  if $K' = K$ **then**
    $\rho \leftarrow \rho + 1$
  Reset the decoder.
return ρ/λ

(ii) Pirate Codeword Generation: based on the approximated values p_j and q_j we will generate the pirate codeword. The analysis relies on the following observation: the set of receivers in set $S_{j,1} = S \cap S_j = S \cap \{i : W(i)[j] = 1\}$ will be able to retrieve the message in a transmission given in the experiment of type $(j,1)$ while the receivers in the set $S_{j,2} = S \setminus (S \cap S_j)$ (i.e. $S_{j,2} = S \cap \{i : W(i)[j] = 0\}$) will fail to decrypt. If the decoder succeeds in decrypting tracing ciphertexts of type $(j,1)$ we conclude that a traitor exists in set $S_{j,1}$.

This can be formally justified by checking if the approximated value p_j is high enough. More specifically, we check if p_j is at least $2\epsilon' + \Delta$ where ϵ' is the probability of breaking the underlying encryption schemes `BEncrypt` and `SymEnc`. In this case we set $w_j = 1$.

On the other hand, assume that there is an upper bound on the value of p_j. We, then, check how different the decoder behaves between the types of tracing ciphertexts. If the difference of $q_j - p_j$ is high enough then we conclude that the decoder distinguishes the tracing ciphertexts of type $(j, 0)$ from type $(j, 1)$; hence a traitor exists in set $S_{j,2}$. Since p_j is already assumed to be bounded by above, we check for a lower bound on q_j such that the distinguishability of the tracing ciphertext types will lead a break to the underlying encryption schemes `BEncrypt` and `SymEnc`. More specifically, if q_j is at least $3\epsilon' + 3\Delta$ then we set $w_j = 0$. In any other case, i.e. both q_j and p_j are bounded from above, the decoder does not reveal enough information. Hence we can not make a decision and set $w_j =?$.

(iii) Traitor Identification: End of the previous stage we obtain a pirate codeword $w = w_1, \ldots, w_\ell$. We finally run the identification algorithm on input w which returns an index k from the set $\{1, 2, \ldots, n\}$. The k-th receiver is accused of being a traitor. The overall failure probability is bounded by $\epsilon_f + \ell\epsilon$ (for the failure in identification and the failure in approximation respectively).

Now, we state a theorem that asserts a tracing ability for our scheme assuming primitive BE and symmetric encryption schemes satisfying certain security definitions and a fingerprint coding scheme with a certain identifying ability.

Theorem 5.5.4 *[Traceability of Imperfect Decoders] Consider the generic trace and revoke scheme* `T` *that is constructed in Section 5.5 by employing a broadcast encryption scheme* `B`*, a symmetric encryption scheme* `SYM` *and a fingerprinting code* `F`*.*

Let $\mathtt{B}$ *be KEM-IND-CCA secure with probability* ϵ_b *in the sense of Definition 5.3.1.2, and* $\mathtt{SYM}$ *be IND-CCA secure with probability* ϵ_s *in the regular sense and* $\mathtt{F}$ *be a* δ*-robust* (ϵ_f, t)*-identifier fingerprinting code that is publicly samplable by sampler* `Z()` *with* ϵ_z *probability of failure in the sense of definition 5.5.*

$\mathtt{T}$ *is a trace and revoke scheme with success probability* $1-\epsilon_f-\ell(\epsilon_s+\epsilon_b+\epsilon_z)$ *against* t*-coalition* σ*-pirates if we run the Algorithm 5.5.4 as a tracing process with parameters* $\epsilon' = \epsilon_s + \epsilon_b$*,* Δ *and* ϵ *for the choice of* $\delta = \frac{1-\sigma}{1-\gamma}$ *where* $\sigma > \gamma = 3(\epsilon_s + \epsilon_b) + 4\Delta$ *holds.*

Proof We consider a σ-pirate decoder $\mathcal{D}_S^{\sigma}$ constructed for a subset $S \in [n]$ by coalitions of at most t traitors. The tracing process of Algorithm 5.5.4 can be considered as three stages: (1) Approximating the success probability of the decoder for each type $(j, b) \in [\ell] \times \{0, 1\}$, (2) Producing the pirate codeword w and finally (3) Identifying a traitor index.

(1) *Approximation:* We define $\mu_{j,b}$ as the expected number of times the decoder succeeds in experiment of type (j, b) and $\rho_{j,b}$ as the actual number of successes during the approximation process (each experiment is repeated λ times). We would like to bound the approximation difference $|\rho_{j,b} - \mu_{j,b}|$. Choosing $\lambda = \frac{3\ln(2/\epsilon)}{\Delta^2}$, we claim that $|\rho_{j,b} - \mu_{j,b}| \geq \lambda \cdot \Delta$ with probability at most ϵ.

Due to the allowed resettability of the decoder after each tracing query we can consider the experiments performed by the tracer are independent. By applying a two-tailed form of the Chernoff bound we will have:

$$Pro[|\rho_{j,b} - \mu_{j,b}| \geq \alpha] \leq 2e^{-\frac{\alpha^2}{3\mu_{j,b}}} \leq 2e^{-\frac{\alpha^2}{3\lambda}}$$

Substituting $\alpha = \lambda \cdot \Delta$ and $\lambda = \frac{3\ln(2/\epsilon)}{\Delta^2}$ we obtain:

$$\begin{aligned} 2e^{-\alpha^2/3\lambda} &= 2e^{-\frac{\lambda^2\Delta^2}{3\lambda}} \\ &\leq 2e^{-\Delta^2\lambda/3} \\ &\leq 2e^{-\ln(2/\epsilon)} \\ &\leq \epsilon \end{aligned}$$

The above analysis conclude the fact that for any $j \in \{1, \ldots, \ell\}$ and $b \in \{0, 1\}$, we have $|\rho_{j,b} - \mu_{j,b}| \leq \lambda \cdot \Delta$ with probability at least $1 - \epsilon$. Denoting the actual success probability of the decoder in an experiment of type (j, b) by $\sigma_{j,b}$ we conclude that $|p_j - \sigma_{j,1}| \leq \Delta$ and $|q_j - \sigma_{j,0}| \leq \Delta$ hold with probability at least $1 - \epsilon$.

(2) *Pirate Codeword Generation:* We next argue whether the codeword w is in the descendant set of the traitor coalition, i.e. the constructed pirate codeword, indeed, is formed by the actual coalition of traitors.

Our argument is based on the following observation: the set of receivers in set $S_{j,1} = S \cap S_j = S \cap \{i : CW(i)[j] = 1\}$ will be able to retrieve the message in a transmission given in the experiment of type $(j, 1)$ while the receivers in the set $S_{j,2} = S \setminus (S \cap S_j)$ (i.e. $S_{j,2} = S \cap \{i : CW(i)[j] = 0\}$) will fail to decrypt. If the approximated success probability p_j is high enough (will be made clearer below), then we claim the existence of a traitor in set $S_{j,1}$. Otherwise, we check the difference of $q_j - p_j$, i.e. checking a lower bound for q_j as p_j is already bounded by above: if the difference is substantially high (again will be made clearer below) then we conclude that there is a traitor in set $S_{j,2}$ that can differentiate the experiment performed. In any other case, we can not make a choice, hence we put ?.

We next make an analysis of our argument summarized above:

First Case (i): If $p_j > 2(\epsilon_s + \epsilon_b) + \Delta$, we claim that there exists a traitor in the set $S_{j,1}$. Considering the accuracy of approximation we obtain $\sigma_{j,1} > 2(\epsilon_s + \epsilon_b)$. We, then, proceed with proof by contradiction: Assume that the traitor coalition set T satisfies $(T \cap S) \subset S_{j,2}$. Hence, the ciphertexts transmitted in

an experiment of type $(j, 1)$ are hiding the plaintext that is needed to return in order for the experiment to succeed with a failure probability of $2(\epsilon_s + \epsilon_b)$ due to the semantic security of the underlying symmetric encryption scheme $(\mathtt{E}, \mathtt{D})$ and broadcast encryption scheme $\mathtt{BE}$. It concludes that the pirate decoder can win the experiment of type $(j, 1)$ with probability at most $2(\epsilon_s + \epsilon_b)$, i.e. $p_j \leq 2(\epsilon_s + \epsilon_b) + \Delta$ which contradicts with the current-condition on p_j. We set $w_j = 1$ in this case.

Second Case (ii): If $p_j < 2(\epsilon_s + \epsilon_b) + \Delta$ and $q_j > 3(\epsilon_s + \epsilon_b) + 3\Delta$, then we claim that there exists a traitor in the set $S_{j,2}$. We again proceed with proof by contradiction: assume that the traitor coalition set T satisfies $(T \cap S) \subset S_{j,1}$. Provided that all traitors are included in set $S_{j,1}$, the pirate decoder can distinguish between the distributions $\mathtt{Transmit}(PK, S, m)$ and a transmission given in experiment of type (j) only with probability at most $2(\epsilon_s + \epsilon_b)$ due to the semantic security of the underlying symmetric encryption scheme $(\mathtt{E}, \mathtt{D})$ and broadcast encryption scheme $\mathtt{BE}$. Hence we obtain $|\sigma_{j,0} - \sigma_{j,1}| \leq 2(\epsilon_s + \epsilon_b)$. Combining with the accuracy of the approximation of $\sigma_{j,0}$ and $\sigma_{j,1}$, we obtain $|q_j - p_j| \leq 2(\epsilon_s + \epsilon_b) + 2\Delta$. Apparently, this contradicts with the current conditions on q_j and p_j. We set $w_j = 0$ in this case.

Third Case (iii): If $p_j < 2(\epsilon_s + \epsilon_b) + \Delta$ and $q_j < 3(\epsilon_s + \epsilon_b) + 3\Delta$, this corresponds to the case that the pirate decoder denies responding, hence we can not get enough information. We set $w_j =?$.

(3) *Traitor Identification* The pirate codeword constructed above contains a fraction of ?'s that is chosen for columns with $q_j < 3(\epsilon_s + \epsilon_b) + 3\Delta$. Due to the approximation, with probability $1 - \epsilon$, it holds that $\sigma_{j,0} < 3(\epsilon_s + \epsilon_b) + 4\Delta$. For simplicity of the notation let us say $3(\epsilon_s + \epsilon_b) + 4\Delta = \gamma$ Now, if δ is the fraction of columns where $\sigma_{j,0} < \gamma$ then the decoder's error rate (that is $1 - \sigma$) is at least $\delta(1 - \gamma)$. Solving for δ we obtain $\delta < \frac{1-\sigma}{1-\gamma}$ which is satisfied due to the given bounds for σ. (given that $\sigma > \gamma$ as part of the theorem statement, we also ensure that $\delta < 1$ holds.)

As the pirate codeword is in the descendant set of the traitor coalition

with a fraction of at most δ ?'s, the `Identify`(w) returns a traitor index with probability at least $1 - \epsilon_f$.

This completes the proof of the traceability. The overall failure probability of accusing an innocent user is bounded by $\epsilon_f + \ell\epsilon$ (for the failures in identification, and in approximation, respectively) for the given parameters.

5.6 Comparison with Existing T&R Schemes

In this section, we will discuss the performance of the resulting T&R schemes that emerges when we use some of the popular BE schemes in the literature with our construction technique.

5.6.1 Our Instantiations

We instantiate our generic construction with the open fingerprinting code of Chor-Fiar-Naor in [32] and the following three broadcast encryption schemes. The efficiency characteristics of the below instantiations are compared to the existing trace and revoke schemes in the introduction(see Table 5.2)

`BGW1:` One seminal work on public key BE is the scheme of Boneh, Gentry, and Waters [21]. In the basic scheme given in this chapter, which we will denote with `BGW1`, the public key consists of $O(n)$ group elements whereas private keys and ciphertexts are of size $O(1)$. The encryption algorithm of this scheme is dominated by $O(p)$ group operations needed, where p is the size of the privileged user set. However, with a caching technique, this complexity can be reduced to $O(\min(p, r))$ where r is the size of the revoked user set. The decryption process consists of $O(1)$ pairing calculations, and $O(p)$ group operations which again can be reduced to $O(\min(p, r))$.

When we instantiate our generic T&R construction with this scheme, public and private key sizes are unchanged (i.e., $O(n)$ and $O(1)$, respectively) and the ciphertext size is only doubled and hence is $O(1)$. Encryption and decryption complexities become $O(p)$, instead of $O(\min(p, r))$ since we partition the privileged set and the caching technique will no longer be effective.

`BGW2:` The general scheme of [21], which we will denote with `BGW2`, has the idea to use several instances of the basic scheme in parallel. Since all instances can use the same set of public keys, with an optimal trade-off, the public key size is reduced to $O(\sqrt{n})$ with the cost of increasing ciphertext size to $O(\sqrt{n})$ group elements. Users still have private keys of size $O(1)$.

As far as the encryption and decryption algorithms are concerned, computation cost is reduced to $O(\sqrt{n} + \min(p, r))$ since for each of the $\sqrt{n}$ parts, another miniature basic scheme instance is run each having r' revoked users. $O(\sqrt{n})$ comes from the fact that calculations must be done for each instance, and $O(\min(p, r))$ comes from the total number of calculations in all these instances using the caching idea. For details we refer the reader to [21].

When we instantiate our generic T&R construction with `BGW2`, public and private key sizes are unchanged. Ciphertext size is only doubled, therefore it remains $O(\sqrt{n})$.

In the complexities of the encryption and decryption algorithms, a similar performance drop takes place as in `BGW1` again because of the fact that caching technique becomes ineffective due to partitioning. So, encryption has a complexity of $O(\sqrt{n}+p)$. In the decryption process, receivers make calculations only within their instance, therefore the complexity becomes $O(\sqrt{n}+p')$ where p' is the number of privileged users within the instance of the decrypting receiver. Since $p' \leq \sqrt{n}$, we can say that the complexity of decryption is $O(\sqrt{n})$.

`Del:` For the ID-based setting, we consider the BE schemes of Delerablée [65] and the virtually identical scheme of Sakai and Furukawa [72]. This scheme

is dynamic in the sense that the total number of users is not fixed in the setup and any new user can decrypt all previously distributed messages. Another property of it is that it puts a bound m for the number of receivers per transmission, and the public key size is linear in this number instead of the number of all users as in public key BE schemes we mentioned above.

In this scheme, the public key is of size $O(m)$ where m is the maximum number of receivers per transmissions. Private keys of users consist of single group elements hence private keys are of size $O(1)$. Ciphertexts consist of two group elements which means it is also $O(1)$. One may assume that $m = n$ or $m = \sqrt{n}$, where n is considered to be the initial number of users in the system. We call the special case $m = n$ as `Del1`. In case $m < n$, several encryptions are done $\lceil n/m \rceil$ times. i.e., ciphertext becomes $O(\sqrt{n})$ when $m = \sqrt{n}$. We call this case as `Del2`.

Using our construction with the scheme of [65] leads to the following result. Keys do not change, therefore key sizes remain the same: $O(m) = O(\sqrt{n})$ public key and $O(1)$ private keys per users. Ciphertext is only doubled, hence it remains $O(1)$ since it is four group elements.

`GW1`: In [66] three different schemes with different properties are given. One is a standard BE scheme (not ID-based) which we will call `GW1`. In this scheme, public key is of size $O(m)$ where m is the maximum number of receivers in a broadcast. Ciphertext and private keys are of constant size. This scheme satisfies semi-static security, where the attacker commits to a subset of users before seeing public keys first, and afterwards can choose any subset of it as the final set to be attacked. Semi-static security is a sort of middle-ground between static and adaptive security, and hence shown as supporting static security in table 5.2. When our T&R construction is applied to `GW1`, only the ciphertext size and computation complexities will double. Therefore we get a T&R scheme with the same asymptotic performance as `GW1`.

`GW2` and `GW3`: Second and third schemes we will consider from [66] are identity based BE schemes. `GW2` is an adaptively secure identity based BE

scheme in the random oracle model. In this scheme, constant size ciphertexts are achieved by using a hash function. `GW3` does not use any hash functions but instead uses the idea of running parallel smaller instances of the basic scheme given in [66] and achieve $O(\sqrt{n})$ size ciphertexts as in `Del2` above. Therefore, public key size is $O(n)$ and ciphertext is $O(1)$ in `GW2`, whereas in `GW3` both are $O(\sqrt{n})$. Private keys are $O(1)$ for both schemes.

Instantiating `GW1` and `GW2` with our construction leads to T&R systems with inherited asymptotic ciphertext, public key and private key sizes, since only ciphertexts as well as computations are doubled.

5.6.1.1 Other Schemes

We listed above only a small number of the broadcast encryption schemes that are either well-known or are suitable for instantiations with reasonable efficiency performance. It would be worth to note that a broadcast encryption scheme with a ciphertext size of $O(r)$ will result a ciphertext length of $O(n)$ in our generic trace and revoke scheme. A number of schemes in the literature (a non-exhaustive list would include [12, 73, 74] in both symmetric and asymmetric settings) are of this type that are preferably in revocation-only settings due to other parameters. Even though we do not illustrate instantiations based on such schemes in Table 5.2, we note that they may serve good in certain application domains for small n values. For instance, [73] provides a scheme with the public and private keys as a constant number of group elements from an elliptic-curve group of prime order; this can be used in our generic construction to keep any kind of key storage (both private and public keys especially for off-line decryption process) low in small devices like sensor nodes.

5.6.1.2 Remarks

The round complexity of a black-box tracing mechanism is the number of queries asked to the decoder and it is an important efficiency parameter as formalized in [40]. Most of the schemes in the literature (e.g. [37, 43]) leads to a quadratic number of tracing queries in number of users regardless of the traitor coalition. On the other hand, our generic construction, when employs a fingerprinting code with a bound w on traitor coalition, would result in round complexity of $O(w^4)$. Here in this paper, we applied the linear tracing strategy to locate a subset containing a traitor. As an alternative suggested by [40], we may have used the tracing strategy of [40] which would reduce the round complexity to $O(w^2)$.

A further improvement over the scheme is possible if the fingerprinting code is generated at the time of tracing for $|S|$ many codewords instead of as many as the number of receivers n. This is a substantial improvement over the encryption time as it is sufficient to flip $|S|$ coins. This improvement should be considered for applications where the enabled set of receivers is substantially less than the whole population.

5.6.2 Comparison

A number of trace and revoke schemes in the non-black box setting that we do not include in comparison.

`BW:` A popular public key T&R scheme is given in [43], which we call `BW`, where the public key, private key, and ciphertext sizes are $O(\sqrt{n})$. Encryption complexity is $O(\sqrt{n} + \min(p, r))$ and the decryption complexity is $O(\sqrt{n} + \min(p', r'))$, where r' is the number of revoked users within the instance of the decrypting receiver. Since $\min(p', r') \leq \sqrt{n}$, we can say that the complexity of decryption is $O(\sqrt{n})$. For details we refer the reader to [43].

A comparison of the scheme generated by our construction using `BGW1` and

BGW2 to the T&R scheme of [43] is provided in Table 5.2.

FA: In comparison to [43] a trade-off between the private-key length and the public key length is achieved in a trace and scheme due to Furukawa and Attrapadung [46]: this scheme has a constant private key size in exchange for a public key of linear size in the number n of receivers. However, it is noted in [46] that only $O(\log \sqrt{n})$ of the public key is needed by a receiver along with its unique private key to decrypt a broadcast. Hence, in a setting where the public key is also stored in decryption device, there seems to be no observed problem of having $O(n)$ public key size in total. There is still a disadvantage of FA related to its security proof: it is proved in the generic bilinear group model under the subgroup decision assumption. Some weaknesses in the generic group model has been discussed in [67] which are comparable with the ones in the random oracle model. Hence, we can easily argue that the constant private key size of a trace and revoke scheme in the standard model does not exist till our results.

ADMNPS: We compare our results in the ID-based setting with a recent trace-only scheme (to the best of our knowledge we are not aware of any id-based trace and revoke scheme) by Abdalla et al. [35], which we will call ADMNPS. This scheme achieves adaptive security in the standard model with a constant private key size that is as small as a four group elements. As we have already seen in the case of FA, this gain comes with its price: the scheme of ADMNPS, based on a fingerprinting code, has its public key and ciphertext of size in proportion to the length of the code. Based on fully-collusion resistant fingerprinting code, the typical length would be of $O(n^2 \log \frac{n}{\epsilon})$ (ϵ is the failure probability of tracing) which makes the scheme quite inefficient due to the huge ciphertext and public key sizes. A full comparison in terms of usual parameters is given in Table 5.2.

Scheme	Max coll.	Failure prob.	Pirate success(σ)	Tracing round complexity
T&R scheme of BW [43]	n	ϵ	$4n\epsilon_s$	$O(\frac{n^2 \log n}{\sigma^2} \log \frac{1}{\epsilon})$
Our Instantiations	n	ϵ	$4n\epsilon_s + 2\delta$	$O(\frac{n^2}{\delta^2} \log \frac{n}{\epsilon} \log \frac{n^2 \log \frac{n}{\epsilon}}{\epsilon})$
Our Instantiations	w	ϵ	$4n\epsilon_s + 2\delta$	$O(\frac{w^2}{\delta^2} \log \frac{n}{\epsilon} \log \frac{n^2 \log \frac{n}{\epsilon}}{\epsilon})$

Table 5.2: The table shows the comparison of our instantiations with the T&R scheme of [43] in terms of their tracing round complexities. We suppose $\epsilon = 2\epsilon_f = \epsilon'/2\ell$ to get the entries in the case of our construction.

5.6.2.1 Comparing Round Complexities

The round complexity (i.e., the number of queries needed in tracing) of [43] is $O(\frac{n^2 \log n}{\sigma^2} \log \frac{1}{\epsilon})$ where σ is the success rate of a useful pirate decoder that one might want to trace and ϵ is the failure probability of tracing. Our construction which uses [21] and Tardos codes [64], has a round complexity of $O(\frac{n^2}{\delta^2} \log \frac{n}{\epsilon} \log \frac{n^2 \log \frac{n}{\epsilon}}{\epsilon})$ if we set $\epsilon = 2\epsilon_f = \epsilon'/2\ell$ where ϵ_f is the failure probability of the fingerprinting code in identifying the pirate codeword and ϵ' is a special security parameter of the tracing algorithm that corresponds to the failure probability of deciding a wrong character for a single bit position of the codeword. Also recall that we assumed $\sigma \geq 4n\epsilon_s + 2\delta$, ϵ_s being the probability of symmetric encryption to be broken, which means $O(\sigma) = O(\delta)$ when we assume ϵ_s is negligible.

An interesting point of our method is that it allows partially collusion secure constructions and in such a construction, the round complexity of tracing is quadratic in terms of the collusion bound which we call w instead of n. This generalized case is given in Table 5.2 as a separate row.

5.7 Stronger Traceability Modes

5.7.1 Public Traceability

In Eurocrypt 2005, Chabanne, Phan and Pointcheval [59] introduced the notion of public traceability where tracing is a procedure that requires no secrets. A two user solution was presented in [59] and further improved to the multiuser setting with short transmissions in [38] and [60]. In above schemes, the public key size and the private key sizes are all linear in length of the fingerprinting code employed for key distribution. The trace and revoke scheme of [43] is also publicly traceable with shorter key sizes, i.e. $O(\sqrt{n})$ many, but requires higher bandwidth, i.e. it has a ciphertext length of $O(\sqrt{n})$.

Our proposed generic construction supports the public traceability as there is no tracing key. The fingerprinting code is used to variate the way receivers decrypt logically without affecting the key-distribution. The encryption is done through a sampler that is of public knowledge, and any third-party can trace by generating a code. The code may have secrets available to the tracing party but this does not affect any other party to run her tracing capability. Hence, we provide the first publicly traceable schemes that have constant private key sizes with reasonable public key size and ciphertext length.

5.7.2 Tracing and Revoking Pirate Rebroadcasts

It is possible to obtain a scheme for tracing and revoking pirate rebroadcasting based on our generic construction. In such adversarial setting, an adversary, corrupting a number of traitors, decrypts the message through the key material available to him and rebroadcasts the clear message. Note that the rebroadcast does not reveal any information about the traitor-keys unless the clear message itself is bound to a specific user key. In this direction, we transmit different versions of the content so that each version is decryptable by different set

of keys. This is achieved in the literature by watermarking the content. In such setting, traitor-identification is achieved through observing the pattern of watermarks available to the pirate.

Let us provide a simple description on how to make our generic transformation work in the pirate rebroadcasting setting. We first generate the watermarked versions of the content m denoted by $m_1, m_2, \ldots, m_q$. For simplicity, we prefer an encryption in the standard model, a KEM version is possible by replacing m_i with a further level of symmetric encryption key. Similar to the original construction, we have a partition $\{V_1, V_2, \ldots, V_q\}$ of $[n]$ (the choice of the partition is through a sampler in regular transmission and through fingerprinting code in tracing transmissions). Setting $S_i = V_i \cap S$ for each $i = 1, \ldots, q$, we broadcast the message $c = (c_1||c_2|| \ldots ||c_q)$ where, for each $i = 1, \ldots, q$, we construct $c_i = hdr_i||e_i$ and

$$(hdr_i, K_i) \leftarrow \text{BE}_{\text{NC}}PK_{\mathtt{B}}, S_i, \quad e_i \leftarrow SEnc_{K_i}(m_i)$$

The traceability of the scheme above can be proven in almost exact way as we did for the original transformation in Section 5.5. We will not require linear tracing strategy as watermarking already differentiates the way we encrypt for different subsets in the partition.

Our generic scheme, instantiated with any of the schemes [21, 65, 66], will lead the first tracing and revoking pirate rebroadcasts in the public key setting with constant private key size and short transmission lengths.

5.8 Discussion

In this chapter, we have proposed a method for converting any BE scheme into a T&R scheme. The tracing round complexity of a T&R scheme generated by our generic method turns out to be quite decent. Performance of such a scheme turns out to be asymptotically same as that of the BE used as a primitive.

Chapter 6

Anonymous Trace and Revoke using Broadcast Encryption

As we see in the previous chapters, in a digital content distribution system, both BE and T&R are crucial features. However, while these advances are done in DRM systems, interestingly, a very important feature, privacy, was ignored. In almost all BE schemes and all T&R systems, the set of authorized users are sent together with the broadcast and readily available to everyone eavesdropping the broadcast. Although this is indeed a big privacy concern in many digital content distribution systems it is addressed in only three works so far [75, 76, 77], all being BE schemes.

The BE scheme of Barth et al. [75] uses a key-private IND-CCA secure public key encryption (PKE) scheme. The security of this work is proved in the random oracle model. In another paper of these authors [37] a private linear broadcast encryption primitive is used but only as a building block to achieve a tracing functionality over their BE scheme.

In a recent work [76], Fazio and Perera propose what they call an outsider anonymous broadcast encryption scheme which achieves a weaker notion of anonymity. In this work, the identities are hidden only from outsiders and

authorized users are still allowed to learn other authorized users.

The most recent and complete work on anonymous BE schemes is by Libert et al. [77], where anonymous BE is achieved either from key-private IND-CCA secure PKE schemes or any identity based encryption scheme. Their main construction yields the most efficient anonymous BE scheme and they prove adaptive IND-CCA security for all of their constructions.

In this chapter we employ our tracing technique from the previous chapter that makes use of fingerprinting codes in order to obtain anonymous T&R schemes from anonymous BE schemes. As a result, we propose the first anonymous T&R scheme.

The rest of the chapter is organized as follows. Section 6.1 gives the formal definitions of the cryptographic primitives we will use, and the security definitions regarding them. Then we will explain the generic method for obtaining anonymous T&R from anonymous BE schemes. Finally, we conclude the paper with the conclusion section.

6.1 Preliminaries and Definitions

6.1.1 Boneh-Shaw Fingerprinting Codes

Since we will use Boneh-Shaw fingerprinting codes in this paper, we briefly give their definition here.

In order to obtain an n-collusion secure Boneh-Shaw (ℓ, n)-code with ϵ-error probability, we first need to form a n-by-$(n-1)d$ matrix, where d is the block size which has a close relation to the performance of the code that will be explained later. ℓ is the alphabet size of the code. In this matrix, ith row consists of $d(i-1)$ 0s, followed by all 1s. For example, for a $(4, 2)$-code with

$d = 3$, we form the following matrix:

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Then, we choose a $(n-1)d$-permutation π to permute the columns. This permutation plays an important role and needs to stay hidden.

According to the original use of fingerprinting codes, all copies of the digital content to be sold is marked with a user codeword, which is a row in this matrix permuted by π. The idea is that when users with different copies marked in different positions of the content forms a coalition, they will not be able to obtain a copy that will traced back to an innocent user. In order to see the idea, consider users 1,3, and 4 forms a coalition. Then, their descendant set will consist of pirate codewords having same letters (bits in this case) in certain positions and they will not be able to know which ones to alter in order to frame user 2. Note that this is due to the hidden permutation, otherwise, they would know that they need to leave first half as 0 and the second half as 1.

For complete security analysis of Boneh-Shaw codes, see [63].

6.1.2 Anonymous Broadcast Encryption

In a usual BE setting, there are n users, each of which maintains a set of keys distributed by the broadcaster beforehand. Throughout the chapter we will denote the set of all users $\{1, 2, \ldots, n\}$ by $[n]$. When the broadcaster wants to broadcast a message, it encrypts the message in such a way that only the intended users can decrypt it. In order keep our construction as general as possible, we consider the following model of BE consisting of three algorithms (`KeyDist`, `Encrypt`, `Decrypt`):

- `KeyDist`(1^n) generates private keys sk_i for users $i \in [n]$ and a public

key PK. (These private keys are assumed to be transferred to the user securely. For example they can be embedded into users' devices beforehand.)

- `Encrypt`(PK, S) prepares an enabling header hdr and a master key K for the intended receiver set $S \subseteq [n]$.
- `Decrypt`(PK, sk_i, hdr), using their private key sk_i, users try to decrypt the header hdr to retrieve the master key K.

Note that the decrypt algorithm does not get S as a parameter. Although in many BE schemes, this would not mean anything since hdr can implicitly include the information of S, but since we are dealing with anonymous BE schemes, it is obvious that hdr must not leak any information about the set S. Of course, we will show that our generic construction inherits this property from the anonymous BE scheme used.

One may also note that there is no message in the definition above. This is due to the model we adhere to, namely the Key Encapsulation Mechanism (KEM) model, where the encrypt algorithm encrypts a master secret key instead of the message itself, and the actual message is encrypted with a secure secret key encryption scheme such as AES using this master secret key. We use this model because of two reasons. First, any BE scheme can be used as a KEM by encrypting a randomly chosen master secret key instead of the message, which makes KEM model more general in the sense of capturing the most BE schemes. Second, KEM is the natural use of BE schemes in real world applications since encrypting the possibly large message/data with a fast secret key encryption scheme is favorable to encrypting it directly with the BE scheme for performance reasons obviously.

Correctness and confidentiality definitions for BE schemes are same as the previous chapter, therefore we proceed with the definition of anonymity for BE schemes.

6.1.2.1 Anonymity

Now we formally define anonymity property for BE schemes. We will first define the anonymity game for BE schemes in the KEM model against CCA adversaries as follows.

Game 4 (Broadcast ANO-KEM-IND-CCA2 Game) *Both $\mathcal{A}$ and $\mathcal{C}$ are given the number of users, n.*

- ***Setup.*** *$\mathcal{C}$ runs* $\texttt{KeyDist}(1^n)$ *to obtain private keys $sk_1, \ldots, sk_n$ and the public key PK. Then, $\mathcal{C}$ sends the public key PK to $\mathcal{A}$.*
- ***Query Phase 1.*** *$\mathcal{A}$ makes polynomially many private key and decryption queries of the form (i) and (i, hdr) where $i \in [n]$, respectively. $\mathcal{C}$ responds with sk_i and $K \leftarrow$* $\texttt{Decrypt}(PK, sk_i, hdr)$*, to respective types of queries.*
- ***Challenge.*** *$\mathcal{A}$ chooses two sets S_0 and S_1, and sends them to $\mathcal{C}$. $\mathcal{C}$ selects a random bit b runs algorithm* $\texttt{Encrypt}(PK, S_b)$ *and obtains (hdr^*, K^*). The challenger then sends hdr^* to $\mathcal{A}$ as the challenge (header).*
- ***Query Phase 2.*** *$\mathcal{A}$ makes polynomially many decryption queries of the form (i, hdr) where $i \in [n]$, and $hdr \neq hdr^*$. $\mathcal{C}$ responds with $K \leftarrow$* $\texttt{Decrypt}(PK, sk_i, hdr)$.
- ***Guess.*** *$\mathcal{A}$ guesses b' for b and wins if $b' = b$.*

Definition A broadcast encryption scheme B is ϵ-anonymous in the ANO-KEM-IND-CCA2 model if for any polynomial time attacker $\mathcal{A}$ we have

$$Adv_{\mathcal{A}} = |Pr[\mathcal{A} \text{ wins}] - 1/2| = |Pr[b' = b] - 1/2| \leq \epsilon$$

where $Adv_{\mathcal{A}}$ denotes the advantage of the attacker $\mathcal{A}$ for winning the anonymity game described above.

6.1.3 Anonymous Trace and Revoke

A T&R system is essentially a BE scheme with tracing capabilities. So, we may define a T&R system by the following four algorithms:

- `Setup`(1^n) generates private keys $sk_1, \ldots, sk_n$, public key PK and possibly a tracing key TK.
- `Transmit`(PK, S) prepares a header hdr and a symmetric key K.
- `Receive`(PK, sk_i, hdr), using the private key sk_i, decrypts the header hdr to retrieve the key K. It will be successful if and only if user i is in the subset S that is used in the `Transmit` algorithm but not actually transmitted.
- `Trace`$(S, \mathcal{D}_S^\sigma, PK, TK)$ identifies a set of traitors, denoted by $A \subseteq S$, whose key(s) must have contributed in the construction of the pirate decoder $\mathcal{D}_S^\sigma$.

Assumptions: In this chapter, we consider black-box tracing where decoders cannot be reverse engineered and the keys inside cannot be revealed directly. Instead, tracer is only allowed to make transmissions to the pirate decoder and see whether it can decrypt or not (i.e., it has black-box access to it). Pirate decoders are also assumed to be resettable (does not maintain state during tracing) and available (remains available as long as the tracing process continues). In the literature, almost all of the positive results in designing traitor tracing schemes (including the schemes that we compare to our constructions) are successful against such decoders.

Correctness, confidentiality, and anonymity definitions for T&R schemes are the same as their BE counterparts. So we skip them here. There is one additional property for T&R systems, though, which is traceability. Traceability is defined via the following game between an attacker $\mathcal{A}$ and a challenger $\mathcal{C}$:

Game 5 (Tracing Game) *Both $\mathcal{A}$ and $\mathcal{C}$ are given the number of users, n, and the upper bound t on traitor coalition size.*

- ***Request.*** *$\mathcal{A}$ chooses a traitor subset T of size at most t and requests their private keys from $\mathcal{C}$.*
- ***Provide.*** *$\mathcal{C}$ runs* $\texttt{Setup}(1^n)$ *to obtain the keys. Then, $\mathcal{C}$ sends all sk_i such that $i \in T$ and the public key PK to $\mathcal{A}$. It keeps the tracing key TK.*
- ***Forge decoder.*** *$\mathcal{A}$ chooses a set S, and creates a (σ, S)-pirate decoder box $\mathcal{D}_S^\sigma$ which, by definition, correctly decrypts the broadcasts to set S with probability at least σ. It outputs $\mathcal{D}_S^\sigma$.*
- ***Trace.*** *The challenger $\mathcal{C}$ runs* $\texttt{Trace}(S, \mathcal{D}_S^\sigma, PK, TK)$ *to obtain an accused traitor set $A \subseteq S$.*

Attacker $\mathcal{A}$ wins the game if the set A is empty or it is not a subset of T.

Definition We say that $\texttt{T=}$ $(\texttt{Setup}, \texttt{Transmit}, \texttt{Receive}, \texttt{Trace})$ is a T&R scheme with tracing success probability α against t-coalition σ-pirates if no polynomial time attacker $\mathcal{A}$, forging a σ-decoder by corrupting a traitor coalition of size t, can win the game described above with probability more than $1 - \alpha$.

6.2 Generic Transformation

In this section, we show how to transform any anonymous broadcast encryption (ABE) scheme into an anonymous T&R scheme by employing a tracing mechanism. This transformation requires a simple modification in the regular transmission of the ABE scheme: whenever a transmission is to be made to a set S, instead of encrypting directly for S, we first partition S into two subsets S_1, S_2 and encrypt for both of them separately.

This partitioning is the base of our tracing method. More specifically, in the tracing transmissions, we send two different messages to different partitions and according to which one the pirate decrypts, we deduce which side the traitor key resides. But the transmission is done according to a fingerprinting code so that the results of several partitionings identifies a specific traitor with overwhelming probability as we will show in our analysis. The T&R construction will inherit the anonymity of the primitive ABE since neither side will leak information about their respective S_i. With this intuition, we will prove anonymity of the T&R scheme obtained.

Let $\mathtt{B}$ be an ABE scheme consisting of three algorithms $\mathtt{B.KeyDist}(1^n)$, $\mathtt{B.Enc}(PK_{\mathtt{B}}, S)$, and $\mathtt{B.Dec}(PK_{\mathtt{B}}, sk_i, hdr)$. We design the algorithms of our generic scheme $\mathtt{T}$, $\mathtt{TKeyDist}$, $\mathtt{Transmit}$, $\mathtt{Receive}$ and $\mathtt{Trace}$ as follows.

- $\mathtt{TRKeyDist}(1^n)$ runs the key distribution algorithm $\mathtt{BKeyDist}(1^n)$ of $\mathtt{B}$. This will produce a public key $PK_{\mathtt{B}}$ and a set of private keys sk_i, $i \in U$, which will be distributed to the receivers. A symmetric encryption scheme, $\mathtt{Sym}$ = ($\mathtt{Sym.Enc}$, $\mathtt{Sym.Dec}$), is determined to be used in transmissions. It sets up a Boneh-Shaw fingerprinting code $\mathtt{F}$ = ($\mathtt{CodeGen}$, $\mathtt{Identify}$), and some auxiliary information aux. Here, we note that the actual codewords are not generated at this moment. Finally, the public key will be $PK_{\mathtt{T}} =< 1^n, PK_{\mathtt{B}}, \mathtt{Sym}, \mathtt{F}, aux >$.

$\mathtt{Transmit}$ and $\mathtt{Receive}$ algorithms are depicted in Figures 6.1 and 6.2. We design our transmission algorithm in the KEM setting, i.e. we do not take any message as input but rather choose a random encapsulation key K to be used in symmetric encryption of the message.

- $\mathtt{Transmit}(PK_{\mathtt{T}}, S)$ The algorithm first choses a random key K to be transmitted and a partition $\{V_1, V_2\}$ of $[n]$. It sets $S_i = V_i \cap S$ for $i = 1, 2$. The transmission algorithm then runs the encryption algorithm of the BE scheme for both subsets and broadcasts the message $\hat{c} = (hdr, c)$

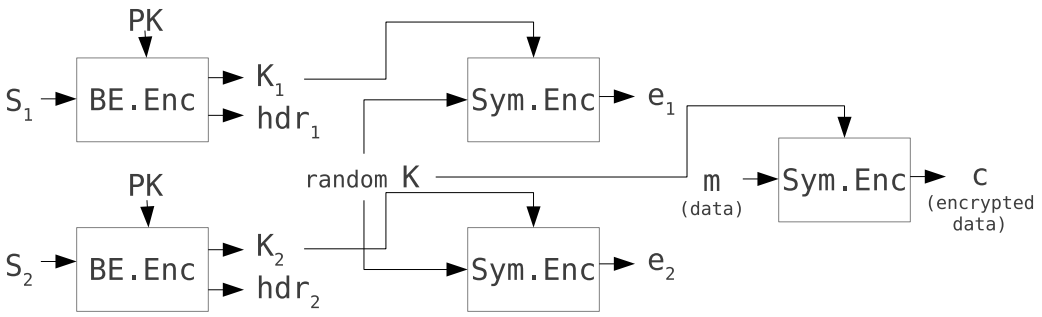


Figure 6.1: `Transmit` algorithm is shown. First two BE encryption algorithms are run in order to get two different encapsulations, (K_1, hdr_1) and (K_2, hdr_2). Then, a random master encapsulation key K is chosen and encrypted with the symmetric scheme using both keys K_1 and K_2. The actual data is encrypted only once with K. The values that actually gets transmitted are: $\hat{c} = (hdr, c)$ where $hdr = (hdr_1, hdr_2, e_1, e_2)$.

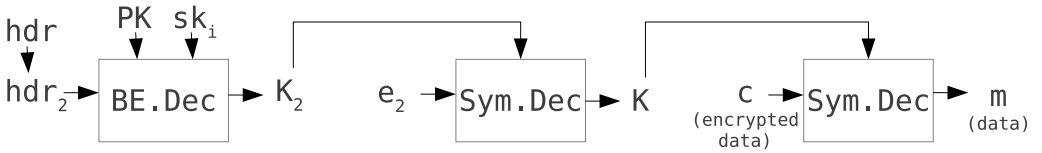


Figure 6.2: `Receive` algorithm for a user $i \in S_2$ is shown. So, when it parses $\hat{c}$, obtains hdr_1 and hdr_2 and when it tries the second encapsulation as in the figure, since it is assumed to be in the set S_2, it can obtain m by first finding K_2 via the BE decryption algorithm, then finding K by decrypting e_2 with K_2 and then decrypting c with K.

where $hdr = (hdr_1, hdr_2, e_1, e_2)$ and for $i = 1, 2$,

$$(hdr_i, K_i) \leftarrow \mathtt{BE.Enc}(PK_{\mathtt{B}}, S_i), \quad e_i \leftarrow \mathtt{Sym.Enc}_{K_i}(K)$$

A special attention is needed to understand why we use a two-level key encapsulation mechanism. First of all, we would like to encrypt the data only once with a symmetric scheme, because the data is typically long and it can be expensive to encrypt it directly with any BE mechanism. However, it is impossible to force both BE encryptions to produce the same key. Therefore, the symmetric key to encrypt the data, K, is selected randomly and this key is encapsulated by encrypting it with the keys produced by the BE encryptions. Note that the BE primitive is also formulated as a KEM. This is because we want to make our transformation be applicable for all BE schemes, we have to take into account that in some broadcast encryption schemes that support key-encapsulation (e.g. the scheme of [21]), the sender has no control on the choice of the key (K_i's in our construction) transmitted.

- $\mathtt{Receive}(PK_{\mathtt{T}}, sk_j, \hat{c})$ The j-th user parses the public key $PK_{\mathtt{B}}$ from $PK_{\mathtt{T}}$ and extracts $(hdr_1, hdr_2, e_1, e_2, c)$ from $\hat{c}$. It then tries both hdr_1 and hdr_2 with the decryption function of $\mathtt{B}$ and retrieves the key K as follows:

  $$K_k \leftarrow \mathtt{BE.Dec}(PK_{\mathtt{B}}, sk_j, hdr_k)$$

  and the valid side is used to obtain K:

  $$K \leftarrow \mathtt{Sym.Dec}_{K_k}(e_k)$$

Now we explain our tracing algorithm. As usual, we assume that a pirate decoder with a certain ability is detected and the tracer can make queries to it with ciphertexts of its choice. We call these ciphertexts as *tracing ciphertexts*. In our algorithm, the tracer prepares tracing ciphertexts as follows:

We first generate a Boneh-Shaw code $\mathcal{C} = \{c_1, \ldots, c_n\}$ of length ℓ by running $\mathcal{F}.\mathtt{CodeGen}(1^n)$. Then, for every column $j \in [\ell]$ of this code, we prepare

new kind of tracing ciphertext. For the j-th column, the partition of a set S in a tracing ciphertext is based on this particular j-th column of the code rather than partitioning randomly as in normal ciphertexts. Denoting the j-th partition by $S_j = \{S_{j,0}, S_{j,1}\}$, the subset $S_{j,i}$ is chosen to be $S \cap \{k : c_k[j] = i\}$. In other words, S is partitioned according to 0s and 1s in the j-th column. However, we encrypt a random message to $S_{j,0}$ whereas we encrypt a valid message m to $S_{j,1}$. Therefore, if the pirate decoder can decrypt this tracing ciphertext, i.e., if it can retrieve m, we deduce that the letter of the pirate codeword in this position (column) is 1. Otherwise, we deduce that it is 0.

After we make several tries for each column, we get a pirate codeword w. (The number of tries for each column will be explained below, in the formal description.) Then we give this codeword to the $\mathcal{F}.\texttt{Identify}$ algorithm which will output a user index that is responsible for the acts of the pirate decoder.

Now we describe the tracing algorithm formally. For simplicity, we consider tracing against the pirate decoders of type $\mathcal{D}_S^1$ first. Such a decoder is called a perfect decoder as it correctly decrypts all well-formed ciphertexts. In reality, the pirate may be content with a decoder that works only a fraction of the time, that is formulated in our definition as σ-pirate with $\sigma \leq 1$. A solution for $\sigma < 1$ values will be discussed later in Section 5.5.4.

- $\texttt{Trace}(S, \mathcal{D}_S^1, PK_{\texttt{T}}, TK)$ first parses $1^n, PK_{\texttt{B}}$, $\texttt{Sym}$ and the description of $\texttt{F}$ from $PK_{\texttt{T}}$. It produces a Boneh-Shaw fingerprinting code $\mathcal{C} = \{c_1, \ldots, c_n\} \leftarrow \mathcal{F}.\texttt{CodeGen}(1^n)$ and initializes a pirate codeword $w \leftarrow 0^\ell$. Denoting the length of the code $\mathcal{C}$ by ℓ, the tracing algorithm repeats the following sub-procedure for each $j = 1, \ldots, \ell$:

  We create a partition $\{S_{j,0}, S_{j,1}\}$ of S where $S_{j,v} \leftarrow S \cap \{k : c_k[j] = v\}$ holds for $v = \{0, 1\}$. Next, we approximate the decryption probability of $\mathcal{D}_S^1$ when queried by tracing ciphertexts of type (j, v) which consists of the transmission $c = (c_0||c_1)$ where, for each $i = \{0, 1\}$, we have

$c_i = hdr_i || e_i$, where $(hdr_i, K_i) \leftarrow \texttt{BE.Enc}(PK_{\texttt{B}}, S_i)$ and

$$e_i \leftarrow \begin{cases} \texttt{Sym.Enc}_{K_i}(R), & i = 0 \\ \texttt{Sym.Enc}_{K_i}(K), & i = 1 \end{cases}$$

for randomly chosen R and K. We say the decoder succeeds in decryption if it returns K. If the the decoder returns K, we set $w_j \leftarrow 1$, otherwise $w_j \leftarrow 0$.

Once we perform this procedure for all j, we get a pirate codeword w and we use this codeword with the identification algorithm of the fingerprinting code: $t \leftarrow \mathcal{F}.\texttt{Identify}(w)$. Finally, the user associated with the t-th codeword is accused of being a traitor.

To have a correct tracing, it is important that the produced pirate codeword is in the descendant set of the traitor coalition T. This is equivalent of claiming that if $w_j = s$ then $S_{j,s} \cap T \neq \emptyset$. As we will argue formally later in our traceability proof, this can be ensured by the security of the underlying primitives.

6.2.1 Security Properties of the Construction

In this section, we will prove the security properties achieved by our generic construction. Here we state the complete theorem and in the following subsections we will prove three different properties, confidentiality, anonymity and traceability, with three lemmas which together will constitute the proof of the theorem.

Theorem 6.2.1 *[Security properties of the generic construction] Consider the generic anonymous T&R scheme* `T` *that is constructed as above by employing an anonymous BE scheme* `B`*, a symmetric encryption scheme* `SYM` *and Boneh-Shaw fingerprinting code* `F`*.*

Let $\mathtt{B}$ *be KEM-IND-CCA secure with probability* ϵ_b, $\mathtt{SYM}$ *be IND-CCA secure with probability* ϵ_s *and* $\mathtt{F}$ *be an* (ϵ_f, t)*-identifier Boneh-Shaw fingerprinting code.*

Then, $\mathtt{T}$ *is an anonymous T&R scheme satisfies confidentiality with respect to Definition 5.3.1.2, anonymity with respect to Definition 6.1.2.1, and traceability with respect to Definition 6.1.3.*

6.2.1.1 Broadcast Confidentiality

We first prove the security of our construction regarding the confidentiality of the broadcast messages.

Lemma 6.2.2 (Confidentiality) *Let* $\mathtt{T}$ *be a trace and revoke scheme that is constructed through our generic transformation from a BE scheme* $\mathtt{B}$ *that is* ϵ_b*-secure in the KEM-IND-CCA model, a symmetric encryption scheme* $\mathtt{S}$ *that is* ϵ_s*-secure in the IND-CPA model, and a Boneh-Shaw fingerprinting code* $\mathtt{F}$*. Then,* $\mathtt{T}$ *would satisfy the KEM-IND-CCA security for any polynomial time attacker* A_T *such that*

$$Adv_{A_T} \leq 2 \cdot \epsilon_b + 2 \cdot \epsilon_s$$

It further holds that if the underlying scheme $\mathtt{B}$ *supports adaptive security then so does the scheme* $\mathtt{T}$*.*

Proof We will use a game hopping approach to prove the lemma: we start with the basic confidentiality game for trace and revoke scheme. We next modify the basic game gradually to reach a final game which provides the adversary no advantage. This is a standard proof technique that bounds the advantage of the adversary in the original game by the differences in the subsequent games.

Let G_0 be the real KEM-IND-CCA message confidentiality game (Game 1) played between an adversary A_T and a challenger C_T of the generic T&R construction. The details of this game is depicted in Figure 6.3. We are interested in showing that the success probability of any attacker in this game is negligible.

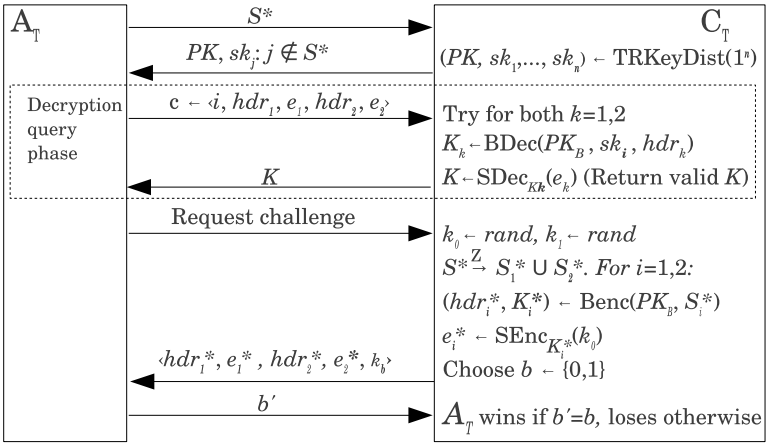


Figure 6.3: Game G_0: the actual KEM-IND-CCA game.

Remark. In the figure, we considered a static attack model where the adversary commits to the set S^* it wants to attack. The challenger publishes the public key afterwards. In contrast, it is also possible that the adversary commits to the set S^* after it observes the public key. The latter, denoted by adaptive attack, is a stronger attack model as the public key may let the adversary have some non-trivial information that is useful for the choice of the target set. The order of the commitment of the target set and the publication of the public key will not affect the validity of our proof arguments below. The choice of the order is propagated in our transformation smoothly. Hence we will consider the security for static attack model, the adaptive case follows in a similar way.

Now, we proceed with the formal description of the subsequent games starting with G_0. Let W_j denote the event that the adversary A_T wins the j-th game G_j:

Game 0: The first game, G_0, depicted in Figure 6.3 is identical to the KEM security game for trace and revoke schemes. Thus,

$$|Pr[W_0] - \frac{1}{2}| = Adv_{A_T}$$

In this game, after the key distribution and decryption query phases, the challenger prepares a valid ciphertext. It performs the partitions $\{S_i^*\}_{i\in\{1,2\}}$ in a way indistinguishable from Boneh-Shaw codes and constructs the headers

$$(hdr_1^*, K_1^*) \leftarrow \texttt{BE.Enc}(PK_{\texttt{B}}, S_1^*), \quad e_1^* \leftarrow \texttt{Sym.Enc}_{K_1^*}(k_0)$$

$$(hdr_2^*, K_2^*) \leftarrow \texttt{BE.Enc}(PK_{\texttt{B}}, S_2^*), \quad e_2^* \leftarrow \texttt{Sym.Enc}_{K_2^*}(k_0)$$

where k_0 and k_1 are randomly chosen keys compatible with the symmetric encryption algorithm `Sym.Enc`. Along with the full headers $\langle hdr_i^*, e_i^*\rangle_{i\in\{1,2\}}$, the challenger transmits k_b for a randomly selected bit b. So, the challenge question is "Is this key the same as the encrypted one?" We say the adversary wins the game if it guesses b correctly.

Game 1: This game (G_1) is identical to G_0 except the way the challenger prepares the encryption for e_1^*. In G_1, the encryption e_1^* is made under a randomly chosen key K_1^+ instead of K_1^*:

$$e_1^* \leftarrow \texttt{Sym.Enc}_{K_1^+}(k_0)$$

Such modification breaks the relation between the header hdr_1^* and e_1^*. We claim that any adversary that can distinguish its views among games G_0 and G_1 can be used to break the security of the underlying BE scheme. Formally stated, there exists a broadcast encryption adversary A_B whose running time is about the same as A_T such that:

$$|Pr[W_0] - Pr[W_1]| = 2Adv_{A_B}$$

Figure 6.4 shows how to construct such an adversary A_B that intends to break the KEM-IND-CCA security of the broadcast encryption `B`. The adversary A_B will simulate the challenger C_T of the trace and revoke security game. The simulator will embed the challenge it receives from the broadcast challenger C_B to the challenge requested by the adversary A_T.

After receiving the set S^*, the simulator will create the partition $S^* = \{S_1^*, S_2^*\}$ immediately and forwards S_1^* to the broadcast encryption challenger C_B. This is a crucial step to be able to simulate the secret keys of the

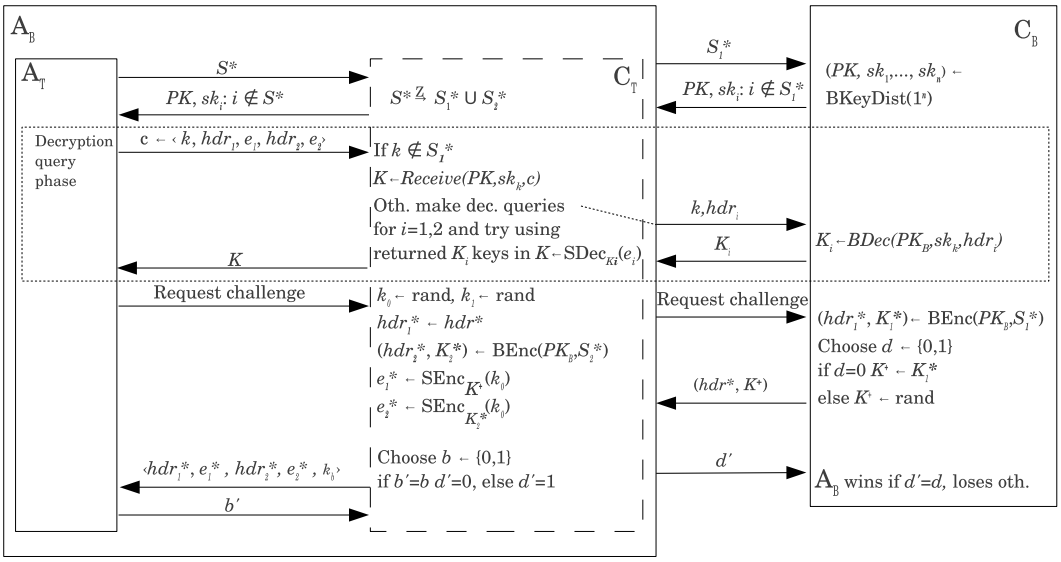


Figure 6.4: Constructing a broadcast encryption adversary A_B that simulates the challenger of the trace and revoke adversary A_T. Its advantage is reduced to the A_T's ability of distinguishing its views among games G_0 and G_1.

scheme $\mathtt{T}$ whose keys are basically the keys of the underlying broadcast encryption scheme. The simulator will be able to respond the decryption queries $\langle k, hdr_1, e_1, hdr_2, e_2\rangle$ of the adversary A_T as long as the secret key of the intended user k is available to the adversary A_B. Otherwise, the adversary forwards the decryption queries $\langle k, hdr_j^*\rangle$ for $j = 1, 2$ to the challenger C_B and retrieves the key to decrypt the symmetric encryption e_j^*.

After requesting the challenge from A_T, the adversary A_B simulates the challenger C_T as follows: The first side of the challenge comes from C_B which provides (hdr^*, K^+) as its challenge. Upon receiving (hdr^*, K^+) from the challenger C_B we set $hdr_1^* = hdr^*$ and $e_1^* = \mathtt{Sym.Enc}_{K^+}(k_0)$. The second side of the challenge is easily produced by the simulator as $(hdr_2^*, K_2^*) \leftarrow \mathtt{BE.Enc}(PK_B, S_2^*)$ and $e_2^* \leftarrow \mathtt{Sym.Enc}_{K_2^*}(k_0)$.

Observe, now, that if the challenge of C_B is a valid BE ciphertext (this corresponds to the case $d = 0$ in Figure 6.4), the adversary A_T plays in G_0. In contrast, A_T plays in G_1 if the challenge is not valid ($d = 1$ in Figure 6.4). Let us compute the winning probability of A_B: (i) if $d = 0$ A_B wins

the game if A_T wins the game, hence bounded by $Pr[W_0]$; (ii) if $d = 1$ A_B wins the game if A_T loses the game, hence bounded by $1 - Pr[W_1]$. Thus, $|Pr[W_0] - Pr[W_1]| = Adv_{A_B}$.

Game 2: G_2 is the same as G_1 except that this time, second side of the T&R challenge comes from the BE challenge. The first side is prepared as an invalid encryption by the simulator. With the same reasoning above, $|Pr[W_1] - Pr[W_2]| = Adv_{A_B}$. We omit the details since it will be almost the same as above.

Note that, at this point, both BE keys K_1^* and K_2^* are distorted. Hence, $|Pr[W_0] - Pr[W_2]| = 2Adv_{A_B}$ which is then upper-bounded by $2\epsilon_b$.

We continue with two more games gradually replacing the key k_0 with random keys.

Game 3: G_3 is identical to G_2 except the way the challenger prepares the encryption for e_1^*. In Game G_3, we set:

$$e_1^* \leftarrow \texttt{Sym.Enc}_{K_1^+}(k_0^+)$$

where k_0^+ is randomly chosen. Such modification in Game G_3 hides totally the information of k_b in the first part of the header. We next claim that there exists a symmetric encryption adversary A_S whose running time is about the same as A_T such that:

$$|Pr[W_2] - Pr[W_3]| = Adv_{A_S}$$

We construct the adversary A_S in a similar way we have constructed the adversary A_B. We omit the details of the simulation due to simplicity and similarity. Hence the probability differences above are upper-bounded by ϵ_s.

Game 4: As the reader might guess, G_4 is identical to G_3 except that the way the challenger prepares the encryption for e_2^*. Once again, same analysis as the last one leads us

$$|Pr[W_3] - Pr[W_4]| = Adv_{A_S}$$

Note that this last game G_4 gives absolutely no information about k_b thus the probability $Pr[W_4]$ of the adversary winning the game G_4 is $\frac{1}{2}$. Applying the triangular inequalities over the probability differences above we obtain:

$$2 \cdot \epsilon_b + 2 \cdot \epsilon_s \geq Adv_{A_T}$$

which completes the security proof of our generic transformation.

6.2.1.2 Anonymity

In this section we will show that if the underlying BE scheme is anonymous, so is the resulting T&R scheme produced by our construction.

Lemma 6.2.3 *[Anonymity of the generic construction] Given the construction described in Theorem 6.2.1, the resulting T&R scheme satisfies anonymity with respect to Definition 6.1.2.1.*

Proof The standard anonymity game applied to our generic construction is shown in Figure 6.5.

We prove the lemma by showing that given an attacker that breaks the anonymity of the T&R scheme, we can build an attacker that breaks the anonymity of the underlying ABE scheme.

Construction of such an attacker is as follows: It simulates the T&R challenger in Figure 6.5. Upon receiving the challenge request (S_0, S_1), it finds the intersection $S_2 \leftarrow S_0 \cap S_1$. Then, it sends to the ABE challenger the challenge request $(S_0 \backslash S_2, S_1 \backslash S_2)$. It also makes the encryption $hdr_2 \leftarrow \texttt{BE.Enc}(PK, S_2)$. Upon receiving the challenge, hdr, it sets $hdr_1 \leftarrow hdr$ and sends its own challenge (hdr_1, hdr_2) to the T&R attacker A_T it has. The answer returned by A_T is exactly directly returned as the answer to C_B. Clearly, the success probability of A_B constructed is exactly the same as that of A_T and this concludes the proof.

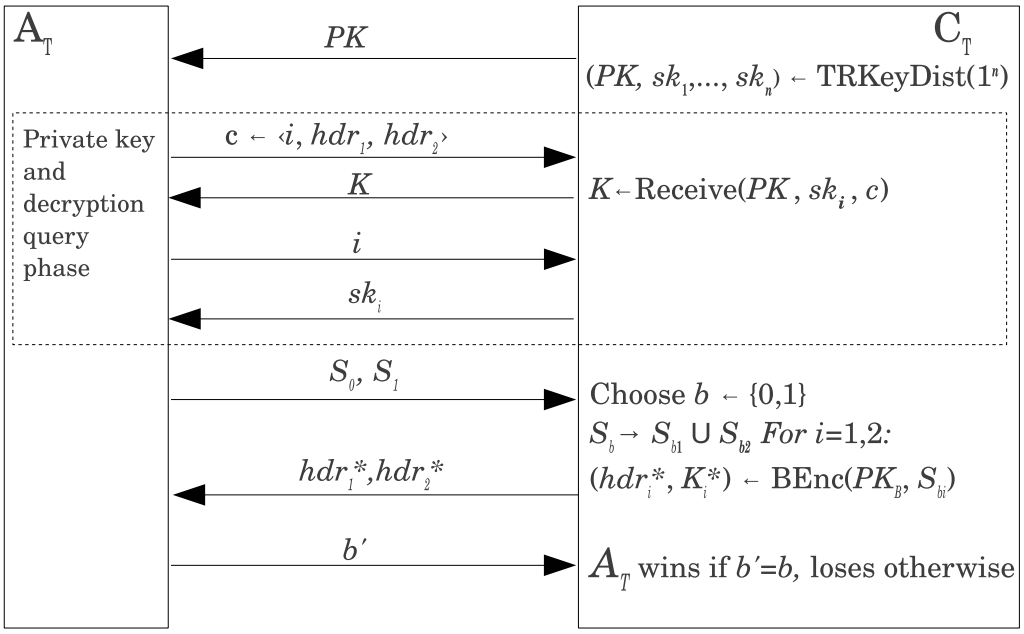


Figure 6.5: Standard anonymity game.

6.2.1.3 Traceability of Perfect Decoders

In this section, we will prove that perfect pirate decoders can be traced back to at least one of the traitor users that contributed to its forgery. A perfect decoder $\mathcal{D}_S^1$ is a decoder which can decrypt all messages addressed to set S. Specifically, we will prove that no polynomial time attacker A that forges a perfect decoder can win the tracing game (Game 5) with some non-negligible probability. i.e., we will bound the winning probability of such an attacker A by a function of the security bounds of the underlying primitives. We leave the tracing of imperfect decoders to the next section.

Before we prove the traceability of our scheme, we need to point out one subtle yet important issue. Recall that in regular transmissions, the receiver subset is partitioned by imitating a Boneh-Shaw code, i.e., by selecting a random index before which all letters are thought as 0 and the rest are thought as 1. This guarantees statistical indistinguishability between tracing and regular ciphertexts. This is indeed a crucial point because if the pirate were to be able to distinguish between two types of ciphertext, it could have been designed

in such a way that in tracing ciphertexts, it can perform a special trick and frame an innocent user.

Lemma 6.2.4 *[Traceability of the generic construction] Consider the generic anonymous T&R scheme* `T` *that is constructed as above by employing an anonymous BE scheme* `B`*, and Boneh-Shaw fingerprinting code* `F`*. a symmetric encryption scheme* `SYM`

Let `B` *be KEM-IND-CCA secure with probability* ϵ_b*,* `SYM` *be IND-CCA secure with probability* ϵ_s *and* `F` *be an* (ϵ_f, t)*-identifier Boneh-Shaw fingerprinting code.*

Then, `T` *is an anonymous T&R scheme with success probability* $1-\epsilon_f-\ell\epsilon/4$ *against* t*-coalition* 1*-pirate's if it holds that*

$$\epsilon_s + \epsilon_b < 1/8$$

where ϵ *is a probability that can be decreased arbitrarily by increasing the number of experiments in the tracing procedure.*

Proof We consider a resettable pirate decoder $\mathcal{D}_S^1$ constructed for a subset $S \in [n]$ by coalitions of at most t traitors. The tracing process can be considered as three stages: (1) Approximating the success probability of choosing the correct value for each ℓ pirate codeword bits, (2) Producing a pirate codeword w in the descendant set, and (3) Identifying a traitor index through the fingerprinting code.

Before these steps, first remember that the trace algorithm performs experiments to find out individual pirate codeword bits. For a readable notation, for each bit position j, we define three experiments:

- $Exp_{j,0}$: Hypothetical experiment where both sides get valid messages.
- $Exp_{j,1}$: Experiments run by the tracer where only second part gets a valid message and first part gets a random one.

- $Exp_{j,2}$: Hypothetical experiment where both sides get random messages.

We define $\sigma_{j,v}$ as the expected success probabilities in experiments of type $Exp_{j,v}$. Since we assumed a perfect decoder, we say that $\sigma_{j,0} = 1$. Since experiments of type $Exp_{j,2}$ leaks no information about the message, $\sigma_{j,2} = 1/|M|$. For simplicity, we assume that $\sigma_{j,2} = 0$.

We also define the approximation counterparts: We define $p_{j,v}$ as the actual success ratios in respective experiments of type $Exp_{j,v}$. Since we do not need to actually approximate the hypothetical types of experiments, we assume that $p_{j,0} = \sigma_{j,0} = 1$ and $p_{j,2} = \sigma_{j,2} = 0$. In the first step, we will bound the difference between $p_{j,1}$ and $\sigma_{j,1}$.

(1) *Approximation:* First, assuming λ experiments are performed, we define $\mu_{j,v} = \lambda \cdot \sigma_{j,v}$, which is the expected number of times the decoder succeeds out of λ experiments of type (j, v) and $\rho_{j,v} = \lambda \cdot p_{j,v}$ as the actual number of successes during the approximation process. We would like to bound the approximation difference $|\rho_{j,1} - \mu_{j,1}|$ and thus $|p_{j,1} - \sigma_{j,1}|$. We claim that, choosing $\lambda = 3\ln(8/\epsilon)/\Delta^2$, $Pr[|\rho_{j,1} - \mu_{j,1}| \geq \lambda \cdot \Delta] \leq \epsilon/4$.

Due to the resettability of the decoder after each tracing query we can consider the experiments performed by the tracer are independent. By applying a two-tailed form of the Chernoff bound we will have:

$$Pr[|\rho_{j,1} - \mu_{j,1}| \geq \alpha] \leq 2e^{-\frac{\alpha^2}{3\mu_{j,1}}} \leq 2e^{-\frac{\alpha^2}{3\lambda}}$$

Substituting $\alpha = \lambda \cdot \Delta$ and $\lambda = 3\ln(8/\epsilon)/\Delta^2$ we obtain:

$$\begin{aligned} 2e^{-\alpha^2/3\lambda} &= 2e^{-\frac{\lambda^2\Delta^2}{3\lambda}} &&= 2e^{-\Delta^2\lambda/3} \\ &= 2e^{-\ln(8/\epsilon)} &&= \epsilon/4 \end{aligned}$$

The above analysis shows that $|\rho_{j,1} - \mu_{j,1}| \leq \lambda \cdot \Delta = 3\ln(8/\epsilon)/\Delta$ with probability at least $1 - \epsilon/4$. This is equivalent to $|p_{j,1} - \sigma_{j,1}| \leq \Delta$ with probability

at least $1 - \epsilon/4$.

(2) *Pirate Codeword Generation:* First, remember that the tracer sets $w_j = 0$ if $p_{j,1} < 1/2$ and $w_j = 1$ otherwise. We argue that the produced pirate codeword w is in the descendant set of the traitor coalition T with high probability. This is equivalent of claiming that, for all $1 \leq j \leq \ell$, if $w_j = b$ then $S_{j,b} \cap T \neq \emptyset$ for $b \in \{0, 1\}$. So, we need to show that with a high probability, for all positions, decision of the trace algorithm is not false.

Now, suppose that for jth position, tracer sets $w_j = 0$. We claim that with overwhelming probability, there exists a traitor in $S_{j,0}$. Since tracer sets $w_j = 0$ it must be the case that $p_{j,1} < 1/2$. We already assumed that $\sigma_{j,0} = p_{j,0} = 1$. Together with the inequality $|\sigma_{j,1} - p_{j,1}| \leq \Delta$ which has a proability at least $1 - \epsilon/4$, we have

$$|\sigma_{j,0} - \sigma_{j,1}| \geq |p_{j,0} - p_{j,1}| - \Delta$$

with probability at least $1 - \epsilon/4$. Now, since $p_{j,1} < 1/2$, and $p_{j,0} = 1$,

It follows that if the tracer returns the value 0, i.e.,

$$|\sigma_{j,0} - \sigma_{j,1}| \geq 1 - p_{j,1} - \Delta \geq 1/2 - \Delta$$

and choosing $\Delta = 1/4$ we get:

$$|\sigma_{j,0} - \sigma_{j,1}| \geq 1/4$$

Since we assumed that $\epsilon_s + \epsilon_b < 1/8$ at the beginning,

$$|\sigma_{j,0} - \sigma_{j,1}| \geq 1/4 > 2(\epsilon_s + \epsilon_b)$$

with probability at least $1 - \epsilon/4$.

In such a case, we can prove that there exists a traitor in $S_{j,0}$ by contradiction, as follows. Assume that $|\sigma_{j,0} - \sigma_{j,1}| > 2(\epsilon_s + \epsilon_b)$ and there exists no traitor in $S_{j,0}$. This contradicts with the security claims of the underlying symmetric encryption scheme `SYM` and broadcast encryption scheme `B`. Indeed, if there is no traitor in set $S_{j,0}$, the pirate decoder can distinguish between the

tracing ciphertext of type $(j, 0)$ and of type $(j, 1)$ by only breaking the underlying encryption schemes. Hence, the distinguishing probability is bounded by $2(\epsilon_s + \epsilon_b)$.

Due to symmetry, the case where tracer sets $w_j = 1$ succeeds with the same probability, $\epsilon/4$.

Finally, this analysis must be done for each $1 \leq j \leq \ell$, therefore the probability that a pirate codeword in the descendant set is created would be at least $1 - \ell\epsilon/4$.

(3) *Traitor Identification.* We argue above that the pirate codeword is in the descendant set of the traitor coalition. In our application of fingerprinting code, the partition in a tracing transmission does not hide the user codewords, i.e. the code is open to the adversary. Hence, `Identify(`w`)` returns a traitor index with probability at least $1-\epsilon_f$ as long as the fingerprinting code is open (not secret as in the cases of Tardos or Boneh-Shaw codes). This completes the proof of the traceability. The overall failure probability of accusing an innocent user is bounded by $\epsilon_f + \ell\epsilon/4$ (for the failures in identification, and in approximations, respectively) for the given parameters.

6.2.2 Instantiation with Libert et al. and its properties

Since the scheme of Libert et al. [77] is the most recent and complete anonymous BE scheme, we consider it for an instantiation of our method. When we instantiate our generic method with the scheme of [77], the complexities are not affected because the BE scheme requires linear in the number of users, and our generic construction not even doubles the ciphertext size because total number of users does not change. The resulting T&R scheme is indeed too expensive to be used as is, but still it is the first anonymous T&R scheme.

6.3 Discussion

We have proposed a generic way of obtaining anonymous T&R systems from anonymous BE schemes. Thus we obtained the first anonymous T&R system and showed that it satisfies IND-CCA security. This is indeed an important result that opens a direction towards considering anonymity in T&R systems.

Chapter 7

Conclusion

In this thesis, we have considered different types of broadcast encryption schemes and improved their performances or capabilities in different ways. First, we investigated the CS and SD schemes which are the standard methods in use today and we proposed using profiling in order to improve their transmission complexity. With the new advancements of technology, it may be possible to maintain such a system especially in systems like the pay-TV applications. Second, we considered free rider relaxation problem for the PI scheme, which is a recently developed BE scheme, having performances similar to that of SD scheme, and we showed that by allowing a decent amount of free riders, transmission complexity can be reduced significantly. This kind of relaxation can be used in systems where allowing free riders is not a big problem. For example sharing non-critical content on the Internet is a good candidate for being such a system, since neither the privacy of content is highly critical, nor the probability of a free rider to be interested in a content he can access as a free rider is high due to large number of users.

In the last two chapters, we considered broadcast encryption schemes in more detail, looking at their inside. We first proposed a generic way of obtaining a more capable system, called a trace and revoke system by using primitive broadcast encryption schemes. This lead us to new results such as the first

ID-based trace and revoke scheme in the literature, first publicly traceable trace and revoke schemes with short transmissions. It also allowed tracing and revoking pirate rebroadcasts in the public key setting. Finally, in the last chapter, we considered the mostly-ignored problem of user privacy in broadcast systems, and we showed how an anonymous trace and revoke scheme can be built using anonymous broadcast encryption schemes in a generic way. Instantiated with a recently proposed anonymous BE scheme, we obtained the first trace and revoke scheme that hides the receiver set.

As a result, we obtained a number of positive results that can stimulate new research in the context of digital rights management systems, while each individual result has its own novelty.

Bibliography

[1] J. Lotspiech, S. Nusser, and F. Pestoni, "Broadcast encryption's bright future," *Computer*, vol. 35, pp. 57–63, 2002.

[2] C. B. S. Traw, "Protecting digital content within the home," *Computer*, vol. 34, pp. 42–47, 2001.

[3] M. Abdalla, Y. Shavitt, and A. Wool, "Key management for restricted multicast using broadcast encryption," *IEEE/ACM Trans. Networking*, vol. 8, no. 4, pp. 443–454, 2000.

[4] W. Aiello, S. Lodha, and R. Ostrovsky, "Fast digital identity revocation," in *CRYPTO'98*, vol. 1462 of *LNCS*, pp. 137–152, Springer-Verlag, 1998.

[5] D. M. Wallner, E. J. Harder, and R. C. Agee, "Key management for multicast: Issues and architectures," 1999. Internet draft.

[6] C. K. Wong, M. Gouda, and S. S. Lam, "Secure group communication using key graphs," in *SIGCOMM'98*, pp. 68–79, September 1998.

[7] C. Blundo and A. Cresti, "Unconditional secure conference key distribution schemes with disenrollment capability," *Information Sciences*, vol. 120, no. 1-4, pp. 113 – 130, 1999.

[8] J.-T. Chung, C.-M. Li, and T. Hwang, "All-in-one group-oriented cryptosystem based on bilinear pairing," *Information Sciences*, vol. 177, no. 24, pp. 5651 – 5663, 2007.

[9] J. Nam, J. Paik, U. M. Kim, and D. Won, "Resource-aware protocols for authenticated group key exchange in integrated wired and wireless networks," *Information Sciences*, vol. 177, no. 23, pp. 5441 – 5467, 2007. Including: Mathematics of Uncertainty, A selection of the very best extended papers of the IMS-2004 held at Sarkaya University in Turkey.

[10] S. Berkovits, "How to broadcast a secret," in *EUROCRYPT'91*, vol. 547 of *LNCS*, pp. 535–541, Springer-Verlag, 1991.

[11] A. Fiat and M. Naor, "Broadcast encryption," in *CRYPTO'93*, vol. 773 of *LNCS*, pp. 480–491, Springer-Verlag, 1993.

[12] D. Naor, M. Naor, and J. Lotspiech, "Revocation and tracing schemes for stateless receivers," in *CRYPTO'01*, vol. 2139 of *LNCS*, pp. 41–62, Springer-Verlag, 2001.

[13] D. Halevy and A. Shamir, "The LSD broadcast encryption scheme," in *CRYPTO'02*, vol. 2442 of *LNCS*, (London, UK), pp. 47–60, Springer-Verlag, 2002.

[14] M. T. Goodrich, J. Z. Sun, and R. Tamassia, "Efficient tree based revocation in groups of low-state devices," in *CRYPTO'04*, vol. 3152 of *LNCS*, pp. 511–527, Springer-Verlag, 2004.

[15] J. Horwitz, "A survey of broadcast encryption," 2003. Manuscript.

[16] "AACS - Advanced Access Content System," 2007. http://www.aacsla.com.

[17] J. H. Cheon, N.-S. Jho, M.-H. Kim, and E. S. Yoo, "Skipping, cascade, and combined chain schemes for broadcast encryption," *IEEE Transactions on Information Theory*, vol. 54, no. 11, pp. 5155–5171, 2008.

[18] N.-S. Jho, E. S. Yoo, J. H. Cheon, and M.-H. Kim, "New broadcast encryption scheme using tree-based circle," in *Proceedings of the 5th ACM workshop on Digital Rights Management*, DRM '05, pp. 37–44, 2005.

[19] N.-S. Jho, J. Y. Hwang, J. H. Cheon, M.-H. Kim, D. H. Lee, and E. S. Yoo, "One-way chain based broadcast encryption schemes," in *EUROCRYPT*, vol. 3494 of *LNCS*, pp. 559–574, Springer-Verlag, 2005.

[20] N.-S. Jho, J. H. Cheon, M.-H. Kim, and E. S. Yoo, "Broadcast encryption π." Cryptology ePrint Archive, Report 2005/073, 2005. http://eprint.iacr.org/.

[21] D. Boneh, C. Gentry, and B. Waters, "Collusion resistent broadcast encryption with shorter ciphertexts and private keys," in *CRYPTO'05*, vol. 3621 of *LNCS*, pp. 258–275, Springer-Verlag, 2005.

[22] D. Boneh and M. Hamburg, "Generalized identity based and broadcast encryption schemes," in *ASIACRYPT'08*, vol. 5350 of *LNCS*, pp. 455–470, Springer-Berlin, 2008.

[23] V. Daza, J. Herranz, P. Morillo, and C. Ráfols, "Ad-hoc threshold broadcast encryption with shorter ciphertexts," *Electronic Notes in Theoretical Computer Science*, vol. 192, no. 2, pp. 3–15, 2008.

[24] C. Delerableé and D. Pointcheval, "Dynamic threshold public key broadcast encryption," in *CRYPTO'08*, vol. 5157 of *LNCS*, pp. 317–334, Springer-Verlag, 2008.

[25] M. Kusakawa, H. Hiwatari, T. Asano, and S. Matsuda, "Efficient dynamic broadcast encryption and its extension to authenticated dynamic broadcast encryption," in *CANS'08*, vol. 5339 of *LNCS*, pp. 31–48, Springer-Verlag, 2008.

[26] Y. R. Liu and W. G. .Tzeng, "Public key broadcast encryption with low number of keys and constant decryption time," in *PKC'08*, vol. 4939 of *LNCS*, pp. 380–396, Springer-Verlag, 2008.

[27] J. H. Park, H. J. Kim, M. H. Sung, and D. H. Lee, "Public key broadcast encryption schemes with shorter transmissions," *IEEE Transactions on Broadcasting*, vol. 54, no. 3, pp. 401–411, 2008.

[28] R. Kosala and H. Blockeel, "Web mining research: A survey," *ACM SIGKDD Explorations*, vol. 2, 2000.

[29] E. David and S. Kraus, "Agents for information broadcasting," in *ATAL'99: 6th International Workshop on Intelligent Agents VI, Agent Theories, Architectures, and Languages (ATAL),*, (London, UK), pp. 91–105, Springer-Verlag, 2000.

[30] M. Kim, S. Kang, M. Kim, and J. Kim, "Target advertisement service using TV viewers' profile inference," in *Advances in Mulitmedia Information Processing - PCM 2005*, (Berlin, Germany), pp. 202–211, Springer, 2005.

[31] J. Lim, M. Kim, B. Lee, M. Kim, H. Lee, and H. Lee, "A target advertisement system based on TV viewer's profile reasoning," *Multimedia Tools and Applications*, vol. 2, pp. 11–35, 2007.

[32] B. Chor, A. Fiat, and M. Naor, "Tracing traitors," in *Advances in Cryptology – CRYPTO '94* (Y. Desmedt, ed.), vol. 839 of *Lecture Notes in Computer Science*, pp. 257–270, Springer Berlin Heidelberg, 1994.

[33] D. Boneh and M. Franklin, "An efficient public key traitor tracing scheme," in *Advances in Cryptology — CRYPTO' 99* (M. Wiener, ed.), vol. 1666 of *Lecture Notes in Computer Science*, pp. 338–353, Springer Berlin Heidelberg, 1999.

[34] K. Kurosawa and Y. Desmedt, "Optimum traitor tracing and asymmetric schemes," in *Advances in Cryptology — EUROCRYPT'98* (K. Nyberg, ed.), vol. 1403 of *Lecture Notes in Computer Science*, pp. 145–157, Springer Berlin Heidelberg, 1998.

[35] M. Abdalla, A. W. Dent, J. Malone-Lee, G. Neven, D. H. Phan, and N. P. Smart, "Identity-based traitor tracing," in *Proceedings of the 10th international conference on Practice and theory in public-key cryptography*, PKC'07, (Berlin, Heidelberg), pp. 361–376, Springer-Verlag, 2007.

[36] D. Boneh and M. Naor, "Traitor tracing with constant size ciphertext," in *Proceedings of the 15th ACM conference on Computer and communications security*, CCS '08, (New York, NY, USA), pp. 501–510, ACM, 2008.

[37] D. Boneh, A. Sahai, and B. Waters, "Fully collusion resistant traitor tracing with short ciphertexts and private keys," in *Proceedings of the 24th annual international conference on The Theory and Applications of Cryptographic Techniques*, EUROCRYPT'06, (Berlin, Heidelberg), pp. 573–592, Springer-Verlag, 2006.

[38] N. Fazio, A. Nicolosi, and D. H. Phan, "Traitor tracing with optimal transmission rate," in *Proceedings of the 10th international conference on Information Security*, ISC'07, (Berlin, Heidelberg), pp. 71–88, Springer-Verlag, 2007.

[39] A. Kiayias and M. Yung, "On crafty pirates and foxy tracers," in *Revised Papers from the ACM CCS-8 Workshop on Security and Privacy in Digital Rights Management*, DRM '01, (London, UK, UK), pp. 22–39, Springer-Verlag, 2002.

[40] A. Kiayias and S. Pehlivanoglu, "Improving the round complexity of traitor tracing schemes," in *Proceedings of the 8th international conference on Applied cryptography and network security*, ACNS'10, (Berlin, Heidelberg), pp. 273–290, Springer-Verlag, 2010.

[41] T. Matsushita and H. Imai, "A public-key black-box traitor tracing scheme with sublinear ciphertext size against self-defensive pirates," in *Advances in Cryptology - ASIACRYPT 2004* (P. Lee, ed.), vol. 3329 of *Lecture Notes in Computer Science*, pp. 260–275, Springer Berlin Heidelberg, 2004.

[42] M. Naor and B. Pinkas, "Efficient trace and revoke schemes," in *Proceedings of the 4th International Conference on Financial Cryptography*, FC '00, (London, UK, UK), pp. 1–20, Springer-Verlag, 2001.

[43] D. Boneh and B. Waters, "A fully collusion resistant broadcast, trace, and revoke system," in *Proceedings of the 13th ACM conference on Computer and communications security*, CCS '06, (New York, NY, USA), pp. 211–220, ACM, 2006.

[44] A. Kiayias and S. Pehlivanoglu, "Tracing and revoking pirate rebroadcasts," in *Proceedings of the 7th International Conference on Applied Cryptography and Network Security*, ACNS '09, (Berlin, Heidelberg), pp. 253–271, Springer-Verlag, 2009.

[45] A. Kiayias and S. Pehlivanoglu, "Pirate evolution: how to make the most of your traitor keys," in *Proceedings of the 27th annual international cryptology conference on Advances in cryptology*, CRYPTO'07, (Berlin, Heidelberg), pp. 448–465, Springer-Verlag, 2007.

[46] J. Furukawa and N. Attrapadung, "Fully collusion resistant black-box traitor revocable broadcast encryption with short private keys," in *Automata, Languages and Programming* (L. Arge, C. Cachin, T. Jurdziński, and A. Tarlecki, eds.), vol. 4596 of *Lecture Notes in Computer Science*, pp. 496–508, Springer Berlin Heidelberg, 2007.

[47] M. Luby and J. Staddon, "Combinatorial bounds for broadcast encryption," in *EUROCRYPT'93*, vol. 1403 of *LNCS*, pp. 512–526, Springer-Verlag, 1998.

[48] M. Ak, K. Kaya, and A. A. Selçuk, "Optimal subset-difference broadcast encryption with free riders," *Information Sciences*, vol. 179, no. 20, pp. 3673 – 3684, 2009.

[49] Z. Ramzan and D. Woodruff, "Fast algorithms for the free riders problem in broadcast encryption," in *CRYPTO'06*, vol. 4117 of *LNCS*, pp. 308–325, Springer-Verlag, 2006.

[50] E. Dees, "Decentralized advertisement recommendation on IPTV." Vrije Universiteit Amsterdam, July 2007.

[51] O. Nasraoui, "World wide web personalization." Invited chapter in "Encyclopedia of Data Mining and Data Warehousing", J. Wang, Ed, Idea Group, 2005.

[52] A. A. Selçuk and D. Sidhu, "Probabilistic optimization techniques for multicast key management," *Computer Networks*, vol. 40, no. 2, pp. 219–234, 2002.

[53] P. D'Arco and A. D. Santis, "Optimizing SD and LSD in presence of non-uniform probabilities of revocation.," in *Proc. of International Conference on Information Theoretic Security (ICITS)*, 2007.

[54] D. Huffman, "A method for the construction of minimum redundancy codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, 1952.

[55] M. Ak, K. Kaya, and A. A. Selçuk, "Optimal subset-difference broadcast encryption with free riders," *Information Sciences*, vol. 179, pp. 3673–3684, September 2009.

[56] A. Fiat and T. Tassa, "Dynamic traitor tracing," *Journal of Cryptology*, vol. 14, pp. 211–223, 2001.

[57] R. Safavi-Naini and Y. Wang, "Sequential traitor tracing," in *Proceedings of the 20th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '00, (London, UK, UK), pp. 316–332, Springer-Verlag, 2000.

[58] H. Jin and J. Lotspiech, "Renewable traitor tracing: A trace-revoke-trace system for anonymous attack," in *ESORICS'07*, pp. 563–577, 2007.

[59] H. Chabanne, D. H. Phan, and D. Pointcheval, "Public traceability in traitor tracing schemes," in *Proceedings of the 24th annual international conference on Theory and Applications of Cryptographic Techniques*, EUROCRYPT'05, (Berlin, Heidelberg), pp. 542–558, Springer-Verlag, 2005.

[60] D. H. Phan, R. Safavi-Naini, and D. Tonien, "Generic construction of hybrid public key traitor tracing with full-public-traceability," in *Proceedings of the 33rd international conference on Automata, Languages and*

Programming - Volume Part II, ICALP'06, (Berlin, Heidelberg), pp. 264–275, Springer-Verlag, 2006.

[61] A. Kiayias and S. Pehlivanoglu, "On the security of a public-key traitor tracing scheme with sublinear ciphertext size," in *Proceedings of the nineth ACM workshop on Digital rights management*, DRM '09, (New York, NY, USA), pp. 1–10, ACM, 2009.

[62] M. Lee, D. Ma, and M. Seo, "Breaking two k-resilient traitor tracing schemes with sublinear ciphertext size," in *Proceedings of the 7th International Conference on Applied Cryptography and Network Security*, ACNS '09, (Berlin, Heidelberg), pp. 238–252, Springer-Verlag, 2009.

[63] D. Boneh and J. Shaw, "Collusion-secure fingerprinting for digital data," *IEEE Trans. Inf. Theor.*, vol. 44, pp. 1897–1905, Sept. 2006.

[64] G. Tardos, "Optimal probabilistic fingerprint codes," in *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, STOC '03, (New York, NY, USA), pp. 116–125, ACM, 2003.

[65] C. Delerablée, "Identity-based broadcast encryption with constant size ciphertexts and private keys," in *Proceedings of the Advances in Crypotology 13th international conference on Theory and application of cryptology and information security*, ASIACRYPT'07, (Berlin, Heidelberg), pp. 200–215, Springer-Verlag, 2007.

[66] C. Gentry and B. Waters, "Adaptive security in broadcast encryption systems (with short ciphertexts)," in *Proceedings of the 28th Annual International Conference on Advances in Cryptology: the Theory and Applications of Cryptographic Techniques*, EUROCRYPT '09, (Berlin, Heidelberg), pp. 171–188, Springer-Verlag, 2009.

[67] A. W. Dent, "Adapting the weaknesses of the random oracle model to the generic group model," in *Proceedings of the 8th International Conference on the Theory and Application of Cryptology and Information Security:*

Advances in Cryptology, ASIACRYPT '02, (London, UK, UK), pp. 100–109, Springer-Verlag, 2002.

[68] Y. Dodis and N. Fazio, "Public key trace and revoke scheme secure against adaptive chosen ciphertext attack," in *Proceedings of the 6th International Workshop on Theory and Practice in Public Key Cryptography: Public Key Cryptography*, PKC '03, (London, UK, UK), pp. 100–115, Springer-Verlag, 2003.

[69] D. H. Phan and V. C. Trinh, "Identity-based trace and revoke schemes," in *Proceedings of the 5th international conference on Provable security*, ProvSec'11, (Berlin, Heidelberg), pp. 204–221, Springer-Verlag, 2011.

[70] R. Cramer and V. Shoup, "A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack," in *Proceedings of the 18th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '98, (London, UK, UK), pp. 13–25, Springer-Verlag, 1998.

[71] D. Boneh, A. Kiayias, and H. W. Montgomery, "Robust fingerprinting codes: a near optimal construction," in *Proceedings of the tenth annual ACM workshop on Digital rights management*, DRM '10, (New York, NY, USA), pp. 3–12, ACM, 2010.

[72] R. Sakai and J. Furukawa, "Identity-based broadcast encryption." Cryptology ePrint Archive, Report 2007/217, 2007. http://eprint.iacr.org/.

[73] A. Lewko, A. Sahai, and B. Waters, "Revocation systems with very small private keys," in *Proceedings of the 2010 IEEE Symposium on Security and Privacy*, SP '10, (Washington, DC, USA), pp. 273–285, IEEE Computer Society, 2010.

[74] C. Delerablée, P. Paillier, and D. Pointcheval, "Fully collusion secure dynamic broadcast encryption with constant-size ciphertexts or decryption keys," in *Pairing'07*, pp. 39–59, 2007.

[75] A. Barth, D. Boneh, and B. Waters, "Privacy in encrypted content distribution using private broadcast encryption," in *Proceedings of the 10th international conference on Financial Cryptography and Data Security*, FC'06, (Berlin, Heidelberg), pp. 52–64, Springer-Verlag, 2006.

[76] N. Fazio and I. M. Perera, "Outsider-anonymous broadcast encryption with sublinear ciphertexts," in *Proceedings of the 15th international conference on Practice and Theory in Public Key Cryptography*, PKC'12, (Berlin, Heidelberg), pp. 225–242, Springer-Verlag, 2012.

[77] B. Libert, K. G. Paterson, and E. A. Quaglia, "Anonymous broadcast encryption: adaptive security and efficient constructions in the standard model," in *Proceedings of the 15th international conference on Practice and Theory in Public Key Cryptography*, PKC'12, (Berlin, Heidelberg), pp. 206–224, Springer-Verlag, 2012.