Fighting Pirates 2.0

Paolo D'Arco[1] and Angel L. Perez del Pozo[2]

[1] Dipartimento di Informatica
Università degli Studi di Salerno, 84084, Fisciano (SA), Italy
paodar@dia.unisa.it
[2] Universidad Rey Juan Carlos, 28933, Móstoles, Madrid, Spain
angel.perez@urjc.es

Abstract. In this paper we propose methods to cope with the Pirates 2.0 attack strategy against tracing and revoking schemes presented at Eurocrypt 2009. In the Pirates 2.0 attack model traitors collaborate in *public* and *partially* share their secret information with a *certified* guarantee of anonymity. Several classes of tracing and revoking schemes are subject to such a new threat. We focus our attention on the tree-based class of schemes. We start by discussing some simple techniques which can partially help to deal with the attack, and point out their limits. Then, we describe a new hybrid scheme which can be used to face up the Pirates 2.0 attack strategy.

Keywords: Broadcast encryption, user revocation, traitor tracing.

1 Introduction

Secure Content Distribution. A central research issue within the cryptographic community, starting from the 90's, has been providing methods to enable a center to deliver encrypted data to a large set of users, in such a way that only a privileged subset of them can decrypt the data. Applications for these schemes range from pay-tv systems to systems for delivering sensitive information stored on media like CDs, DVDs or even available through web-based services.

The research efforts have been basically directed toward solving two problems: the *access* problem, i.e., privileged users decrypt the content, unauthorized do not; the *tracing* problem, i.e., the ability to trace and, hence, discourage dishonest users, to illegally help unauthorized users to set up a decoder to gain access to the content. Specifically, in a *broadcast encryption* scheme (and its variants), during a set-up phase, every user receives a set of predefined keys. Then, at the beginning of each data transmission, the center sends a broadcast message enabling privileged users to compute a session key, by means of which, the encrypted data, that will be delivered later on, can be decrypted. The most challenging, useful and well-studied setting is the one were the users are equipped with *stateless decoders*. In such a setting *a)* from time to time, the subset of privileged users *changes*, but *b)* the sets of predefined keys held by the users *stay the same* for the lifetime of the scheme. A broadcast encryption scheme is required

J. Lopez and G. Tsudik (Eds.): ACNS 2011, LNCS 6715, pp. 359–376, 2011.

to be *collusion resistant*, in the sense that even if all the revoked users share their secret information, they cannot compute a session key they are not entitled to.

On the other hand, a *traitor tracing* scheme is designed to get the same functionality but, the key material embedded in the decoders (i.e., given to the users) in the set up phase, is diversified on *a user basis*, in such a way that when a pirate decoder, built through the contribution of legal users which collude, is found, at least the identity of one of them can be caught.

At the state of current knowledge, efficient broadcast encryption schemes, in terms of user storage and bandwidth (i.e., size of the broadcast message sent) requirements, are known (e.g. [25,19,15,3,17,12]). Similarly, tracing schemes very efficient in terms of bandwidth requirements have been proposed in the literature (e.g., [4,6]). Unfortunately, such tracing schemes are not so efficient in terms of user memory storage, when the number of traitors grows. Moreover, several schemes proposed in the literature exhibit both functionalities, e.g., some broadcast encryption schemes (and variants) support efficient tracing procedures.

A look back. Berkovits, in [2], addressed for the first time the issue of how to broadcast a secret to privileged users. Later on, Fiat and Naor, in [14], formalized the *broadcast encryption paradigm*. Since then, it has become a major topic in Cryptography, due to the large number of possible applications, and it has evolved in several directions (e.g., *multicasting schemes* [9], *traitor tracing schemes* [8], *revoking schemes*, also referred to as *user exclusion* schemes or *blacklisting* schemes, [24,23,1]).

Among these, *tracing and revoking schemes*, are schemes which combine tracing and revocation. [26] started studying efficient schemes exhibiting both functionalities. A seminal paper along this line is [25], where a framework for stateless tracing and revoking schemes enabling efficient user revocation, referred to as *Subset-Cover*, was proposed. The most efficient construction described in [25], the SD scheme, was later enhanced in [19], which proposed the LSD scheme. Related works are [11,15,18,20,17,22].

The Pirates 2.0 attack model. Historically, traitors have been considered users who *privately* share their secret information to enable unauthorized users to gain access to protected contents. In the Pirates 2.0 attack model [7], traitors collaborate in *public* and *partially* share their secret information with a *certified* guarantee of anonymity. It has been shown that several classes of tracing and revoking schemes, like tree-based revocation schemes (e.g., CS, SD, LSD...) and code-based tracing schemes (e.g., tracing schemes based on collusion secure codes, IPP codes...) are subject to such a new threat.

Our Contribution. We propose methods to cope with the new Pirates 2.0 attack strategy against tree-based tracing and revoking schemes. First, we discuss some simple techniques which can partially help to deal with the attack, and point out their limits. Then, looking through the literature, we recover some ideas, which can be used to strengthen revocation and tracing schemes. We analyze the trade-off that can be obtained by applying these ideas to the schemes. Finally, we describe a new hybrid scheme, obtained by mixing two previous constructions, which can be used to face up the Pirates 2.0 attack strategy.

2 Pirates 2.0: A New Attack Scenario

Let us briefly recall the attack model introduced in [7].

Basic Features. The main characteristics of Pirates 2.0 attacks are the following:

- *Anonymity Guarantee.* Traitors that participate are provided with guarantee, through the exhibition of a mathematical proof, that they cannot be traced by the authority.
- *Partial Contribution.* They never need to reveal their whole set of secret keys.
- *Public Collusion.* Traitors operate in a public environment: they publish secret data from their decoders.
- *Large Coalitions.* They take part in unusual large coalitions.
- *Dynamic Coalitions.* Traitors can come into action only when necessary.
- *Imperfect Decoders.* The pirate decoders are useful even if they decrypt only a certain percentage of ciphertexts.

The motivation for a Pirates 2.0 attack might be for example the need to get rid of a protection system to which a large number of users are hostile. In such an attack model the traitors contribute information at their discretion, which can be collected through a centralized system or a distributed system.

Setting. The N users in the system belong to three different groups: honest users, traitors, and pirates. *Honest users* are legitimate users who keep secret their secret information. *Traitors* are legitimate but dishonest users who (partially) disclose their secret information. *Pirates* are not legitimate users, not entitled to secret information, but able to collect relevant secret information from the public environment, in order to produce a pirate decoder.

The attack set up by traitors is modeled through the following concepts: the *contributed information*, denoted with $\mathcal{C}$, represents the sum of information released to the public domain by the traitors. Initially $\mathcal{C} = \emptyset$. Traitors, to provide information to pirates, implement and run a public available probabilistic algorithm, $Contribute()$. Such an algorithm takes as input the secret information sk of the traitor, information already published by traitors, denoted by I, and the history H of the traitor's contribution to the public. When $Contribute(sk, I, H)$ is executed, the contributed information $\mathcal{C}$ becomes $\mathcal{C} = \mathcal{C} \cup Contribute(sk, I, H)$. Moreover, the term *Public Information*, denoted by $\mathcal{P}$, refers to all public data available from the broadcaster, along with the contributed information. The *anonymity level* of a traitor is quantified through a publicly available procedure, $Anonymity()$. It takes as input the secret information sk of the traitor, information the traitor released, denoted with S, and the public information $\mathcal{P}$. $Anonymity(sk, S, \mathcal{P})$ outputs an integer $\ell \in \{1, \ldots, N\}$. The value $\ell = 1$ means the traitor is known, while the value $\ell = N$ means the traitor is indistinguishable from any other user.

Finally, a *pirate decoder* is the output of an algorithm, $Pirate()$. Such an algorithm takes as input $\mathcal{P}$ and, if the amount of public information is enough, it produces a decrypting device (pirate decoder). Otherwise, it outputs 'failed'.

Definition 1. *A traitor tracing scheme is α-secure against the Pirates 2.0 attack if it prevents the construction of pirate decoders from information published by traitors with an anonymity level greater than α.*

Anonymity Treatment. Traitors are free to contribute some pieces of secret data as long as plenty of users of the system *could have contributed* exactly the same information following the same public strategy. More formally, user anonymity is defined and estimated as follows.

Definition 2. Extraction function. *An extraction function is an efficiently computable function f that outputs information about the secret key.*

Definition 3. Masked traitor. *A traitor t is said to be masked by a user u for an extraction function f if $f(sk_t) = f(sk_u)$.*

Definition 4. Anonymity level. *The level of anonymity of a traitor t after a contribution $\cup_{1\leq i\leq n} f_i(sk_t)$ is defined as the number α of users masking t for each of the n extraction functions f_i simultaneously:*

$$\alpha = \#\{u | \forall i, f_i(sk_t) = f_i(sk_u)\}.$$

In the following the class of extraction functions considered are projection functions, i.e., the secret information is a vector of secret keys, and f_i is a projection function, which returns the i-th coordinate (key) of the vector. Hence, projection functions model partial public release of key-material.

3 The Subset-Cover Framework: CS, SD, and LSD

In a seminal paper [25] a framework for tracing and revoking schemes enabling efficient user revocation[1], referred to as *Subset-Cover*, was proposed. We briefly recall it here and refer the reader to [25] for details.

Let $\mathcal{N}$ be the set of all users. Every user $u \in \mathcal{N}$ gets, at the beginning, some secret information I_u. Let $\mathcal{R} \subset \mathcal{N}$ be a subset of r users which should be revoked. The center sends a message M containing a new session key K such that all users $u \in \mathcal{N} \setminus \mathcal{R}$ can decrypt the message correctly, while even a coalition consisting of all members of $\mathcal{R}$ cannot decrypt it.

An algorithm defines a collection of subsets $S_1, \ldots, S_\omega \subseteq \mathcal{N}$. Each subset S_j is assigned (perhaps implicitly) a key L_j; each member $u \in S_j$ can compute L_j from its secret information I_u. Given a set of revoked users, the remaining users $\mathcal{N} \setminus \mathcal{R}$ are partitioned into disjoint subsets $S_{i_1}, \ldots, S_{i_m}$ so that $\mathcal{N} \setminus \mathcal{R} = \bigcup_{j=1}^{m} S_{i_j}$ and the session key K is encrypted m times with $L_{i_1}, \ldots, L_{i_m}$, i.e., denoting by E and F two different encryption functions (see [25] for a discussion about the properties of the functions), the broadcast message has the following structure:

$$[i_1, \ldots, i_m, E_{L_{i_1}}(K), \ldots, E_{L_{i_m}}(K), F_K(M)].$$

[1] The join operation is trivially performed by constructing at the beginning an oversized tree structure, capable of accommodating new users. Hence, the focus is only posed on efficient ways to revoke users.

A particular implementation of such a scheme is specified by *a)* the collection of subsets $S_1, \ldots, S_\omega$, *b)* the key assignment to each subset in the collection, *c)* a method to cover the non-revoked users $\mathcal{N} \setminus \mathcal{R}$ by disjoint subsets from this collection, and *d)* a method that allows each user u to find its cover-set S and compute its key L_S from I_u.

The schemes of [25] we consider in the following are all based on trees, i.e., receivers are associated to the leaves of a full binary tree.

Complete Subtree Method (CS, for short). The collection of subsets $\{S_1, \ldots, S_\omega\}$ in the CS scheme corresponds to all subtrees in the full binary tree with n leaves. For any node v_i in the full binary tree (either an internal node or a leaf, $2n-1$ altogether) let the subset S_i be the collection of receivers u that correspond to the leaves of the subtree rooted at node v_i. In other words, $u \in S_i$ iff v_i is an ancestor of u. The key assignment method is simple: assign an independent and random key L_i to every node v_i in the complete tree. Provide every receiver u with the $\log n + 1$ keys associated with nodes along the path from the root to leaf u. For a given set $\mathcal{R}$ of revoked receivers, let $\{u_1, \ldots, u_r\}$ be the leaves corresponding to the elements in $\mathcal{R}$. The method to cover $\mathcal{N} \setminus \mathcal{R}$ with disjoint subsets essentially consists in selecting the minimal number of subtrees which do not contain any of $\{u_1, \ldots, u_r\}$ (see [25] for details). It can be shown that the size of the broadcast message is $O(r \log \frac{N}{r})$.

Subset Difference Method (SD, for short). The collection of subsets in the SD scheme corresponds to subsets of the form "a group of receivers G_1 minus another group G_2." More precisely, a valid subset $S_{i,j}$ is represented by two vertices (v_i, v_j) such that v_i is an ancestor of v_j. A leaf u is in $S_{i,j}$ iff it is in the subtree rooted at v_i but not in the subtree rooted at v_j (see Figure 1).

The key assignment method associates to each *internal node* v_i of the full binary tree a random and independent value $LABEL_i$ (kept secret). This value induces the keys for all legitimate subsets of the form $S_{i,j}$. More precisely, let G be a cryptographic pseudorandom generator that triples the input, and denote

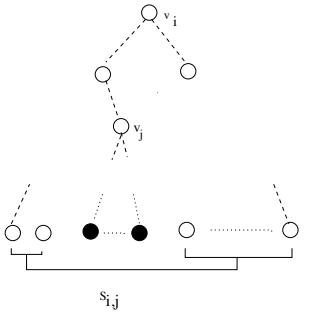

Fig. 1. $S_{i,j}$: set of leaves descending from v_i but not from v_j

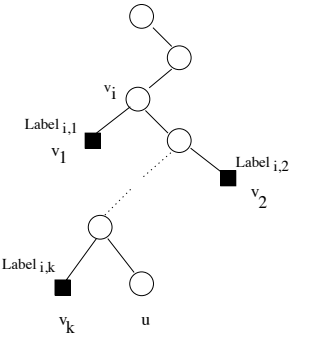

Fig. 2. Labels of nodes hanging off the path from v_i to user u

by $G_L(S), G_R(S)$, and $G_M(S)$ the left, right and middle parts, respectively. Consider a subtree T_i rooted at v_i. To the root node is assigned $LABEL_i$. From this label are computed recursively labels for all subtrees associated to subsets $S_{i,j}$: given that a parent node was labeled S, its two children are labeled $G_L(S)$ and $G_R(S)$. Let $LABEL_{i,j}$ be the label of node v_j derived in the subtree T_i from $LABEL_i$. Following such a labeling, the key $L_{i,j}$ assigned to $S_{i,j}$ is $G_M(LABEL_{i,j})$. The information I_u that each receiver u gets consists in a set of values $LABEL_{i,j}$ defined as follows: *for each* subtree T_i such that u is a leaf of T_i, consider the path from v_i to u and let $v_1, \ldots, v_k$ be the nodes adjacent to the path but not ancestor of u (see Figure 2). Then, u receives the labels $LABEL_{i,1}, \ldots, LABEL_{i,k}$. Notice that each node v_j that is not an ancestor of u is a descendant of one of the nodes $v_1, \ldots, v_k$. Therefore, u can compute the labels $LABEL_{i,j}$ for *any* j such that v_j is not an ancestor of u. It is possible to show that in SD each user stores $O((\log n)^2)$ labels and the size of the broadcast message is $O(r)$ (see [25] for details and how to construct the covering).

Layered Subset Difference Method (LSD, for short). The LSD scheme [19] shows that a small subcollection of subsets $S_{i,j}$ from the collection used by SD suffices to represent any set P as the union of $O(r)$ of the remaining subsets, with a slightly larger constant hidden within the asymptotic notation.

Security. The security of the CS scheme and of the SD scheme was shown in [25]. CS is unconditionally secure w.r.t. coalitions of users of any size (see Claim 1 of [25]). SD is computationally secure, according to an indistinguishability definition, and based on some assumptions on the encryption scheme and the generator used in the scheme (see Theorem 16 of [25]). The security of LSD follows from the security of SD.

Pirates 2.0 **attack.** CS, SD and LSD are subject to the new attack. All the traitors do is to publish the keys (labels) associated to nodes of the *first levels* of the tree i.e., up in the tree. The idea is that keys up in the tree are stored by many users (all leaves of the corresponding subtrees) who could have published them.

More precisely, in [7], the following theorem was proved:

Theorem 1. *On average, a randomly chosen group of* $\rho \log \rho$ *(operating isolated) users is able to mount a Pirates* 2.0 *attack against a complete subtree scheme in which the center wants to ensure a ciphertext rate of at most* $\frac{\rho(N-r)}{N}$. *Moreover, each traitor is guaranteed an anonymity level of* N/ρ.

In other words, traitors, in CS, by publishing keys associated to nodes belonging to levels λ such that $\lambda \leq \lfloor \log \rho \rfloor$, are guaranteed an anonymity level of N/ρ. Hence, the center, to avoid the attack has to use keys at lower levels. Unfortunately, this means that the broadcast message size from $O(r \log (N/r))$ moves to $O(\frac{\rho(N-r)}{N} + r \log (N/r))$. Similarly, in SD, by publishing direct labels (i.e., labels $LABEL_{i,j}$ were j is either the left child or the right child of i) associated to nodes belonging to levels λ such that $\lambda \leq \lfloor \log \frac{\rho}{2} \rfloor$, traitors are guaranteed an anonymity level of $N/2\rho$.

4 Partial Measures against Pirates 2.0

In this section we discuss some measures which can partially cope with the Pirates 2.0 attack. The key observation is that the attack is successful because in CS, SD and LSD, the users *know the levels* of their keys. Hence, a first attempt to reduce the impact of the attack, could be to look for a strategy which *reduces such a knowledge* but, at the same time, keeps the functionality of the scheme. The measures we discuss are intended for the CS scheme.

Permuting labels. Let us focus on the CS scheme. Let $\mathcal{V} = \{v_i\}_{i \in I}$ be the set of nodes of the complete tree and $\mathcal{W} = \{w_i\}_{i \in I}$ be a set of labels (w.l.o.g., we can assume, for example, that $\mathcal{W} = \{1, \ldots, |I|\}$). We modify the CS scheme (see Table 1) by associating *random labels to nodes*. The labels do not provide any information about the levels of the nodes.

Table 1. Permutation-based Construction

Permutation-based Construction.

Setup phase. The Broadcasting Center

- Chooses uniformly at random a *secret* bijection $\pi : \mathcal{V} \longrightarrow \mathcal{W}$.
- Associates to each node v_i a key K_i, chosen u.a.r.
- For each complete subtree S_i rooted in v_i, sends to each user corresponding to a leaf of S_i, the pair $(\pi(v_i), K_i)$.

Broadcast phase. At the beginning of a new session, to send message M, the broadcasting center

- Computes a cover $\{S_{i_1}, \ldots S_{i_m}\}$ for non-revoked users in $\mathcal{N} \setminus \mathcal{R}$.
- Chooses u.a.r a session key K, computes $F_K(M)$ and, for each subtree S_i in the cover, computes $E_{K_i}(K)$.
- Sends
$$[(\pi(v_{i_1}), E_{K_{i_1}}(K)), \ldots, (\pi(v_{i_m}), E_{K_{i_m}}(K)), F_K(M)].$$

Decryption phase. Upon receiving a broadcast message, a non-revoked user u

- Looks for a matching label contained both in the header of the message and in his set of labeled long-term keys. Let w_l be such a label.
- Computes $K = D_{K_{w_l}}(E_{K_{w_l}}(K))$ and $M = F_K^{-1}(F_K(M))$.

The purpose of this modification is to make more difficult a Pirates 2.0 type attack. In the attack proposed in [7] a traitor publishes all his keys corresponding to complete subtrees rooted at a level of the complete tree lower than a certain threshold λ. In this way, a traitor is guaranteed an anonymity level of $N/2^{\lambda}$, where N is the total number of users. The proposed modification thwarts in some way

this attack, as a user *does not* have a priori information about the correspondence between the keys in his possession and their levels in the complete tree.

Unfortunately, notice that users could gain some information about this correspondence after seeing and decrypting broadcasted messages. Just to exemplify, pirates could get information about the level of their keys by *publicly collaborating*. For example, the users could use a shared table, in which in the first column are reported the labels $\pi(v_i)$, associated to the keys. In the second, each user puts a cross in correspondence of the label associated to the key he used to decrypt. In such a way, they can estimate the level of the key associated to a certain label. However, the important point is that pirates *lose the anonymity guarantee* they get in CS (no certificate is available any more). They can get a partial guarantee but they need *to trust* the other pirates and they have to assume that nobody else adds crosses in the table to falsify the results.

Adding some randomness. Another approach which seems to be of some help consists in looking for a strategy which *reduces the level of anonymity* of the user. To this aim, we modify the CS scheme (see Table 2), following an idea from [16], where the family of "OR"-protocols was introduced. If at each node are associated more keys and a user which descends from the node gets one of the keys chosen *uniformly at random*, then the level of anonymity depends on the level of the keys but also on the number of keys associated to that node. Of course there is a trade-off: if the node is used, the broadcast message has to be encrypted with all keys associated to the node.

Let $h = max\{h_\lambda : \lambda = 0, \ldots, \log N\}$. The parameters of the modified scheme (compared to CS) are:

	CS	modified CS
# keys per user	$\log N + 1$	$\log N + 1$
total # of keys	$2N - 1$	$\sum_{\lambda=0}^{\log N} 2^\lambda h_\lambda \leq h(2N-1)$
header length	$O(r\log(N/r))$	$O(hr\log(N/r))$

The modifications are useful to fight against Pirates 2.0 type attacks. In the modified scheme, a traitor following the strategy of making public his unique key with label equal to a certain level λ is covered, on average, by $N/2^\lambda h_\lambda$ users. Moreover, a traitor following the strategy of making public *all* his keys with label lower than a certain level λ is covered, on average, by $\frac{N}{2^\lambda}\prod_{l=0}^{\lambda}\left(\frac{1}{h_l}\right)$ users.

Notice that an interesting variant of the scheme could be the following: for each node, instead of associating to users of the subtree a key of the pool chosen *uniformly at random*, the keys could be associated according to a *non-uniform* probability distribution (kept secret by the center) e.g., a degenerate-like probability distribution which associates a sort of *trap* key to a single or few users. In such a case, if the user publishes such a key, the level of anonymity is much lower than he is expecting. Of course, also this measure is useless if the traitor try to estimate how many of them have decrypted a broadcast message by using a certain key. But, again, traitors need to trust each other and no anonymity guarantee is available anymore.

Table 2. Or-based Construction

Or-based Construction.

Setup phase. W.l.o.g., assume that $\log N$ is an integer value.

1. For each $\lambda \in \{0, \ldots, \log N\}$ choose an integer $h_\lambda > 0$.
2. For each subtree S_i, rooted at node v_i at level λ in the tree, generate a set
$$\mathcal{K}_i = \{K_{i,1}, \ldots, K_{i,h_\lambda}\}$$
of h_λ keys.
3. For each subtree S_i, send to each user covered by S_i a single pair $((i,j), K_{i,j})$, where $K_{i,j}$ is chosen u.a.r. from $\mathcal{K}_i$, independently for each user.

Broadcast phase. At the beginning of a new session, to send message M, the broadcasting center

- Computes a cover $\{S_{i_1}, \ldots S_{i_m}\}$ for non-revoked users in $\mathcal{N} \setminus \mathcal{R}$.
- Chooses u.a.r a session key K, computes $F_K(M)$ and, for each subtree S_i in the cover, computes $E_{K_{i,1}}(K), \ldots, E_{K_{i,h_\lambda}}(K)$.
- Sends
$$[i_1, E_{K_{i_1,1}}(K), \ldots, E_{K_{i_1,h_\lambda}}(K), \ldots, i_m, E_{K_{i_m,1}}(K), \ldots, E_{K_{i_m,h_\lambda}}(K), F_K(M)].$$

Decryption phase. Upon receiving a broadcast message, a non-revoked user u

- Identifies the subtree S_i he belongs to, and looks up the key $K_{i,j}$ he holds for S_i
- Computes $K = D_{K_{i,j}}(E_{K_{i,j}}(K))$ and $M = F_K^{-1}(F_K(M))$.

5 A New Scheme

All measures described before are partial solutions against the attack. The problem is that the secret information is not bound to the user identity. We need a method through which, if a user publishes part of his secret information, then he is immediately compromised. Of course, we would like to keep the scheme as efficient as possible, comparable to the version which does not deal with the Pirates 2.0 attack. To this aim, we combine two tracing and revoking schemes previously proposed in the literature. The first one [26] in a certain sense exposes the identities of the users which publicly contribute to foil the scheme, but can be used to revoke a fixed (and a-priori known) number of users. The second, the CS scheme [25], as we have seen, has no bound on the number of users which can be revoked, but the traitors have a high level of anonymity. The hybrid inherits the nice properties of both (due to lack of space the proof of security is provided in Appendix A).

5.1 Adding Secret Sharing Schemes to CS

The hybrid scheme we describe is a modification of the CS scheme, obtained by combining it with the polynomial-based broadcast scheme described in [26] (independently introduced in [1]), which are based on a former idea of [13]. Notice that also [21] proposed a polynomial-based traitor tracing scheme, but the use of the polynomial in [21] is a bit different compared to the use done in [1,26]. Moreover, the authors of [27] showed that the scheme in [21] is not collusion resistant since two users can generate a decryption key which cannot be traced back to one of them.

Consider a cyclic group G of prime order q such that the DDH problem is believed to be hard in G. Let g be a generator of G. Each user u receives a *secret* identifier $I_u \in \mathbb{Z}_q$, chosen uniformly at random, in such a way that different users get different identifiers. Choose t different values $I_1, \ldots, I_t \in \mathbb{Z}_q$, also different from 0 and from all the previous identifiers. The scheme is given in the box in the next page. Notice that the total number of keys and the number of keys per user *remain the same* as in the basic CS scheme. For the length of the broadcasted message, $t+1$ additional elements of G are added to each fragment of the header, but, asymptotically, it remains $O(r\log N/r)$.

Pirates 2.0*: No anonymity for traitors.* The scheme completely prevents a Pirates 2.0 attack. If any user u makes public just one of his secret keys, say $(I_u, P_i(I_u))$, then the authority, immediately determines the identity of the user associated with $(I_u, P_i(I_u))$ through a look up operation in its database. Actually, we can do more: we can apply a slightly modified version of the self-enforcement technique used in [26] (see paragraph 3.1) in order to discourage users to become traitors. Briefly, a public file contains, for each user u, a row where some sensitive information S_u of user u (e.g., social security number, credit card number, etc...) is encrypted by using as a key $y_{i,u} = P_i(I_u)$, for all $logN+1$ values $P_i(I_u)$ stored by user u.

The ElGamal encryption scheme, semantically secure under the DDH assumption, is used (any CPA-secure encryption scheme is fine, but by using ElGamal, the security of the scheme follows only from the DDH assumption): the system administrator, in set up phase, for each encryption for user u, chooses a random $s \in Z_q$, computes g^s and $g^{sy_{i,u}}$, computes $C = H(g^{sy_{i,u}}||I_u) \oplus S_u$, where H is a pairwise independent hash function, and publishes $(ind_{i,u}, g^s, C)$, where $ind_{i,u}$ is a prefix of $g^{sy_{i,u}}$ used for indexing the elements of the table. If user u discloses $(I_u, P_i(I_u))$, then everybody else gain access to his sensitive information.

5.2 Subset Difference and Layered Subset Difference

The technique above described does not immediately apply to SD and LSD. Indeed, a simple generalization might consist in the use of more polynomials. More precisely, in the former construction, we associate $P_i(x)$ to subset S_i. Since in SD we have subsets defined by two indices, we could associate $P_{i,j}(x)$ to $S_{i,j}(x)$ and define, at the beginning of each session, the encryption key for subset $S_{i,j}(x)$ as $L_{i,j} = g^{rP_{i,j}(0)}$. Each user u receives a point $P_{i,j}(I_u)$, for each subset $S_{i,j}(x)$ he

Hybrid CS Construction.

Setup phase. For each complete subtree S_i

1. Choose u.a.r. a secret t-degree polynomial $P_i(x) = a_0^i + a_1^i x + \ldots + a_t^i x^t \in \mathbb{Z}_q[x]$.
2. Send to each user u covered by S_i the pair $(i, P_i(I_u))$.

Broadcast phase. At the beginning of a new session, to send message M, the broadcasting center

1. Computes a cover $\{S_{i_1}, \ldots S_{i_m}\}$ for non-revoked users in $\mathcal{N} \setminus \mathcal{R}$. Then, chooses u.a.r. a session key K, computes $F_K(M)$ and, for each $S_i \in \{S_{i_1}, \ldots S_{i_m}\}$
   (a) Choose a random $r_i \in \mathbb{Z}_q$.
   (b) Computes $\{\delta_{i,j} = g^{r_i P_i(I_j)}\}_{j=1,\ldots,t}$, $K_i = g^{r_i P_i(0)}$, and $E_{K_i}(K)$.
2. Sends
$$[(i_l, g^{r_{i_l}}, \delta_{i_l,1}, \ldots, \delta_{i_l,t}, E_{K_{i_l}}(K))_{1 \le l \le m}, F_K(M)].$$

Decryption phase. Upon receiving a broadcast message, a non-revoked user u

1. Identifies the subtree S_i he belongs to, and looks up in the header of the message the corresponding tuple $(i, \gamma, \delta_1, \ldots, \delta_t, c)$.
2. Computes $\lambda_u = \prod_{k=1}^{t} \frac{I_k}{I_k - I_u}$ and $\lambda_j = \frac{I_u}{I_u - I_j} \prod_{k=1\ (k \neq j)}^{t} \frac{I_k}{I_k - I_u}$ for $j = 1, \ldots, t$.
3. Since, from Lagrange's interpolation formula,
$$P_i(0) = \lambda_u P_i(I_u) + \sum_{j=1}^{t} \lambda_j P_i(I_j)$$
then, u computes
$$K_i = g^{r_i P_i(0)} = g^{r_i \lambda_u P_i(I_u)} \prod_{j=1}^{t} g^{r_i \lambda_j P_i(I_j)} = \gamma^{\lambda_u P_i(I_u)} \prod_{j=1}^{t} \delta^{\lambda_j}.$$
4. Finally u computes the session key $K = D_{K_i}(c)$ and $M = F_K^{-1}(F_K(M))$.

belongs to which, along with $(I_k, g^r, g^{r P_{i,j}(I_k)})$ for $k = 1, \ldots, t$, enables computing $L_{i,j}$. The drawback of this solution is that each user has to *explicitly* store a point for each subset in which he belongs to. In other words, the advantages of the pseudorandom generation of the labels (and keys) which is crucial in SD in order to store $O((\log n)^2)$ labels is lost[2]. Actually, we can do better. We could *compose* the two schemes as follows:

It is easy to check that the scheme is correct and that each user stores $O((\log n)^2)$ labels plus $\log N + 1$ points. Hence, the storage requirements stay the same of SD. Note that the same construction can be applied to the LSD scheme. Moreover, the same observations we have done for the CS case apply.

[2] A similar problem is faced in [11] when trying to generalize the SD scheme to the public key setting.

Hybrid SD Construction.

Setup phase. For each complete subtree S_i

1. Choose u.a.r. a secret t-degree polynomial $P_i(x) = a_0^i + a_1^i x + \ldots + a_t^i x^t \in \mathbb{Z}_q[x]$.
2. Generate an instance of the SD scheme with $\mathbb{Z}_q$ as set for keys $L_{i,j}$.
3. Send to each user u covered by $S_{i,*}$ the pair $(i, P_i(I_u))$ and the labels that SD assigns to him.

Broadcast phase. At the beginning of a new session, to send message M, the broadcasting center

1. Computes a cover $\{S_{i_1,j_1}, \ldots S_{i_m,j_m}\}$ for non-revoked users in $\mathcal{N}\backslash\mathcal{R}$. Then, chooses u.a.r a session key K, computes $F_K(M)$ and, for each $S_{i,j} \in \{S_{i_1,j_1}, \ldots S_{i_m,j_m}\}$
   (a) Choose a random $r_i \in \mathbb{Z}_q$.
   (b) Computes $L_{i,j}, \{g^{r_i P_i(I_k) L_{i,j}}\}_{k=1,\ldots,t}, K_{i,j} = g^{r_i P_i(0) L_{i,j}}$, and $E_{K_{i,j}}(K)$.
2. Sends

$$[(i_\ell, j_\ell, g^{r_{i_\ell}}, g^{r_{i_\ell} P_{i_\ell}(I_1) L_{i_\ell,j_\ell}}, \ldots, g^{r_{i_\ell} P_{i_\ell}(I_t) L_{i_\ell,j_\ell}}, E_{K_{i_\ell,j_\ell}}(K))]_{\ell=1,\ldots,m}.$$

Decryption phase. Upon receiving a broadcast message, a non-revoked user u

1. Identifies the subtree $S_{i,j}$ he belongs to, and looks up in the header of the message the corresponding tuple $((i,j), \gamma, \delta_1, \ldots, \delta_t, c)$.
2. Computes λ_u, λ_k for $k = 1, \ldots, t$, $L_{i,j}$ and

$$K_{i,j} = g^{r_i P_i(0) L_{i,j}} = \left(g^{r_i \lambda_u P_i(I_u)} \prod_{k=1}^{t} g^{r_i \lambda_k P_i(I_k)} \right)^{L_{i,j}} = \gamma^{\lambda_u P_i(I_u) L_{i,j}} \prod_{k=1}^{t} \delta^{\lambda_k}.$$

3. Finally u computes the session key $K = D_{K_{i,j}}(c)$ and $M = F_K^{-1}(F_K(M))$.

As reported in Theorem 1, the authors of [7] showed that in CS and LSD, traitors, by publishing keys associated to nodes belonging to levels λ such that $\lambda \leq \lfloor \log \rho \rfloor$, are guaranteed an anonymity level of N/ρ and $N/2\rho$, respectively. It is immediate to check that in the above hybrid versions, the anonymity levels drop to 1. More precisely, referring to Definition 1, we get:

Theorem 2. *The Hybrid-CS and Hybrid-SD traitor tracing schemes are 1-secure against the Pirates 2.0 attack, implemented by using projection functions and are, in terms of key and ciphertext sizes, as efficient as the former CS and SD.*

5.3 Anonymity against Private Collusion

Let us look again at the Hybrid CS Construction. Notice that, as long as traitors set up a pirate decoder by giving away privately *some pieces* of their private keys (i.e., points of the polynomials associated to nodes along the path a user belongs

to), our hybrid construction inherits exactly the same (efficient) black-box tracing procedure that uses the CS scheme. Pirate decoders are either disabled or at least a traitor is found. The scheme presented in [26] has a black-box tracing procedure which requires exponential time.

However, our scheme is subject to another form of *private* collusion, which does not permit to identify traitors. A coalition consisting on $t+1$ traitors $u_0, \ldots, u_t$, collaborating in secret, is able to recover $P_i(x)$ from their secret values $(I_{u_j}, P_i(I_{u_j}))$. Then they can learn and distribute the value $P_i(0)$ which allows a pirate decoder to compute the key $K_i = (g^r)^{P_i(0)}$ and therefore to decrypt the broadcasted content. Any $t+1$ users covered by the same subtree S_i are able to do this. In such a case, the tracing procedure can still make the decoder useless, but the anonymity of the traitors among the subset S_i is preserved. Moreover, along the same line, the scheme enables the coalition to interpolate all polynomials for which they have $t+1$ points. However, notice that, if the self-enforcement mechanism we have described before is used, *it does not* mean that the coalition *gain access* to the sensitive information of all other users who share the same polynomials: indeed, since the value I_v of user v is secret, they have no idea of which values of the polynomials are used by the users. This is also the reason for which, compared to [26], we have slightly modified the encryption in the self-enforcement technique by including I_u in the computation of the encryption $C = H(g^{sy_{i,u}}||I_u) \oplus S_u$.

Notice that this problem is also present in [26], where a coalition of $t+1$ users can interpolate the t-degree polynomial: unfortunately, therein the $t+1$ colluders gain access to the sensitive information of all revoked users, whose values are broadcasted. In our scheme, the points in broadcast are $I_1, \ldots, I_t$ which are not associated to any user. We emphasize, however, that the scheme provided in [26] is secure as long as the coalition of revoked users is not greater in size than the fixed a-priori threshold t. In our scheme, due to the CS revocation strategy, security holds w.r.t. coalitions of *any size*. What is lost if the coalition of traitors is greater than t is the protection against *anonymous* collaboration to the set up of a pirate decoder. Moreover, we could further strengthen the scheme in order to reduce the anonymity guarantee of the traitors, by using the technique employed within the Or-based Construction from [16]: associate to each node v_i a set of λ different polynomials, and assign to each user u of the subset S_i the value $P_{i,\ell}(I_u)$, for an $\ell \in \{1, \ldots, \lambda\}$, chosen uniformly at random. If both measures are used the total length of the broadcast message in CS moves from $O(r \log \frac{N}{r})$ to $O(\lambda tr \log \frac{N}{r})$ and in SD from $O(r)$ to $O(\lambda tr)$.

Nevertheless, notice that one of the motivations for introducing the Pirates 2.0 attack was that public collusion is easier than private collusion. Indeed, the first one enables large coalitions, since users consider themselves protected by anonymity; on the other hand, the second does not provide any guarantee to traitors, they need to trust each other and *share* their sensitive information, which means that large coalitions should be difficult to set up.

Using Cryptography against Cryptography. Notice that the above hybrid schemes could also be attacked by a group of traitors who "publicly" collaborate

but still *preserve* the privacy of their inputs. Indeed, it is not difficult to see that the computation of $P_i(0)$ can be casted as an instance of the general multi-party computation problem, where each traitor has an input, the traitors jointly want to compute the output of the function, but each of them wants to keep private his own input, i.e., without disclosing any information about it, apart what can be inferred from the output of the function. However, such an attack requires a set up phase and, if the computations are publicly carried out, some other forms of protection for the traitors (e.g., anonymous communication channels) have to be considered. We do not go into details but emphasize that such a strategy could be pursued. Our scheme is not robust against it. We are only protected by the threshold on the size of the coalition of traitors. It seems an interesting open problem to study the resistance of known broadcast encryption schemes against this type of attacks.

6 Conclusions and Open Problems

In this paper we started studying measures to deal with the Pirates 2.0 attack strategy. Among these, we have proposed an hybrid scheme, which combines a scheme which enables efficient tracing and revoking with a scheme which exposes traitors who publicly disclose secret information. Several open problems need to be addressed: first of all, it is necessary to clearly conceptualize and formally define a *security model* which incorporates the Pirates 2.0 attack strategy and all forms of public collaborations. Then, a scheme which fulfills all requirements should be designed and rigorously proven secure in the model.

Acknowledgments. This work was partially done while the second author was hosted at the Dipartimento di Infomatica, University of Salerno. The authors are supported by Project MTM2010-15167 from the Spanish Ministry of Science and Technology.

References

1. Anzai, J., Matsuzaki, N., Matsumoto, T.: A quick group key distribution scheme with "Entity revocation". In: Lam, K.-Y., Okamoto, E., Xing, C. (eds.) ASIACRYPT 1999. LNCS, vol. 1716, pp. 333–347. Springer, Heidelberg (1999)
2. Berkovits, S.: How to broadcast a secret. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 535–541. Springer, Heidelberg (1991)
3. Boneh, D., Gentry, C., Waters, B.: Collusion resistant broadcast encryption with short ciphertexts and private keys. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 258–275. Springer, Heidelberg (2005)
4. Boneh, D., Naor, M.: Traitor tracing with constant size ciphertext. In: CCS 2008: Proceedings of the 15th ACM Conference on Computer and Communications Security, pp. 501–510. ACM, New York (2008)
5. Boneh, D.: The decision diffie-hellman problem. In: Buhler, J.P. (ed.) ANTS 1998. LNCS, vol. 1423, pp. 48–63. Springer, Heidelberg (1998)

6. Billet, O., Phan, D.H.: Efficient traitor tracing from collusion secure codes. In: Safavi-Naini, R. (ed.) ICITS 2008. LNCS, vol. 5155, pp. 171–182. Springer, Heidelberg (2008)
7. Billet, O., Phan, D.H.: Traitors collaborating in public: Pirates 2.0. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 189–205. Springer, Heidelberg (2009)
8. Chor, B., Fiat, A., Naor, M.: Tracing traitors. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 257–270. Springer, Heidelberg (1994)
9. Canetti, R., Garay, J., Itkis, G., Micciancio, D., Naor, M., Pinkas, B.: Multicast security: A taxonomy and some efficient constructions. In: Proceedings of INFOCOMM 1999, pp. 708–716 (1999)
10. D'Arco, P., Perez del Pozo, A.L.: Fighting Pirates 2.0 (full version), `http://www.dia.unisa.it/~paodar`
11. Dodis, Y., Fazio, N.: Public key broadcast encryption for stateless receivers. In: Feigenbaum, J. (ed.) DRM 2002. LNCS, vol. 2696, pp. 61–80. Springer, Heidelberg (2003)
12. Delerablée, C., Paillier, P., Pointcheval, D.: Fully collusion secure dynamic broadcast encryption with constant-size ciphertexts or decryption keys. In: Takagi, T., Okamoto, T., Okamoto, E., Okamoto, T. (eds.) Pairing 2007. LNCS, vol. 4575, pp. 39–59. Springer, Heidelberg (2007)
13. Feldman, P.: A practical scheme for non-interactive verifiable secret sharing. In: Proceedings of the 28th IEEE Symp. on Foundations of Computer Science, pp. 427–437 (1987)
14. Fiat, A., Naor, M.: Broadcast encryption. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 480–491. Springer, Heidelberg (1994)
15. Goodrich, M.T., Sun, J.Z., Tamassia, R.: Efficient tree-based revocation in groups of low-state devices. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 511–527. Springer, Heidelberg (2004)
16. Gafni, E., Staddon, J., Yin, Y.L.: Efficient methods for integrating traceability and broadcast encryption. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 372–387. Springer, Heidelberg (1999)
17. Hwang, Y.H., Lee, P.J.: Efficient broadcast encryption scheme with log-key storage. In: Di Crescenzo, G., Rubin, A. (eds.) FC 2006. LNCS, vol. 4107, pp. 281–295. Springer, Heidelberg (2006)
18. Hwang, J.Y., Lee, D.H., Lim, J.: Generic transformation for scalable broadcast encryption schemes. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 276–292. Springer, Heidelberg (2005)
19. Halevy, D., Shamir, A.: The LSD broadcast encryption scheme. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 47–60. Springer, Heidelberg (2002)
20. Jho, N., Hwang, J.Y., Cheon, J.H., Kim, M., Lee, D.H., Yoo, E.S.: One-way chain based broadcast encryption schemes. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 559–574. Springer, Heidelberg (2005)
21. Kurosawa, K., Desmedt, Y.: Optimum traitor tracing and asymmetric schemes. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 145–157. Springer, Heidelberg (1998)
22. Kiayias, A., Pehlivanoglu, S.: Pirate evolution: How to make the most of your traitor keys. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 448–465. Springer, Heidelberg (2007)
23. Kumar, R., Rajagopalan, S., Sahai, A.: Coding constructions for blacklisting problems without computational assumptions. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 609–623. Springer, Heidelberg (1999)

24. Luby, M., Staddon, J.: Combinatorial bounds for broadcast encryption. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 512–526. Springer, Heidelberg (1998)
25. Naor, D., Naor, M., Lotspiech, J.: Revocation and tracing schemes for stateless receivers. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 41–62. Springer, Heidelberg (2001), Full version available at `http://www.wisdom.weizmann.ac.il/~naor/`
26. Naor, M., Pinkas, B.: Efficient trace and revoke schemes. In: Frankel, Y. (ed.) FC 2000. LNCS, vol. 1962, pp. 1–20. Springer, Heidelberg (2001)
27. Stinson, D.R., Wei, R.: Key preassigned traceability schemes for broadcast encryption. In: Tavares, S., Meijer, H. (eds.) SAC 1998. LNCS, vol. 1556, pp. 144–156. Springer, Heidelberg (1999)

A Security of the Schemes

Tools. The schemes we have proposed, the Or-based construction, the Hybrid CS construction and the Hybrid SD Construction, still belong to the Subset-Cover framework. All we have changed are the key-assignment methods. Hence, in order to prove that the schemes are robust against coalitions of revoked users of any size, we can use the general results for the framework shown in [25]. Specifically, we need to show that our key-assignment methods satisfy the *key-indistinguishability* property. Let us start by recalling some definitions given in [25].

Definition 5. *(*Key-Indistinguishability*)*
Let $\mathcal{A}$ be a Subset-Cover revocation algorithm that defines a collection of subsets $S_1, \ldots, S_\omega$. Consider a feasible adversary $\mathcal{B}$ that: a) selects i, $1 \leq i \leq \omega$; b) receives the I_u's (secret information that u receives) for all $u \in \mathcal{N} \setminus S_i$. We say that $\mathcal{A}$ satisfies the key-indistinguishability property if the probability that $\mathcal{B}$ distinguishes a key L_i from a random string R_{L_i} of the same length is negligible.

The adversary we deal with is characterized as follows:

Definition 6. *(*Adversarial Model*)*
Consider an adversary $\mathcal{B}$ that:

1. *Selects adaptively a set $\mathcal{R}$ of receivers and obtains I_u for all $u \in \mathcal{R}$. By adaptively we mean that $\mathcal{B}$ may select messages $M_1, M_2, \ldots$, and revocation sets $\mathcal{R}_1, \mathcal{R}_2, \ldots$ (the revocation sets need not correspond to the actual corrupted users) and see the encryption of M_i when the revoked sets is $\mathcal{R}_i$. Also $\mathcal{B}$ can create a ciphertext and see how any (non-corrupted) user decrypts it. It then asks to corrupt a receiver u and obtains I_u. This steps is repeated $|\mathcal{R}|$ times (for any $u \in \mathcal{R}$).*
2. *Choose a message M as the challenge plaintext and a set $\mathcal{R}$ of revoked users that must include all the ones it corrupted (but may contain more).*

$\mathcal{B}$ then receives an encrypted message M' with revoked set $\mathcal{R}$. It has to guess whether $M' = M$ or $M' = R_M$, where R_M is a random message of the same length. We say that a revocation scheme is secure if, for any (probabilistic polynomial time) adversary $\mathcal{B}$ as above, the probability that $\mathcal{B}$ distinguishes between the two cases is negligible.

The main result shown in [25] for the subset-cover framework is:

Theorem 3. *Let $\mathcal{A}$ be a subset-cover revocation algorithm where the key assignment satisfies the key-indistinguishability property and where E and F are two encryption schemes $CCA-I$ secure and CPA-secure, respectively. Then $\mathcal{A}$ is secure in the sense of Definition 6 with security parameter $\delta < \epsilon_1 + 2m\omega(\epsilon_2 + 4\omega\epsilon_3)$, where ω is the total number of subsets in the scheme, m is the maximum size of a cover, and ϵ_1, ϵ_2 and ϵ_3 are the probabilities associated to the key-assignment, E and F.*

It is immediate to see that the key-assignment method of the Or-based construction trivially satisfies the key-indistinguishability definition, since all keys are chosen independently and uniformly at random. In the following, we prove that also the key-assignment for the hybrid CS scheme satisfies the key-indistinguishability property. The proof for the SD hybrid scheme, which is obtained by using similar techniques, is included in the full version [10].

Hybrid CS Construction. The key-assignment method for the CS Hybrid Construction satisfies the key-indistinguishability property, assuming that the DDH assumption in the group G holds. The DDH assumption involves a cyclic group G and a generator g. Loosely speaking, it states that no efficient algorithm can distinguish between the two distributions $< g^a, g^b, g^{ab} >$ and $< g^a, g^b, g^c >$, where a, b, c are randomly chosen in $\{1, \ldots, |G|\}$. The reader is referred to [5] for details about the assumption.

More precisely, by using a similar technique to the one employed in [26] we show that if an adversary (i.e., a PPT algorithm) distinguishes a session key from a random value, then such an adversary can be turned into an efficient procedure which solves the DDH problem. We prove the following result:

Theorem 4. *If the DDH assumption holds, then any probabilistic polynomial time adversary $\mathcal{B}$, which operates according to Definition 5, does not distinguish a key K_i associated to subset S_i from a random value.*

Proof. By contradiction. Let us assume that there exists an efficient adversary $\mathcal{B}$, which has received the private information I_u of all users but users in S_i, and is able, with non negligible probability, after a sequence of a poly number m of executions, to distinguish a key K_i associated to S_i from a random value R_i. At the beginning of each session, $\mathcal{B}$ sees vectors of the form $< g^r, I_1, \ldots, I_t, g^{rP_i(I_1)}, \ldots, g^{rP_i(I_t)} >$ where r is a new random value, sent to enable users in S_i to compute the new key K_i. Let us denote by $< history >$ the sequence of *all* vectors sent at the beginning of sessions $1, 2, \ldots, m-1$ to enable *all* authorized subsets to compute the new session key, and let us

assume that $(< history >, < g^r, I_1, \ldots, I_t, g^{rP_i(I_1)}, \ldots, g^{rP_i(I_t)} >, C)$, where $< g^r, I_1, \ldots, I_t, g^{rP_i(I_1)}, \ldots, g^{rP_i(I_t)} >$ is the vector for S_i for the m-th session, and C is either the key $K_i = g^{rP_i(0)}$ or a random element $R_i \in G$, is the input of $\mathcal{B}$. On such an input, $\mathcal{B}$, independently of the specific revocation strategy, guesses with non-negligible probability what the challenge C is.

Then, we can turn $\mathcal{B}$ into an efficient algorithm $\mathcal{B}'$ which solves the DDH problem, i.e., $\mathcal{B}'$ takes in input a triple $< \alpha, \beta, \gamma >$ and outputs with non-negligible probability whether $< \alpha, \beta, \gamma >=< g^a, g^b, g^{ab} >$ or $< \alpha, \beta, \gamma >=< g^a, g^b, g^c >$, where a, b, and c are chosen u.a.r in Z_q. Such an algorithm $\mathcal{B}'$ works as follows:

- By using the public system parameters, g, q, p, $\mathcal{B}'$ generates an instance of the hybrid scheme. In particular, it generates secret keys for all users but the users in S_i. Chooses u.a.r values $(I_1, \ldots, I_t, P_i(I_1), \ldots, P_i(I_t))$, and associates it to subset S_i.
- For session $\ell = 1, \ldots, m-1$, $\mathcal{B}'$ constructs $< history >$ by choosing at random a revocation strategy and revoking all users but users in S_i. Since $\mathcal{B}'$ holds all secret keys but the ones held by users in S_i, then he can construct all encryptions (vectors) he needs for subsets different from S_i. Moreover, any time $\mathcal{B}'$ needs to encrypt the key for users in S_i, it chooses u.a.r a value r' and constructs the vector $< \alpha^{r'}, I_1, \ldots, I_t, \alpha^{r'P_i(I_1)}, \ldots, \alpha^{r'P_i(I_t)} >$.
- Run $\mathcal{B}$ on input $< history >, < \alpha, I_1, \ldots, I_t, \alpha^{P_i(I_1)}, \ldots, \alpha^{P_i(I_t)} >, \gamma$.
- If $\mathcal{B}$ outputs *key*, then output *DH triple*. Otherwise, output *random triple*.

The idea is to embed the challenge C in the m-th session, i.e., by implicitly imposing that $r = a, P_i(0) = b$. In such a way $g^r = g^a, g^b = g^{P_i(0)}$ and $g^{ab} = g^{rP_i(0)}$. It is important to note that $\mathcal{B}$ does not distinguish a *real* $< history >$ from the $< history >$ generated by $\mathcal{B}'$ by using the triple $< \alpha, \beta, \gamma >$. They are *identically distributed*. Indeed:

- the vectors for subsets different from S_i are distributed exactly as in a real execution of the protocol, i.e., we are perfectly simulating the protocol.
- the vectors for S_i use the challenge $< \alpha, \beta, \gamma >$, a different value r at each session, and the random values $P_i(I_1), \ldots, P_i(I_t)$. Hence, it follows that they are "consistent with" a *unique* t-degree polynomial P_i, chosen uniformly at random, implicitly defined by the pairs $(0, b), (I_1, P_i(I_1)), \ldots, (I_t, P_i(I_t))$.

Moreover, $\mathcal{B}'$ runs in probabilistic polynomial time and distinguishes a DH triple from a random triple exactly with the *same advantage* with which $\mathcal{B}$ distinguishes a true key from a random value. Hence, if the DDH assumption in G holds, then $\mathcal{B}'$ (and thus $\mathcal{B}$) does not exist, and the key-assignment method satisfies the key-indistinguishability property. □