2

Broadcast Encryption

A broadcast channel enables a sender to reach many receivers in a very effective way. Broadcasting, due to its very nature, leaves little room for controlling the list of recipients N — once a message is put on the channel any listening party can obtain it. This may very well be against the objectives of the sender. In such case, encryption comes in mind as a potential way to solve the problem: it can be employed to deny eavesdroppers free access to the content that is broadcasted. Nevertheless, the use of encryption raises the issue of how to do key management. Enabled receivers should be capable of descrambling the message while eavesdroppers should just perceive it as noise. It follows that receivers that are enabled for reception should have access to the decryption key, while any other party should not. The major problem that springs up in this scenario is that receivers might get corrupted and thus become cooperative with the adversary. As a result one cannot hope that a party that owns a key will not use it to the fullest extend possible, i.e., for as long as such key allows descrambling which can be the moment that a global rekey operation takes place. Moreover, such a key can even be shared with more than a single listening party and thus enable the reception of the transmission for a multitude of rogue receivers. If a traditional encryption scheme is used then a single corrupted receiver is enough to bring forth such undesired effects. The subject of this chapter, *broadcast encryption* deals with solving the above problem in an effective way.

Based on the above, a path to effective broadcast encryption that avoids rekeying is that all recipients should have different but related keys. Taking advantage of the structure of the key space, the sender should be capable of choosing on the fly any set R of revoked receivers to be excluded from a transmission, and given such R prepare an encryption that can only be decrypted by the set of receivers $\mathsf{N} \setminus \mathsf{R}$.

We can classify broadcast encryption schemes in two major categories. The first one, that can be called *combinatorial*, is characterized as follows : the key-space contains cryptographic keys suitable for a standard encryption scheme. Each user receives a subset of those keys according to some assignment mapping. In the setting of combinatorial schemes, we can think that each key corresponds to a set of users. The transmission problem then becomes a type of a set-cover problem: given the set of enabled users $\mathsf{N} \setminus \mathsf{R}$ find the best way to cover it using the subsets that correspond to the assigned keys. Combinatorial schemes can be constructed by employing probabilistic techniques or with explicit constructions. The second category of broadcast encryption, that can be called *structured*, assumes that the key-space has some structure that enables the preparation of ciphertexts that are decipherable only by the enabled users. For example a polynomial function can be used as a master key and each user can own a point of this polynomial. Messages can be encrypted under a point of a related polynomial and successful decryption can be achieved by the ability to interpolate the related polynomial.

In this chapter we will focus on explicit combinatorial schemes. An important characteristic of such schemes is that they are suitable for efficient implementation as they can be readily paired with an efficient underlying block-cipher such as the Advanced Encryption Standard[1] to yield very effective broadcast encryption in the symmetric key setting. The explicitness of such constructions guarantees that there is no error probability in the expression of their efficiency and security guarantees. Moreover, for such schemes, as we will illustrate, there is a way to express sufficient requirements for effective broadcast encryption in a compact algebraic fashion.

We note that most broadcast encryption schemes incur an overhead in the transmission that makes them unsuitable for the delivery of long plaintexts due to efficiency degradation. This issue is fairly common with cryptographic functions with special properties with the most prominent example being public-key encryption that includes schemes such as ElGamal and RSA. The way this is dealt in practice is through a hybrid approach. In particular two levels of encryption are used: at the first layer, the encryption is employed to encrypt a one-time key. Next, at the second layer, an efficient block or stream cipher is employed in combination with the one-time key. In this chapter we will assume that this approach is taken for the deployment of a broadcast encryption scheme.

2.1 Definition of Broadcast Encryption

A broadcast encryption scheme $\mathtt{BE}$ is a triple $(\mathbf{KeyGen}, \mathbf{Encrypt}, \mathbf{Decrypt})$ of algorithms. The parameter of the scheme is n, the number of receivers and is associated with three sets $\mathsf{K}, \mathsf{M}, \mathsf{C}$ corresponding to the sets of keys, plaintexts and ciphertexts respectively. We describe the I/O of these algorithms below:

- **KeyGen**. It is a probabilistic algorithm that on input 1^n, it produces $(ek, sk_1, \ldots, sk_n)$. The decryption key sk_i is to be assigned to the i-th user

[1] The Advanced Encryption Standard [32] is a symmetric encryption scheme adopted by the National Institute of Standards and Technology, USA in 2002.

while ek is the encryption key. The algorithm also produces a membership test for a language $\mathcal{L}$. The language $\mathcal{L}$ encodes all possible revocation instructions for the encryption function.

- **Encrypt**. It is a probabilistic algorithm that on input $m \in \mathsf{M}$, a string $\psi \in \mathcal{L}$ and ek, it outputs a ciphertext $c \in \mathsf{C}$. We write $c \leftarrow \mathbf{Encrypt}(ek, m, \psi)$ to denote that c is sampled according to the distribution of the encryptions of the plaintext m based on the revocation instruction ψ.
- **Decrypt**. It is a deterministic algorithm that on input c sampled from $\mathbf{Encrypt}(ek, m, \psi)$ and a user-key $sk_i \in \mathsf{K}$ where $(ek, sk_1, \ldots, sk_n) \leftarrow \mathbf{KeyGen}(1^n)$, it either outputs m or fails. Note that **Decrypt** can also be generalized to be a probabilistic algorithm but we will not take advantage of this here.

A broadcast encryption scheme BE can be in the public or symmetric key setting by signifying that the encryption key ek is either public or secret respectively. In case of public encryption this would enable any party to use the broadcast encryption to distribute content to the receiver population. A natural generalization of the above definition (which we will not consider in this chapter) is to accept a vector of messages $M = \langle m_1, \ldots, m_s \rangle \in \mathsf{M}^s$ so that **Decrypt** either outputs m_i for some $i \in [s]$ or fails. We call such scheme an s-ary broadcast encryption.

Regarding the language of revocation instructions we will require that it contains at least the descriptions of some subsets $\mathsf{R} \subseteq [n]$. The way a certain subset R is described by a revocation instruction varies and there can even be many different revocation instructions resulting in the same set of revoked users R. Depending on the scheme it might be the case that any subset of indices R can be encoded in $\mathcal{L}$ or there are only some specific subsets that are included, e.g., all subsets up to a certain size.

Next we define the correctness properties that are required from a broadcast encryption scheme.

Definition 2.1. ***Correctness.*** *We say an s-ary broadcast encryption scheme is correct if for any $\psi \in \mathcal{L}$ that encodes a subset $\mathsf{R} \subseteq [n]$ and for all $M = \langle m_1, \ldots, m_s \rangle \in \mathsf{M}^s$ and for any $u \in [n] \setminus \mathsf{R}$, it holds that*

$$\mathbf{Prob}[\mathbf{Decrypt}(\mathbf{Encrypt}(ek, M, \psi), sk_u) \in \{m_1, \ldots, m_s\}] = 1$$

where $(ek, sk_1, \ldots, sk_n)$ is distributed according to $\mathbf{KeyGen}(1^n)$. Naturally one may generalize the above definition to have decryption fail with some small probability.

The correctness definition ensures that the **Decrypt** algorithm does not fail as long as the index u is not removed from the list of enabled users.

Efficiency Parameters.

The efficiency of a broadcast encryption scheme is evaluated according to the following parameters.

1. Key-Storage: This refers to the size of the information required for each receiver to store so that the decryption operation is enabled.
2. Decryption Overhead: This refers to the computation time required by a receiver in order to perform the recovery of the plaintext.
3. Encryption Overhead: This refers to the computation time the sender is supposed to invest in order to parse the given revocation instruction and sample the ciphertext that disables all users that are meant to be excluded from the transmission and produce the ciphertext.
4. Transmission Overhead. This refers to the actual length of the ciphertexts (or the maximum such length if it varies).

The above parameters will have a functional dependency in the number of users n, as well as on possibly other parameters such as the number of users that the revocation information instructs to be excluded.

Adversarial Model.

The goal of an adversary in the broadcast encryption setting is to circumvent the revocation capability of the sender. In a setting where the hybrid encryption approach is employed, the content distribution operates at two levels: first, a one-time content key k is selected and encrypted with the broadcast encryption mechanism. Second, the actual message will be encrypted with the key k and will be broadcasted alongside the encrypted key. It follows that a minimum requirement would be that the scheme $\mathtt{BE}$ should be sufficiently secure to carry a cryptographic key k. As an encryption mechanism this is known in the context of public key cryptography as a “Key Encapsulation Mechanism”. The security model we present in this section will take this formalization approach, i.e., it will focus on the type of security that needs to be satisfied by a broadcast encryption scheme in order to be used as a key encapsulation mechanism. We note that for simplicity we adopt the syntax of an encryption scheme, i.e., the message is given as an input to the encryption algorithm, while the security property will capture the case where the message is uniformly random as in key encapsulation. Later on in the chapter, the plaintext m in the definition of broadcast encryption schemes will be used to mean the one-time content key k unless otherwise noted.

The adversarial scenario that we envision for broadcast encryption is as follows. The adversary is capable of corrupting a set of users so that the adversary has access to the key material of the users in the corrupted set T. Subsequently, the adversary, given a pair (c, m), tries to distinguish if the pair is an actual plaintext ciphertext pair where m is sampled uniformly at random, i.e., the adversary attempts to see whether c is an encryption of m or m has been sampled in a manner independent of c. If indeed the adversary has no means of distinguishing a valid encryption key pair from an invalid one, then the encryption mechanism would be sufficiently strong to be used for the distribution of cryptographic keys.

The adversary may have at its disposal the following resources that can be thought of as oracles it can query and obtain a response.

1. Chosen Plaintext. The adversary can obtain valid plaintext-ciphertext pairs. In the case that broadcast encryption is used only for cryptographic key distribution the adversary may not be able to influence the distribution of plaintext - nevertheless allowing this capability only makes the security property stronger. When the adversary requests an encryption it will also be allowed to specify the set of revoked users or even choose the revocation information ψ that is passed to the encryption algorithm.
2. Chosen Ciphertext. The adversary can obtain output about how a certain uncorrupted user responds to a decryption request. The query may not necessarily contain a valid ciphertext but rather it can be an arbitrary bitstring created by the adversary to see how a user reacts in decryption.
3. User Corruption. In the static corruption setting, the adversary obtains the key material of all users in a set $\mathsf{T} \subseteq [n]$. In the adaptive corruption setting, the adversary corrupts each user one by one after performing other operations as allowed in the course of the attack.

The security of a broadcast encryption scheme will be defined using a game between the adversary and the challenger. We say the adversary has broken the scheme when the revocation list contains all of the corrupted users, but the adversary, still, is capable of distinguishing a valid plaintext-ciphertext pair from a pair where the plaintext is independent of the ciphertext and uniformly random. In figure 2.1 we present the security game that captures the security for key-encapsulation that we require from a broadcast encryption scheme in order to be useful in a hybrid encryption setting.

```
EncryptOracle($m, \psi$)
    retrieve $ek$;
    $c \leftarrow \mathbf{Encrypt}(ek, m, \psi)$;
    return $c$;

DecryptOracle($c, u$)
    retrieve $sk_u$;
    return $\mathbf{Decrypt}(c, sk_u)$;

CorruptOracle($u$)
    $\mathsf{T} \leftarrow \mathsf{T} \cup \{u\}$
    retrieve $sk_u$;
    return $sk_u$;

Experiment $\mathbf{Exp}_{\mathcal{A}}^{rev}(1^n)$
    $(ek, sk_1, \ldots, sk_n) \leftarrow \mathbf{KeyGen}(1^n)$; $\mathsf{T} \leftarrow \emptyset$
    $\psi \leftarrow \mathcal{A}^{\mathsf{EncryptOracle}(),\mathsf{DecryptOracle}(),\mathsf{CorruptOracle}()}(1^n)$
    $M_0, M_1 \overset{R}{\leftarrow} \mathsf{M}^s$; $b \overset{R}{\leftarrow} \{0,1\}$; $c \leftarrow \mathbf{Encrypt}(ek, M_1, \psi)$
    $b' \leftarrow \mathcal{A}^{\mathsf{EncryptOracle}()}(\{sk_i\}_{i \in \mathsf{T}}, M_b, c)$
    return 1 if and only if $b = b'$ and
                        $\psi$ excludes all members of $\mathsf{T}$.
```

Fig. 2.1. The security game for key encapsulation.

In the definition below we introduce the notion of ε-insecurity that captures the advantage the adversary may have in distinguishing valid plaintext ciphertext pairs from those that are independently chosen.

Definition 2.2. *We say an s-ary broadcast encryption* BE *is ε-insecure if for any probabilistic polynomial-time adversary $\mathcal{A}$, it holds that*

$$\mathbf{Adv}_{\mathcal{A}}^{rev}(1^n) = |\mathbf{Prob}[\mathbf{Exp}_{\mathcal{A}}^{rev}(1^n) = 1] - \frac{1}{2}| \leq \varepsilon$$

where the experiment is defined as in figure 2.1. It is also possible to extend the definition to accept a vector of messages $M = \langle m_1, \ldots, m_s \rangle \in \mathsf{M}^s$ as an input; it will be considered to be correct if **Decrypt** *returns m_i for some $i \in [s]$.*

We note that ε in general is not supposed to be a function n, i.e., the security property should hold for any $\mathcal{A}$ i.e., independently of the number of users n.

2.2 Broadcast Encryption Based on Exclusive-Set Systems

In this section we will focus on concrete combinatorial broadcast encryption schemes. These are also the only such schemes that are currently widely deployed in commercial products (a notable example of such deployment is the AACS[2]). Recall that in combinatorial schemes there is a pool of cryptographic keys for an underlying encryption scheme such as a block cipher. The message m to be broadcasted is encrypted with some of these keys. In order to receive it, the user will need to either possess or be able to derive at least one of these keys.

Given that the keys in the pool are shared by many users, we can obtain a correspondence between such keys and subsets so that a key would correspond to the set of users who possess that key. Hence, the set of keys corresponds to a collection of subsets of users, who without loss of generality are subsets of $[n]$. This collection defines a set system over the user population. The set of keys that are used in a certain transmission of a plaintext is mapped to a set of subsets from the collection that we call the "broadcast pattern" or simply pattern of the transmission. Hence, encryption in this case involves the problem of finding a set of subsets, i.e. a broadcast pattern, that covers the enabled set of receivers.

The reader should observe that the choice of the set system that underlies the assignment of cryptographic keys will play a crucial role in the effectiveness of revocation. As it is quite clear, not any set system would provide a feasible way to revoke any subset of receivers. We will start the investigation of this topic by formally defining exclusive set systems, that are instances of set systems useful for broadcast encryption.

[2] The Advanced Access Content System (AACS, see [1]) is a standard for content distribution and digital rights management, intended to restrict access to and copying of optical discs such as Blu-Ray disks.

Definition 2.3. *Consider a family of subsets $\Phi = \{\mathsf{S}_\mathsf{j}\}_{\mathsf{j}\in\mathcal{J}}$ defined over $[n]$ where $\mathcal{J}$ denotes the set of encodings for the elements in Φ over an alphabet Σ with length of at most $\mathsf{l}(n)$ for some length function $\mathsf{l}(\cdot)$. We say Φ is (n, r, t)-exclusive if for any subset $\mathsf{R} \subseteq [n]$ with $|\mathsf{R}| \le r$, we can write $[n] \setminus \mathsf{R} = \cup_{i=1}^{s} \mathsf{S}_{\mathsf{j}_i}$ where $s \le t$ and $\mathsf{S}_{\mathsf{j}_i} \in \Phi$ for $1 \le i \le s$.*

Having defined exclusive-set systems, we will now give a construction for broadcast encryption schemes based on exclusive-set systems. We note that the recovery of $\mathsf{j}_1, \ldots, \mathsf{j}_s$ given R should be done efficiently for a system to be useful. For this reason when one proposes a set system it is imperative to include a description of how the covering algorithm would work (that is at the heart of the revocation algorithm). The goal of this algorithm would be to produce the indices j_i, $i = 1, \ldots, s$ of the subsets that cover the set of enabled users $[n] \setminus \mathsf{R}$. A trivial covering algorithm for revocation that works with any set system would be to search for the pattern in a brute force manner. Given that in an exclusive set system the target pattern is postulated to exist, the exhaustive search algorithm is guaranteed to find a solution. Nevertheless this will not lead to an efficient implementation for any but entirely trivial set systems. In the rest of the chapter we will be concerned with the design of exclusive set systems with efficient revocation algorithms and in fact we will present a generic revocation algorithm given any set system that satisfies a simple algebraic property.

In Figure 2.2 we present a template for a broadcast encryption system based on exclusive set systems. We assume a family of exclusive set systems indexed by n as well as an underlying symmetric encryption scheme $(\mathtt{E}, \mathtt{D})$ that uses as keys elements of K.

The characteristics of the scheme given in figure 2.2 that make it a template are as follows:

1. The exclusive set system Φ is not explicitly specified but used in a black-box fashion. It is assumed that an exclusive set system can be found for any number of users n. To specify this we may use the notation $\{\Phi_n\}_{n\in\mathbb{N}}$ to denote the collection of exclusive set systems produced for each number of users n, or simply write Φ overloading the set-system notation (in such case there an implicit reference to the number of users n will be assumed).
2. The exact mechanism that **KeyGen** uses to sample the keys $\{k_\mathsf{j}\}_{\mathsf{j}\in\mathcal{J}}$ is not specified. The only restriction is that each key belongs to the set K.
3. The underlying encryption scheme $(\mathtt{E}, \mathtt{D})$ is not specified but used in a black box fashion.

Comments on the Template Construction.

For any exclusive set system Φ and an encryption scheme $(\mathtt{E}, \mathtt{D})$ we can instantiate the template of Figure 2.2 by having the **KeyGen** procedure sample the keys $\{k_\mathsf{j}\}_{\mathsf{j}\in\mathcal{J}}$ independently at random from the set of keys for the encryption scheme $(\mathtt{E}, \mathtt{D})$. We will refer to this scheme as : $\mathtt{BE}^{\Phi}_{\mathtt{basic}}$. As we will see later

Combinatorial broadcast encryption template.

- **KeyGen**. Given 1^n it chooses an (n,r,t)-exclusive set system $\Phi = \{\mathsf{S_j}\}_{\mathsf{j}\in\mathcal{J}}$. The algorithm then generates a collection of keys $\{k_\mathsf{j}\}_{\mathsf{j}\in\mathcal{J}} \subseteq \mathsf{K}$. For any $u \in [n]$, define $\mathcal{J}_u := \{\mathsf{j} \mid u \in \mathsf{S_j}\}$ and $\mathsf{K}_u = \{k_\mathsf{j} \mid \mathsf{j} \in \mathcal{J}_u\}$. Set $ek = \langle \Phi, \{k_\mathsf{j}\}_{\mathsf{j}\in\mathcal{J}}\rangle$ and set $sk_u = (\mathcal{J}_u, \mathsf{K}_u)$ for any $u \in [n]$.
  The language $\mathcal{L}$ consists of the descriptions of those elements of 2^Φ such that $\mathcal{P} = \{\mathsf{S}_{\mathsf{j}_1}, \ldots, \mathsf{S}_{\mathsf{j}_s}\} \in \mathcal{L}$ if and only if $s \le t$ and the set $\mathsf{R} = [n] \setminus \cup_{i=1}^{s} \mathsf{S}_{\mathsf{j}_i}$ satisfies $|\mathsf{R}| \le r$; in such case we say that $\mathcal{P}$ encodes R.
- Encrypt. Given $\mathcal{P} \in \mathcal{L}$ and a message m, say $\mathcal{P} = \{\mathsf{S}_{\mathsf{j}_1}, \ldots, \mathsf{S}_{\mathsf{j}_s}\}$ where $\mathsf{j}_i \in \mathcal{J}$ for $i \in \{1, \ldots, s\}$. Then the set of keys $\{k_\mathsf{j} \mid \mathsf{j} \in \mathcal{J} \text{ and } \mathsf{S_j} \in \mathcal{P}\}$ is selected from $\{k_\mathsf{j}\}_{\mathsf{j}\in\mathcal{J}}$. By employing the encryption scheme $(\mathtt{E}, \mathtt{D})$ the ciphertext is computed as follows:
  $$c \leftarrow \langle \mathsf{j}_1, \ldots, \mathsf{j}_s, \mathtt{E}_{k_{\mathsf{j}_1}}(m), \ldots, \mathtt{E}_{k_{\mathsf{j}_s}}(m)\rangle$$
- Decrypt. Given the key-pair $sk_u = (\mathcal{J}_u, \mathsf{K}_u)$ for some $u \in [n]$ and a ciphertext of the form
  $$c = \langle \mathsf{j}_1, \ldots, \mathsf{j}_s, c_1, \ldots, c_s\rangle$$
  it first searches for an encoding j_i that satisfies $\mathsf{j}_i \in \mathcal{J}_u$ and then returns $\mathtt{D}_{k_{\mathsf{j}_i}}(c_i)$. If no such encoding is found it returns $\perp$.

Fig. 2.2. The construction template for broadcast encryption using an exclusive set system.

on in this chapter there are substantial advantages to be gained by exploiting the particular structure of the exclusive set system and packing the information in the sets $(\mathcal{J}_u, \mathsf{K}_u)$ in a more compact form than simply listing all their elements. In this way we will derive much more efficient schemes compared to $\mathtt{BE}^{\Phi}_{\mathtt{basic}}$. This gain will come at the expense of introducing additional cryptographic assumptions in the security argumentation.

The three procedures in the template broadcast encryption scheme BE play the following role in an actual system instantiation. The **KeyGen** procedure produces a set system Φ which corresponds to the set of keys in the system and the collection of sets $\mathcal{I}_u$ which determines the key assignment for each user u. The procedure **Encrypt**, given the revocation instructions and a message m to be distributed, produces the ciphertext by choosing the corresponding keys from the set of possible keys. This is done by computing the encryption of the plaintext m under the key assigned to the subset S for all subsets that are specified in the revocation instruction. The Decrypt procedure will decrypt the content transmission by using the set of user keys in a straightforward manner : it will parse the transmitted ciphertext sequence for a ciphertext block that it can decrypt and then it will apply the correponding key to it to recover m.

The efficiency of the above construction template depends on the characteristics of the underlying set system and the way **KeyGen** works. Key storage is bounded from above by the number of keys in K_u it can be decreased in favor of increasing the decryption overhead. Such tradeoffs will be possible by either variating the underlying set system or employing a computational key derivation that makes it possible to compress the information in $sk_u = (\mathcal{J}_u, \mathsf{K}_u)$, cf. section 2.3.3. The encryption overhead is also related to the efficacy of the algorithm that accompanies the set system and produces the revocation instruction given the set of users that need to be revoked. This is part of the challenge of the design of good set systems to be used in the above basic construction. Finally, the transmission overhead, i.e., the ciphertext length, is linear in number of subsets that is specified in the revocation instruction. We note that for the exclusive set systems we will see this will be a function of $r = |\mathsf{R}|$ and n where R is the set of revoked users. Intuitively the size of the broadcast pattern as a function of r depends on how dense the set system is.

There are two trivial instantiations of the above basic construction exhibiting a wide tradeoff between the efficiency parameters. In the first trivial instantiation, the set system consists merely of singletons for each receiver, i.e., $\Phi = \{\{1\}, \ldots, \{n\}\}$. Subsequently the encryption overhead would be linear in number of enabled receivers $n - r$. While this solution is optimal from the key-storage point of view, it wastes a lot of bandwidth and exhausts the broadcast center in the preparation of the transmission. In the second trivial instantiation, the set system Φ is the power set of the receiver population so that each receiver possesses the keys for all subsets it belongs to. In this case the ciphertext has minimal length (no larger than $r \cdot \log n + \lambda$ where λ is the ciphertext length of the underlying encryption scheme) but each receiver is required to store 2^{n-1} keys which is exponential in the number of users n. We will discuss combinatorial properties of the set systems that support efficient revocation in Section 2.4.2 and discuss a number of constructions that enjoy non-trivial tradeoffs in Section 2.5.

Next we prove the correctness of the template scheme following Definition 2.1.

Proposition 2.4. *Any broadcast encryption scheme that matches the template of Figure 2.2 satisfies correctness (cf. Definition 2.1).*

Proof. Let any $\mathcal{P} \in \mathcal{L}$ that encodes a set of revoked users R. Then we have that $\mathcal{P} = \{\mathsf{S}_{\mathsf{j}_1}, \ldots, \mathsf{S}_{\mathsf{j}_s}\}$ with $s \leq t$ and $[n] \setminus \cup_{i=1}^{s} \mathsf{S}_{\mathsf{j}_i} = \mathsf{R}$. It follows that any user $u \in [n] \setminus \mathsf{R}$ belongs to a set $\mathsf{S}_{\mathsf{j}_{i'}}$ for some $i' \in \{1, \ldots, s\}$. For user u, the input of the decryption function is some c such that $c \leftarrow \mathbf{Encrypt}(ek, m, \mathcal{P})$ as well as $sk_u = \langle \mathcal{J}_u, \mathsf{K}_u \rangle$ where $\mathcal{J}_u = \{\mathsf{j} \in \mathcal{J} \mid u \in \mathsf{S}_\mathsf{j}\}$ and $\mathsf{K}_u = \{k_\mathsf{j} \mid \mathsf{j} \in \mathcal{J}_u\}$. It follows that u will discover the index i' and apply the key $k_{\mathsf{j}_{i'}}$ to the i'-th component of the ciphertext c to recover the plaintext m correctly always.

2.2.1 Security

In this section we will focus on proving the security of the template construction of broadcast encryption based on exclusive set systems. We will follow the security modeling as expressed by Definition 2.2. The overall security of the scheme is based on the security of the underlying encryption scheme $\mathtt{E}_k, \mathtt{D}_k$ as indexed by a key k as well as the properties of the key assignment, i.e., the way that the keys of user u are sampled by the **KeyGen** algorithm.

Key Encapsulation Mechanisms.

We require the broadcast encryption scheme to be capable of transmitting a cryptographic key. We will ask that this same requirement should also be satisfied by the underlying cryptographic primitive $(\mathtt{E}, \mathtt{D})$, i.e., a cryptographic key should be encapsulated safely by the underlying encryption primitive.

We formalize the security requirement as the following game: for a random choice of the key k, the adversary $\mathcal{A}$ can adaptively choose plaintexts and see how $\mathtt{E}_k$ encrypts them; similarly, is capable of observing the output of decryption procedure $\mathtt{D}_k$. The adversary is challenged with a pair (c, m) for which it holds that either $c \leftarrow \mathtt{E}_k(m)$ or $c \leftarrow \mathtt{E}_k(m')$ where m, m' are selected randomly from the message space. The goal of the adversary is to distinguish between the two cases. This models a CCA1 type of encryption security, or what is known as a security against lunch-time attacks.

```
Experiment $\mathbf{Exp}^{kem}_{\mathcal{A}}$
    Select $k$ at random.
    $aux \leftarrow \mathcal{A}^{\mathtt{E}_k(),\mathtt{D}_k()}()$
    $m_0, m_1 \overset{R}{\leftarrow} \mathsf{M}$; $b \overset{R}{\leftarrow} \{0,1\}$; $c = \mathtt{E}_k(m_1)$
    $b' \leftarrow \mathcal{A}^{\mathtt{E}_k()}(aux, c, m_b)$
    return 1 if and only if $b = b'$;
```

Fig. 2.3. The security game of CCA1 secure key encapsulation for an encryption scheme.

Definition 2.5. *We say the symmetric encryption scheme* $(\mathtt{E}, \mathtt{D})$ *is* ε*-insecure if it holds that for any probabilistic polynomial-time* $\mathcal{A}$

$$\mathbf{Adv}^{kem}_{\mathcal{A}} = |\mathbf{Prob}[\mathbf{Exp}^{kem}_{\mathcal{A}} = 1] - \frac{1}{2}| \leq \varepsilon$$

Observe that the above requirement is weaker that one would typically expect from an encryption scheme that may be desired to protect the plaintext even if it is arbitrarily distributed. We note though that the key encapsulation security requirement will still force the encryption function to be probabilistic: indeed, in the deterministic case, the adversary can easily break security by encrypting m_b and testing the resulting ciphertext for equality to c. Further, since we are only interested in key encapsulation we can require the encryption oracle to only return encryptions of random plaintexts (as opposed to have them adaptively selected by the adversary).

Key-indistinguishability.

To ensure the security of the broadcast encryption scheme based on an exclusive set system, we have to perform the key-assignment in an appropriate way, i.e., a user should not be able to extract any information on a key of a subset that it does not belong to. Recall that for a set system Φ, where $\Phi = \{\mathsf{S}_\mathsf{j}\}_{\mathsf{j}\in\mathcal{J}}$, the **KeyGen** procedure generates a collection of keys $\{k_\mathsf{j}\}_{\mathsf{j}\in\mathcal{J}}$, one for each subset in Φ. Any user $u \in [n]$ is provided with a key assignment that is determined by the pair of sets $\langle \mathcal{J}_u, \mathsf{K}_u \rangle$ where $\mathcal{J}_u = \{\mathsf{j} \in \mathcal{J} \mid u \in \mathsf{S}_\mathsf{j}\}$ and $\mathsf{K}_u = \{k_\mathsf{j} \mid \mathsf{j} \in \mathcal{J}_u\}$. The key-indistinguishability property ensures that any coalition of users are not able to distinguish the key $k_{\mathsf{j}'}$ of a subset $\mathsf{S}_{\mathsf{j}'}$ they do not belong to from a random key. We will formalize the key-indistinguishability requirement through the following security-game.

```
EncryptOracle($m$, j)                         DecryptOracle($c$, j)
    retrieve $k_j$, $j_0$;                        retrieve $k_j$, $j_0$;
    return $c \leftarrow E_{k_j}(m)$;             return $D_{k_j}(c)$
--------------------------------------------------------------------------
Experiment $\mathbf{Exp}_{\mathcal{A}}^{key-ind}(1^n)$
    $b \xleftarrow{R} \{0,1\}$;        $j_0 \leftarrow \mathcal{A}(\Phi)$
    if $b = 0$ then $(\Phi, \{k_j\}_{j\in\mathcal{J}}) \leftarrow \mathbf{KeyGen}(1^n)$
    else $(\Phi, \{k_j\}_{j\in\mathcal{J}}) \leftarrow \mathbf{KeyGen}^{j_0}(1^n)$
    $b' \leftarrow \mathcal{A}^{\mathsf{EncryptOracle}(),\mathsf{DecryptOracle}()}(\langle \mathcal{J}_u, \mathsf{K}_u \rangle_{u \notin \mathsf{S}_{j_0}})$
    return 1 if and only if $b = b'$
```

Fig. 2.4. The security game for the key-indistinguishability property.

Definition 2.6. *We say that the broadcast encryption* BE *based on an exclusive set system satisfies the key indistinguishability property with distinguishing probability ε if there exists a family of key generation procedures $\{\mathbf{KeyGen}^\mathsf{j}\}_{\mathsf{j}\in\mathcal{J}}$ with the property that for all* j*,* **KeyGen**$^\mathsf{j}$ *selects the* j*-th key independently at random and it holds that for any probabilistic polynomial-time $\mathcal{A}$, $\mathbf{Adv}_{\mathcal{A}}^{key-ind}(1^n) = |\mathbf{Prob}[\mathbf{Exp}_{\mathcal{A}}^{key-ind}(1^n) = 1] - \frac{1}{2}| \leq \varepsilon$, where the experiment is defined as in figure 2.4.*

The definition of key indistinguishability suggests the following : the key generation algorithm **KeyGen** makes such a selection of keys that it is impossible for an adversary to distinguish with probability better than ε the key of subset S_j from a random key, even if it is given access to the actual keys of all users that do not belong to S_j as well as arbitrary encryption and decryption capability within the key system.

An easy way to satisfy the property of key indistinguishability is to have all keys of subsets selected randomly and independently from each other as is done in the broadcast encryption scheme $\mathtt{BE}^{\Phi}_{\mathtt{basic}}$. Indeed, we have the following proposition.

Proposition 2.7. *The basic broadcast encryption scheme* $\mathtt{BE}^{\Phi}_{\mathtt{basic}}$ *(refer to Figure 2.2 and comments below) satisfies the key indistinguishability property with distinguishing probability* 0.

Proof. It is easy to see that the choice k_{j_0} by the **KeyGen** algorithm is identically distributed to the random selection of k_1 from K. Based on this it is easy to derive that the scheme $\mathtt{BE}^{\Phi}_{\mathtt{basic}}$ satisfies key indistinguishability.

We, now, come to the point we can state the security theorem for the template broadcast encryption as defined in Figure 2.2. The requirements for security are the key-indistinguishability property and the use of an encryption that is suitable for key encapsulation.

Theorem 2.8. *Consider a broadcast encryption scheme* $\mathtt{BE}$ *that fits the template of Figure 2.2 over an* (n, r, t)*-exclusive set system* Φ *and satisfies (1) the key indistinguishability property with distinguishing probability* ε_1*, (2) its underlying encryption scheme* $(\mathtt{E}, \mathtt{D})$ *is* ε_2*-insecure in the sense of Definition 2.5. Then, the broadcast encryption scheme* $\mathtt{BE}$ *is* ε*-insecure in the sense of Definition 2.2 where* $\varepsilon \leq 2t \cdot |\Phi| \cdot (2\varepsilon_1 + \varepsilon_2)$.

Proof. We will prove the above argument by structuring the proof as a sequence of indistinguishable games all operating over the same underlying probability space. Starting from the actual attack scenario, we consider a sequence of hypothetical games. In each game, the adversary's view is obtained in different ways, but the probability of success will be related in a predictable fashion. Let us start writing the original game $\mathbf{Exp}_0 = \mathbf{Exp}^{rev}_{\mathcal{A}}(n)$ explicitly in Figure 2.5:
Experiment $\mathbf{Exp}^{v}_{1}$. This experiment, for $v = 0, \ldots, t$, is identical to $\mathbf{Exp}_0$, with two slight modifications. The first modification is in the encryption on line 5. The experiment of type v, $\mathbf{Exp}^{v}_{1}$, is one where the encryption is computed so that the first v subset keys are over a random plaintext while the remaining subsets encode the correct message. More specifically, the 5th line in the experiment would look as follows :

$$c = \langle \mathsf{j}_1, \ldots, \mathsf{j}_s, E_{k_{\mathsf{j}_1}}(R_1), \ldots, E_{k_{\mathsf{j}_v}}(R_v), E_{k_{\mathsf{j}_{v+1}}}(m_1), \ldots, E_{k_{\mathsf{j}_s}}(m_1) \rangle$$

where R_i is a random string of the same length as the message m_1 for $i = 1, \ldots, v$. Note that if $v > s$ all plaintexts are selected independently at random.

The second modification of the experiment $\mathbf{Exp}^{v}_{1}$ is the choice of a uniformly random variable $w \in \{1, \ldots, |\Phi|\}$. Consider an enumeration for the key encodings, i.e. $\mathcal{J} = \{\mathsf{j}[1], \ldots, \mathsf{j}[|\Phi|]\}$. The experiment $\mathbf{Exp}^{1}_{v}$ is modified in the

```
EncryptOracle($m, \psi$)              | DecryptOracle($c, u$)              | CorruptOracle($u$)
    retrieve $ek$;                    |     retrieve $sk_u$;               |     $\mathsf{T} \leftarrow \mathsf{T} \cup \{u\}$
    $c \leftarrow$ Encrypt($ek, m, \psi$); |     return Decrypt($c, sk_u$);  |     retrieve $sk_u$;
    return $c$;                       |                                    |     return $sk_u$;
Experiment $\mathbf{Exp}_{\mathcal{A}}^{rev}(1^n)$
    $\langle(\Phi, \{k_{\mathsf{j}}\}_{\mathsf{j}\in\mathcal{J}}), (\mathcal{J}_1, \mathsf{K}_1), \ldots, (\mathcal{J}_n, \mathsf{K}_n)\rangle \leftarrow \mathbf{KeyGen}(1^n)$; $\mathsf{T} \leftarrow \emptyset$
    $ek = (\Phi, \{k_{\mathsf{j}}\}_{\mathsf{j}\in\mathcal{J}})$; $sk_u = (\mathcal{J}_u, \mathsf{K}_u)$ for $u = 1, \ldots, n$
    $\psi^* \leftarrow \mathcal{A}^{\mathsf{EncryptOracle}(),\mathsf{DecryptOracle}(),\mathsf{CorruptOracle}()}(\Phi)$
    $m_0, m_1 \overset{R}{\leftarrow} \mathsf{M}$; $b \overset{R}{\leftarrow} \{0,1\}$;
    $c = \langle \mathsf{j}_1, \ldots, \mathsf{j}_s, \mathsf{E}_{k_{\mathsf{j}_1}}(m_1), \ldots, \mathsf{E}_{k_{\mathsf{j}_s}}(m_1)\rangle \leftarrow \mathbf{Encrypt}(ek, m_1, \psi^*)$
    $b' \leftarrow \mathcal{A}^{\mathsf{EncryptOracle}()}(\{sk_i\}_{i\in\mathsf{T}}, m_b, c)$
    return 1 if and only if $b = b'$ and
                    $\psi^*$ is an instruction that excludes all members of $\mathsf{T}$.
```

Fig. 2.5. The initial security game $\mathbf{Exp}_0$.

output line to behave as follows: if it happens that the index of $(v+1)$-th subset in the challenge ciphertext in line 5 does not match $\mathsf{j}[w]$ then a random coin flip is returned as the output of the experiment. If it happens that $v+1 > s$ the s-th subset is used in the test.

Let p_1^v be the probability that experiment $\mathbf{Exp}_1^v$ returns 1 and similarly let p_0 be the probability that the experiment $\mathbf{Exp}_0$ returns 1. Observe that in the case $v \geq s$, the sequence of encryptions no longer contains any information on b, and the same is true for all other information given to the adversary. It follows that the adversary's view is independent of b and therefore it is easy to see that $p_1^t = \frac{1}{2}$.

On the other hand, let us assume that the adversary has advantage of at least ε in being succesful on the original experiment $\mathbf{Exp}_0$, i.e., without loss of generality, $p_0 \geq 1/2 + \varepsilon$. This suggests that

$$p_1^0 \geq 1/2 + \frac{\varepsilon}{|\Phi|}$$

It follows there is a gap of $\varepsilon/|\Phi|$ between p_1^0, p_1^t and by applying the triangular inequality we obtain that there must exist some $0 < v' \leq t$ such that

$$|\, p_1^{v'-1} - p_1^{v'} \,| \geq \frac{\varepsilon}{t \cdot |\Phi|}$$

Experiment $\mathbf{Exp}_2^v$. This experiment, for $v = 0, \ldots, t$ is identical to $\mathbf{Exp}_1^v$, except that one of the keys is replaced with a random key, i.e., the way the normal key generation algorithm **KeyGen** works in line 1 of the game is modified. Specifically, in the modified game we employ the alternative key generation $\mathbf{KeyGen}^{\mathsf{j}[w]}$. In case a user owning this key gets corrupted the experiment fails and returns a random coin flip. We denote by p_2^v the probability the experiment returns 1.

We claim that for any $v = 0, \ldots, t$

$$|p_1^v - p_2^v| \leq 2\varepsilon_1$$

We next prove the claim. Consider the following key indistinguishability adversary for some parameter v. $\mathcal{B}_1$ follows the structure of the experiments $\mathbf{Exp}_1^v, \mathbf{Exp}_2^v$. It first provides as output the index $\mathsf{j}[w]$; this would be the key that will be attacked for key indistinguishability. Following the game-based definition, $\mathcal{B}_1$ receives the keys of all users that are outside of $\mathsf{S}_{\mathsf{j}[w]}$ as well as it is given oracle access to encryption and decryption for any key j that corresponds to a subset of $\mathsf{S}_{\mathsf{j}[w]}$. $\mathcal{B}_1$ proceeds to simulate the challenger and the adversary $\mathcal{A}$ as in $\mathbf{Exp}_2^v$. Based on the key material that is available to $\mathcal{B}_1$, it can perform the simulation without difficulty except when the adversary decides to corrupt a user $u \in \mathsf{S}_{\mathsf{j}[w]}$. In such case $\mathcal{B}_1$ fails and returns 0. This completes the description of $\mathcal{B}_1$. Based on our assumption about key indistinguishability we know that $|\mathbf{Prob}[\mathbf{Exp}_{\mathcal{B}_1}^{key-ind}(1^n)] - 1/2| \leq \varepsilon_1$.

We observe that $\mathbf{Prob}[\mathbf{Exp}_{\mathcal{B}_1}^{key-ind}(1^n)|b = 0] = 1 - p_1^v$. This holds as the only seeming difference between the way $\mathcal{B}_1$ operates in the conditional space $b = 0$ and the experiment $\mathbf{Exp}_1^v$ is the event when a party in $\mathsf{S}_{\mathsf{j}[w]}$ is corrupted. In this case $\mathcal{B}_1$ returns 0. For the same event in the experiment $\mathbf{Exp}_1^v$ we observe that the end condition will force also a 0 output. In a similar fashion it holds that $\mathbf{Prob}[\mathbf{Exp}_{\mathcal{B}_1}^{key-ind}(1^n)|b = 1] = p_2^v$. Based on this we obtain the proof of the claim.

We next claim that for any $v = 1, \ldots, t$,

$$|p_2^{v-1} - p_2^v| \leq 2\varepsilon_2$$

We prove the claim. Consider the following CCA1 key encapsulation adversary with a parameter v. $\mathcal{B}_2$ follows the structure of the experiment $\mathbf{Exp}_2^v$. To break the security claim of the underlying symmetric encryption scheme $(\mathtt{E}, \mathtt{D})$, $\mathcal{B}_2$, is given access to encryption-decryption oracle of the primitive. $\mathcal{B}_2$ has also access to the decryption/encryption under a key k that is unknown to her. She is then challenged with plaintext-ciphertext pair (c, m_1) for which either $c = \mathtt{E}_k(m_1)$ or $c = \mathtt{E}_k(m)$ for some random message m that has the same length with m_1. She is asked to test if the pair is a correct plaintext-ciphertext pair.

$\mathcal{B}_2$ will simulate the experiment $\mathbf{Exp}_2^v$. $\mathcal{B}_2$ will focus its attack on the key of the $\mathsf{j}[w]$-th subset. In the initial stage it will simulate $\mathcal{A}$ as in the experiment $\mathbf{Exp}_2^v$ by answering all queries except those that involve the $\mathsf{j}[w]$-th subset that will be answered by the encryption and decryption oracle available to $\mathcal{B}_2$. In the second stage, $\mathcal{B}_2$ will be given the challenge (c, m) where either m is the proper plaintext of c or it is randomly selected. $\mathcal{B}_2$ will prepare a challenge for $\mathcal{A}$ in the security game $\mathbf{Exp}_2^v$ by setting the v-th location of the challenge ciphertext with c and the remaining positions from $v+1, \ldots, s$ with the appropriate encryptions of m as dictated in the revocation instruction ψ^*

that is selected by $\mathcal{A}$ in its first stage. The simulation of the second stage of $\mathcal{A}$ within experiment $\mathbf{Exp}_2^v$ can be carried out by $\mathcal{B}_2$ without problem by resorting to its encryption oracle whenever needed.

Based on the assumption on the underlying encryption scheme we have that $\mathbf{Prob}[\mathbf{Exp}_{\mathcal{B}_2}^{kem}(1^n)] - 1/2| \leq \varepsilon_2$. We observe that in the key encapsulation attack of $\mathcal{B}_2$ in the conditional space $b = 0$, the adversary $\mathcal{B}_2^{kem}(1^n)$ operates identically to the experiment $\mathbf{Exp}_2^{v-1}$ while in the conditional space $b = 1$ it operates identically to the experiment $\mathbf{Exp}_2^v$. Based on this derive the proof of the claim.

We are now ready to give the proof of the theorem. First recall from our first claim that there is a $v' \in \{1, \ldots, t\}$ such that

$$| p_1^{v'-1} - p_1^{v'} | \geq \frac{\varepsilon}{t \cdot |\Phi|}$$

From our second claim we know that $|p_1^{v'} - p_2^{v'}| \leq 2\varepsilon_1$ as well as $|p_1^{v'-1} - p_2^{v'-1}| \leq 2\varepsilon_1$. We know that for any a, b, c, d, e, $|a - b| \geq d$ and $|b - c| \leq e$ implies $|a - c| \geq d - e$. By combining this with the above facts we have $|p_2^{v'-1} - p_2^{v'}| \geq \varepsilon/(t \cdot |\Phi|) - 4\varepsilon_1$. Now based on our third claim we have that $|p_2^{v'-1} - p_2^{v'}| \leq 2\varepsilon_2$. From this we obtain $\varepsilon \leq 2t \cdot |\Phi| \cdot (2\varepsilon_1 + \varepsilon_2)$ which completes the proof.

2.2.2 The Subset Cover Framework

Among set systems, *fully-exclusive set systems* are of interest due to their support for revocation for any number of receivers. The subset cover framework class of fully-exclusive set systems to be used in broadcast encryption template given in Figure 2.2. We can list the requirements of a set system Φ to be considered in the framework as follows:

1. The set system Φ is fully-exclusive, i.e. Φ is (n, n, t)-exclusive where Φ is defined over a set of size n, and t is a function of the size r of the revoked set.
2. The set system is accompanied with an efficient revocation algorithm **Revoke** that produces the indices $\mathsf{j}_i, i = 1, \ldots, s$ of the subsets that cover the set of enabled users $[n] \setminus \mathsf{R}$. We require the efficiency in the following sense: (i) The time required to compute the pattern is efficient, i.e. polylog in the number of receivers (ii) The output pattern consists of minimal number of subsets that is a function of n and $r = |\mathsf{R}|$.
3. The set system Φ supports "bifurcation property": it should be possible to split any subset $\mathsf{S} \in \Phi$ into two roughly equal sets, i.e. that there exists $\mathsf{S}_1, \mathsf{S}_2 \in \Phi$ such that $\mathsf{S} = \mathsf{S}_1 \cup \mathsf{S}_2$ and $|\mathsf{S}_1|/|\mathsf{S}_2|$ is bounded above from a constant. In some cases, although the first split might not hold the above property, it is also acceptable if the relative size of the larger subset in consecutive split operations converges to the bound quickly.

The last property is required to defend against traitors leaking their key materials which we will discuss more in Chapter 4.

In general, we will represent a subset-cover scheme by a pair $\langle \Phi, \mathbf{Revoke}\rangle$ where $\Phi = \{\mathsf{S_j}\}_{\mathsf{j}\in\mathcal{J}}$ is a fully-exclusive set system defined over $[n]$ that satisfies the above properties. Later in this chapter, we will start designing fully-exclusive set systems that are accompanied with an efficient revocation algorithm. We found imperative to see the set system as a partial order over a 'subset' relation. This will enable us to define a generic revocation algorithm for set systems that satisfy the necessary requirements we put forth. Section 2.4.2 will elaborate on the actual algorithm after we build the necessary notions and definitions in the following section.

2.3 The Key-Poset Framework for Broadcast Encryption

In this section we will introduce an algebraic perspective for the study of broadcast encryption based on exclusive set systems that we call the Key-Poset (KP) framework. Recall that every key in the construction template for broadcast encryption in Figure 2.2 can be viewed as a set of users (the users that own it or can derive it) and thus the set of keys forms a partially ordered set (or poset) which is a sub-poset of the powerset of $\{1, \ldots, N\}$, the set of all users. This is based on the fact that if a user owns (or can derive) a key corresponding to a set of users S it should own (or be able to derive) all keys that dominate S in the key poset.

The rest of the section is structured as follows. In section 2.3.1 we discuss some basic elements of partial order theory that we need. Next we introduce a more careful formalization of the representation of exclusive set families in section 2.3.2. Finally we discuss how the **KeyGen** algorithm can be modified to compress the information in the secret key-material sk_u available to each user $u \in [n]$. We will provide a generic key compression technique over the key-poset in Section 2.3.3 that satisfies the key indistinguishability property described in Definition 2.6.

2.3.1 Viewing Set Systems as Partial Orders

A partial order set (or poset) is a set Φ equipped with a relation $\leq$ that is reflexive, antisymmetric and transitive. A nonempty subset A of a partially ordered set $(\Phi, \leq)$ is called a *directed set* if for any two elements $a, b \in \mathsf{A}$, there exists c in A such that $a \leq c$ and $b \leq c$. It is called a *lower set* if for every $x \in \mathsf{A}$, $y \leq x$ implies that y is in A. An *atom* in a poset Φ is an element that is minimal among all elements. A poset would be called *atomistic* if $x \not\leq y$ implies that there is an atom a such that $a \leq x$ and $a \not\leq y$.

A nonempty subset I of a partially ordered set $(\Phi, \leq)$ is called an *ideal* if I is lower and directed. The smallest ideal that contains a given element p if it exists is called a principal ideal and p is said to be a principal element of the ideal in this case. The principal ideal $\downarrow p$ for a principal p is thus given by $\downarrow p = \{x \in P \mid x \leq p\}$.

The dual notion of ideal, the one obtained in the reverse partial order, is called a *filter*. We define $\mathsf{F}(x)$ as the set of elements $\{a \in \Phi : x \leq a\}$. It is not hard to prove that $\mathsf{F}(x)$ constitutes a filter. We call $\mathsf{F}(x)$ an *atomic filter* if x is an atom. We also denote by P_x the complement of $\mathsf{F}(x)$ in $(\Phi, \leq)$, i.e., $\mathsf{P}_x = \Phi \setminus \mathsf{F}(x)$.

A useful observation is the following :

Lemma 2.9. *For any atom u in a poset Φ it holds that P_u is a lower set.*

Proof. Let $x, y \in \Phi$ with $y \in \mathsf{P}_u$ and $x \leq y$. Suppose that $x \notin \mathsf{P}_u$, i.e., it must be the case that $x \in \mathsf{F}(u)$, i.e., $u \leq x$. By transitivity we have that $u \leq y$ and as a result $y \in \mathsf{F}(u)$, a contradiction. It follows $x \in \mathsf{P}_u$. ■

The following definition will be an essential tool in our exposition.

Definition 2.10. *Given a nonempty subset A of a finite atomistic partially ordered set $(\Phi, \leq)$ that is a lower set, we say $\langle \mathsf{M}_1, \mathsf{M}_2, \ldots, \mathsf{M}_k \rangle$ is a* lower-maximal partition *of A if*

1. *$\emptyset \neq \mathsf{M}_i \subseteq \mathsf{A}$ is a lower set for $i = 1, \ldots, k$.*
2. *for all i, i', $(i \neq i') \rightarrow \mathsf{M}_i \nsubseteq \mathsf{M}_{i'}$.*
3. *M_i is maximal with respect to A, i.e., for all $a \in \mathsf{M}_i$ if $\exists b \in \mathsf{A}$ s.t $a \leq b$, then $b \in \mathsf{M}_i$.*
4. *k is the largest integer such that all the above hold.*

The order of a lower set A is defined as the size of its lower-maximal partition. We denote the order by $\mathsf{ord}(\mathsf{A})$.

We next prove some properties of the lower-maximal partition of a lower set; we start with a preparatory lemma.

Lemma 2.11. *Let A be a lower set. The set of all subsets of A that are lower and maximal with respect to A is closed under intersection and set subtraction. Specifically, if $\mathsf{M}_1, \mathsf{M}_2 \subseteq \mathsf{A}$ are two lower sets that are maximal with respect to A then the sets $\mathsf{M}_1 \cap \mathsf{M}_2$, $\mathsf{M}_1 \setminus \mathsf{M}_2$ are also lower and maximal with respect to A.*

Proof. Consider the set $\mathsf{M}_1 \cap \mathsf{M}_2$. It is easy to see that it is a lower set as an intersection of lower sets. Further, if there is some $b \in \mathsf{A}$ with $a \in \mathsf{M}_1 \cap \mathsf{M}_2$ and $a \leq b$ it follows that b must belong to both M_1 and M_2 due to the maximality of those sets with respect to A. As a result $\mathsf{M}_1 \cap \mathsf{M}_2$ is also maximal with respect to A. Now consider $\mathsf{D} = \mathsf{M}_1 \setminus \mathsf{M}_2$. Let $b \in \mathsf{D}$ and $a \in \Phi$ such that $a \leq b$. We will show that $a \in \mathsf{D}$. First observe that due to the fact that M_1 is lower we have that $a \in \mathsf{M}_1$. Second, if $a \in \mathsf{M}_2$ then it follows that $b \in \mathsf{M}_2$ by the maximality of M_2 with respect to A. This is a contradiction as a result $a \in \mathsf{D}$. This proves that D is a lower set.

We next show that D is maximal with respect to A. Let $a \in \mathsf{D}$ and some $b \in \mathsf{A}$ with $a \leq b$. We will show that $b \in \mathsf{D}$. First observe that $b \in \mathsf{M}_1$ due to the maximality of M_1. On the other hand, if it is the case that $b \in \mathsf{M}_2$ we would have that $a \in \mathsf{M}_2$ due to the fact that M_2 is lower, a contradiction. It follows that that $b \in \mathsf{D}$ and as a result D is maximal with respect to A. ■

Proposition 2.12. *For any lower set* A*, the sets* $\mathsf{M}_1, \ldots, \mathsf{M}_k$ *in the lower-maximal partition of* A *form a* partition *of* A*, i.e., they are disjoint and their union covers* A*.*

Proof. Regarding disjointness, suppose, without loss of generality, that $\mathsf{M}_1 \cap \mathsf{M}_2$ is a non-empty proper subset of M_1 and M_2. Consider, now, the collection $(\mathsf{M}_1 \cap \mathsf{M}_2), \mathsf{M}_1 \setminus \mathsf{M}_2, \mathsf{M}_2 \setminus \mathsf{M}_1, \mathsf{M}_3 \setminus (\mathsf{M}_1 \cap \mathsf{M}_2), \ldots, \mathsf{M}_k \setminus (\mathsf{M}_1 \cap \mathsf{M}_2)$; observe that all these sets are lower and maximal with respect to A in the light of lemma 2.11. It is easy to see that this collection of subsets also satisfies the requirements of the lower-maximal partition and it is a contradiction as it has $k+1$ elements.

Consider now $a \in \mathsf{A}$ an atom. Suppose that no set in the lower-maximal partition $\mathsf{M}_1, \ldots, \mathsf{M}_k$ contains it. We will derive a contradiction.

We define a *zig-zag* path towards a to be a vector of elements $\langle x_1, \ldots, x_t \rangle$ from A such that $x_1 = a$, and $x_1 \leq x_2, x_2 \geq x_3, x_3 \leq x_4, \ldots$. Consider the set $\mathsf{M} = \{x \mid \exists \text{ zig-zag path } \langle x_1, \ldots, x_t \rangle : (x_1 = a) \wedge (x_t = x)\}$. This set is lower : suppose $x \leq y$ with $y \in \mathsf{M}$. Note that $x \in \mathsf{A}$ due to the fact that A is lower. Consider the zig-zag path, $\langle y_1, \ldots, y_t \rangle$ with $y_1 = a$ and $y_t = y$. If $y_{t-1} \leq y_t = y$ then x can extend the zig-zag path and thus $x \in \mathsf{M}$. If $y_{t-1} \geq y_t$ we have by transitivity that also $y_{t-1} \geq x$ and thus $\langle y_1, \ldots, y_{t-1}, x \rangle$ is a zig-zag path that shows $x \in \mathsf{M}$.

Consider some $y \in \mathsf{A}$ with $x \leq y$ and $x \in \mathsf{M}$. As before, given the zig-zag path $\langle x_1, \ldots, x_t = x \rangle$ we have that if $x_{t-1} \geq x$ we can extend the zig-zag path to y. On the other hand, if $x_{t-1} \leq x$ we have by transitivity that $x_{t-1} \leq y$ and again we obtain a zig-zag path to y. This shows that $y \in \mathsf{M}$ and that M is maximal with respect to A.

Obviously M is non-empty (it contains at least a) and it is a subset of A. Moreover, it cannot be the subset of any of $\mathsf{M}_1, \ldots, \mathsf{M}_k$ (that as none of these sets contain a). Suppose now that for some i it holds that $\mathsf{M}_i \subseteq \mathsf{M}$. This means that for any $b \in \mathsf{M}_i$ we can find a zig-zag path to a, i.e., there is $\langle x_1, \ldots, x_t \rangle$ with $x_1 = a$ and $x_t = b$. Suppose that $b = x_t \geq x_{t-1}$. Due to the fact that M_i is lower we obtain that $x_{t-1} \in \mathsf{M}_i$. On the other hand, if $b = x_t \leq x_{t-1}$ we have that due to the fact that M_i is maximal with respect to A it holds that $x_{t-1} \in \mathsf{M}_i$. Repeating this argument along the zig-zag path shows that $a \in \mathsf{M}_i$ which is a contradiction. It follows that it cannot be the case that M_i is covered with M.

Based on the above we conclude that $\mathsf{M}, \mathsf{M}_1, \ldots, \mathsf{M}_k$ satisfies the properties of a lower maximal partition with length longer than k, a contradiction. Therefore all the atoms of A are included in the union of $\mathsf{M}_1, \ldots, \mathsf{M}_k$. ■

The lower-maximal partition of a lower set is uniquely determined as we show next.

Theorem 2.13. *Any lower set* A *of poset* $(\Phi, \leq)$ *has a unique lower-maximal partition.*

Proof. Theorem 2.13 Suppose that A has two different lower-maximal partitions, i.e. $\langle \mathsf{M}_1, \ldots, \mathsf{M}_k \rangle = \langle \mathsf{M}'_1, \ldots, \mathsf{M}'_{k'} \rangle$ are satisfying the conditions listed in Definition 2.10. Note first that it must be $k = k'$ due to the fact that k, k' are both the largest integers that satisfy the first three conditions, thus the order of A is unique.

Suppose now that there is a set M'_j that is not equal to any of $\mathsf{M}_1, \ldots, \mathsf{M}_k$. Without loss of generality let us consider $j = 1$. Consider the sets $\mathsf{L}_i = \mathsf{M}_i \cap \mathsf{M}'_1$ and $\mathsf{D}_i = \mathsf{M}'_1 \setminus \mathsf{M}_i$. If there is some i for which it holds that both $\mathsf{L}_i, \mathsf{D}_i$ are non empty then we derive a contradiction as we could form a lower-maximal partition longer than k. Indeed, say this holds for $i = 1$, then the collection $\mathsf{L}_1, \mathsf{D}_1, \mathsf{M}'_2 \setminus \mathsf{L}_1, \ldots, \mathsf{M}'_k \setminus \mathsf{L}_1$ satisfies the properties (1-3) of a lower-maximal partition and has $k + 1$ subsets.

It follows that for all i, either there is no intersection between M_i and M'_1 or M'_1 is a strict subset of M_i. Suppose that there is an i for which M'_1 is a strict subset of M_i. This means that there is some atom $a \in \mathsf{M}_i$ for which it holds that it is not in M'_1. Indeed, for the sake of contradiction, suppose that all atoms of M_i are in M'_1. Then if $x \in \mathsf{M}_i$, by the fact that M_i is lower, we can build a chain connecting x to an atom of M_i, which is also an atom of M'_1 and by the fact that M'_1 is maximal with respect to A we have that $x \in \mathsf{M}'_1$, i.e., $\mathsf{M}'_1 = \mathsf{M}_i$, a contradiction to the fact that the former is a strict subset. We derive from this that there is some other set M'_j that includes those atoms of M_i that are excluded from M'_1. Then, it follows that $\mathsf{M}'_j \cap \mathsf{M}_i$ is a strict non-empty subset of M_i and we can apply the same reasoning as before to derive a longer lower-maximal partition. This is a contradiction and as a result it cannot be that there is some i for which M'_1 is a strict subset of M_i. It follows that for all i, M'_1 has no intersection with M_i which is also a contradiction as some overlap is guarranteed by the fact that M'_1 is a non-empty lower set that must contain at least one atom of A. This contradiction is against our initial assumption that there is a set among $\mathsf{M}'_1, \ldots \mathsf{M}'_k$ that is not equal to any of $\mathsf{M}_1, \ldots, \mathsf{M}_k$. From this we derive the equality of the two lower-maximal partitions. ■

In the context of broadcast encryption we will restrict our attention to finite and atomistic posets. In this domain it is easy to see that all ideals have a single maximal element and thus are principal. Further observe that for any ideal I it holds that $\mathsf{ord}(\mathsf{I}) = 1$. Indeed, if it happens that we have a lower maximal partition $\mathsf{M}_1, \ldots, \mathsf{M}_k$ of I with $k > 1$, for any two elements $a \in \mathsf{M}_1, b \in \mathsf{M}_2$ it holds that $a \leq m$ and $b \leq m$ where m is the maximal element of I. Due to the maximality of $\mathsf{M}_1, \mathsf{M}_2$ with respect to I we also obtain that $m \in \mathsf{M}_1 \cap \mathsf{M}_2$. But due to the fact that M_2 is lower it also holds that

$a \in \mathsf{M}_2$. It follows $\mathsf{M}_1 \subseteq \mathsf{M}_2$ a contradiction. It follows that I has as a lower maximal partition just itself.

It is interesting to note though that $\mathsf{ord}(\mathsf{A}) = 1$ is not a sufficient condition to make A an ideal. This is due to the fact that the directed property is not implied by $\mathsf{ord}(\mathsf{A}) = 1$.

In the definition below we define the set of maximal elements of a subset A of a poset.

Definition 2.14. *Given a nonempty subset* A *of a partially ordered set* $(\Phi, \leq)$ *we say* $\langle \mathtt{m}_1, \ldots, \mathtt{m}_k \rangle$ *is the set of* maximal-elements *of* A *if the following holds:*

1. $\mathtt{m}_i \in \mathsf{A}$ *for* $i = 1, \ldots, k$.
2. *For any* $\mathtt{a} \in \mathsf{A}, i \in \{1, \ldots, k\}$*, it holds that either* $\mathtt{a} \leq \mathtt{m}_i$ *or it is not possible to compare* $\mathtt{a}$ *with* $\mathtt{m}_i$.

We will denote the number of maximal-subsets of a lower set A *by* $\mathsf{maxnum}(\mathsf{A})$ *(the maximal-order of* A*).*

We next prove an important property elaborating on the relations between the maximal elements of a lower set with an order 1 : a maximal element in such a lower set can not be disjoint from other maximal elements.

Lemma 2.15. *Given that* $\langle \mathtt{m}_1, \ldots, \mathtt{m}_k \rangle$ *is the set of maximal elements of a lower set* A *that satisfies* $\mathsf{ord}(\mathsf{A}) = 1$*, for each* $\mathtt{m}_i$ *there exists another maximal element* $\mathtt{m}_j$ *that they share a common atom, i.e. there exists an atom* $a \in \mathsf{A}$ *such that* $a \leq \mathtt{m}_i$ *and* $a \leq \mathtt{m}_j$ *hold.*

Proof. of Lemma 2.15: Consider now a maximal element $\mathtt{m}_i$ and suppose that $\mathtt{m}_i$ does not share a common atom with any other maximal element. We will derive a contradiction.

We denote the set of atoms dominated by $\mathtt{m}_i$ by A_i. Consider the set $\mathsf{D} = \{\mathtt{b} \in \mathsf{A} \mid \exists a \in A_i \text{ s.t. } a \leq \mathtt{b}\}$.

The set D is lower: suppose $x \leq y$ with $y \in \mathsf{D}$. We will first prove that $y \leq \mathtt{m}_i$. Suppose the converse is true, $y \not\leq m_i$ and recall that atomicity implies that there exists an atom a with $a \leq y$ and $a \not\leq \mathtt{m}_i$. We know that there exists a maximal element $\mathtt{m}_j$ such that $y \leq \mathtt{m}_j$. Hence, we obtain $\mathtt{m}_j \in \mathsf{D}$ which is a contradiction with the fact that $\mathtt{m}_i$ shares no common element with any other maximal element. Given now that $y \leq m_i$, we have by transitivity that also $x \leq \mathtt{m}_i$. This implies that there is an atom $a \in A_i$ such that $a \leq x$, i.e. $x \in \mathsf{D}$.

We next show that the set D is maximal with respect to A. Consider some $y \in \mathsf{A}$ with $x \leq y$ and $x \in \mathsf{D}$. Since $x \in \mathsf{D}$ then there exists an atom $a \in A_i$ with $a \leq x$ and by transitivity we have $a \leq y$. This shows that $y \in \mathsf{D}$ and thus D is maximal with respect to A.

Now it holds that trivially, A is lower and maximal with respect to itself and thus by Lemma 2.11 we have that $\mathsf{A} \setminus \mathsf{D}$ is a lower set and maximal with respect to set A. Then, the pair $\langle \mathsf{D}, \mathsf{A} \setminus \mathsf{D} \rangle$ satisfies the properties (1-3) of a lower maximal partition which contradicts with the fact that $\mathsf{ord}(\mathsf{A}) = 1$. ■

The following proposition sets the stage for our investigation of fully exclusive set systems as posets.

Proposition 2.16. *Any fully-exclusive set system $\Phi \subseteq 2^{[n]}$ forms a finite atomistic poset ordered by subset inclusion whose ideals are all principal.*

Proof. Proposition 2.16 Suppose that $\Phi = \{\mathsf{S}_\mathsf{j}\}_{\mathsf{j}\in\mathcal{J}}$. Φ indeed forms a poset ordered by subset inclusion as the reflexive, antisymmetric and transitive properties hold for subset inclusion. This poset would be atomistic due to the fact that Φ is a fully-exclusive set system, i.e. any user, itself, should be a valid subset in the set system which is equivalent to a corresponding atom. Consider any lower-directed set (an ideal) I of Φ, this ideal would have a single maximal element. This maximal element, itself, is a subset S in Φ, and the ideal I would be the smallest ideal that contains S. Hence, any ideal I turns out to be a principal ideal. ■

A useful observation we make finally is that in a fully exclusive set system disjoint ideals are made out of disjoint sets of atoms.

Lemma 2.17. *Consider any two disjoint ideals $\mathsf{I}_1, \mathsf{I}_2$ of a fully exclusive set system Φ. It holds for any $\mathsf{S}_1 \in \mathsf{I}_1$ and $\mathsf{S}_2 \in \mathsf{I}_2$ that $\mathsf{S}_1 \cap \mathsf{S}_2 = \emptyset$.*

Proof. Lemma 2.17 Suppose the converse; i.e. $u \in \mathsf{S}_1 \cap \mathsf{S}_2$. Because Φ forms a finite atomistic poset, there exists an atom corresponding to the element u. That atom is subset of both S_1 and S_2, thus, it is an element of both ideals due to the lower property. This contradicts with the definition of being disjoint. ■

In general we may use the notation Φ to refer the corresponding poset $(\Phi, \subseteq)$ interchangeably depending on the context. Similarly, we use the term "subset" in a set system interchangeably with the term "node" or member in the poset. When drawing diagrams of posets, we will always draw the transitive-reduction of the key-poset unless stated otherwise.

2.3.2 Computational Specification of Set Systems

So far we have avoided explaining what is the exact data structure that represents a set system Φ. While it is possible to think of a representation of Φ as a collection of sets this is not the most efficient way that an algorithm can interact with Φ. Representing an explicit data structure that packages Φ will help in the design as well as the exact complexity analysis of the algorithms that operate on a set system.

The formal specification of a set system Φ that we lay out below is comprised by six basic functions. Henceforth it will be possible to design algorithms that employ these basic functions in black-box fashion and hence they can be "family-independent."

Definition 2.18. *A collection of families Φ parameterized by n is defined by six algorithms* $(\mathbf{tst}, \mathbf{mbr}, \mathbf{slo}, \mathbf{sbs}, \mathbf{cvr}, \mathbf{spt})$ *and a constant* $\mathbf{cvm}$*, a finite alphabet and a length function* $\mathsf{l}(\cdot)$*. Each element of Φ is encoded over the alphabet with length of at most $\mathsf{l}(n)$ where n is the number of users, and we denote the set of encodings by $\mathcal{J}$. Below the input s is thought to be a string of length at most $\mathsf{l}(N)$. The algorithms are defined as follows.*

1. **tst***: (Encoding Testing); given s, returns* 1 *if s is a valid encoding of a member of Φ or 0 otherwise, i.e. returns* 1 *if $s \in \mathcal{J}$ or* 0 *otherwise.*
2. **mbr***: (Membership Testing); given (s, u) returns* 1 *if u belongs to the set encoded by s,* 0 *otherwise.*
3. **slo***: (Outside Selection); given a list of $\{s_1, s_2, \ldots s_r\} \subseteq \Phi$ it returns an element u outside of the union of the sets encoded by $\{s_1, s_2, \ldots s_r\}$ or $\perp$ if no such element exists.*
4. **sbs***: (Inclusion Testing); given (s, s') returns* 1 *if and only if s is a subset of s'.*
5. **cvr***: (Parent Finding); given (s, i) for $i \leq$* $\mathbf{cvm}$*, it returns a parent s' of s, i.e., an s' such that $sbs(s, s') = 1$ and there exists no s'' such that $\mathbf{sbs}(s, s'') = \mathbf{sbs}(s'', s') = 1$ holds. There are possibly up to* **cvm** *parents of s; the index i refers to a specific parent according to some order.*

   We also assume that $\mathbf{cvr}(s, j) = \perp$ if no j-th parent of s exists. For convenience we will also assume if $\mathbf{cvr}(s, i), \mathbf{cvr}(s, j)$ are both defined it holds $|\mathbf{cvr}(s, i)| \geq |\mathbf{cvr}(s, j)|$ for $i \leq j$, i.e., the parents are ordered from larger to smaller.
6. **spt***: (Split Finding); given s it returns a pair (s', s'') such that the union of the subsets represented by s', s'' is equal to the subset represented by s and s', s'' are disjoint. If s is an atom it returns $\perp$.*

We say that a family Φ is efficient if $\mathsf{l}(n)$ is polylogarithmic in n and the above six algorithms are polynomial-time.

The split finding algorithm cited above is not needed for the purpose of broadcast encryption but it will become handy for traitor tracing something we will elaborate on in Chapter 4. In some set systems we note that splitting may return more than 2 subsets. We give the computational specification of various collections Φ for existing subset cover schemes in Section 2.5.

2.3.3 Compression of Key Material

In this section we will focus on the issue of key assignment and perform a closer examination of how **KeyGen** can be designed to improve its key storage characteristics. This is an important aspect that also has a bearing in the design of the underlying set systems. So far we have considered $\mathtt{BE}^{\Phi}_{\mathtt{basic}}$, where each subset in the set system has a unique key selected independently. This means that for each user u, for which we have $\mathsf{K}_u = \{k_\mathsf{j} \mid \mathsf{j} \in \mathcal{J}_u\}$ and $\mathcal{J}_u := \{\mathsf{j} \mid u \in \mathsf{S}_\mathsf{j}\}$, the user needs to store all the k_j keys in a resulting key vector of length $|\mathcal{J}_u|$. This approach provides the key-indistinguishability property perfectly as we have seen in Proposition 2.7.

Nevertheless, there are advantages in employing large set systems that exhibit tradeoffs betwen the communication overhead and the number of subsets a user belongs to. In such case, it can easily become very uneconomical to follow the independent key assignment approach above. For this reason we will see alternative strategies that enable the receiver to compress the key material. The compression mechanisms we are interested in should facilitate the selective decompression of the needed keys on the fly.

In this section, we will give a general key material compression technique that works for any set system. In a nutshell, the technique works as follows: some "select" subsets are assigned keys independently while the keys of the remaining subsets are derivable through computations employing the keys of the select subsets. The challenge that immediately springs up with this methodology is to ensure that key-indistinguishability is preserved.

Next we describe the compression technique in more detail. We will split the key poset into a *forest* $\mathcal{F}$ of "upward" looking trees so that the root of any tree is the smallest subset compared to the other subsets in its tree in the poset partial order. Hence, keys in each tree of $\mathcal{F}$ are supersets of the root and any path of the tree from root to leaf is a chain of the underlying poset Φ. In each tree the keys will be dependent and in particular any key that corresponds to a node in a tree would be derivable from the key of any ancestral node.

From the receiver perspective, a receiver u should be capable of computing the keys of the subsets in the filter $\mathsf{F}(u)$, i.e., all those subsets that contain u. It follows that u would need to store all the keys of the roots of the trees in the intersection forest $\mathcal{F} \cap \mathsf{F}(u)$ (by slightly abusing notation whenever $\mathcal{F}$ is a forest and A is a set of nodes, we denote $\mathcal{F} \cap A$ as the forest that is derived from $\mathcal{F}$ after removing any node that is not in A). It is clear that u will be able to derive any key in its filter from the keys given to it. But to preserve key-indistinguishability, we need to employ a suitable cryptographic primitive for deriving keys. We define this next.

Definition 2.19. *Let* K *be a key space and* $c \in \mathbb{N}$. *We say that the function* $\mathsf{f}_c : \mathsf{K} \mapsto \mathsf{K}^c$ *is a* c*-fold key-extender with* ε*-insecurity if any polynomial-time test exhibits statistical distance at most* ε *between the distribution* $\mathsf{f}_c(k)$ *and the distribution* $\langle k_1, \ldots, k_c \rangle$ *where* $k, k_1, \ldots, k_c$ *are random variables uniformly distributed over* K.

Note that K would be typically the set of all bitstrings of a particular length so a key-extender in such case would be a *pseudorandom bit sequence generator* that extends its seed by a multiplicative factor of c.

We next give the definition of a *key-forest* which is a partition of the set system Φ in a set of trees, i.e., a forest.

Definition 2.20. *Given a set system Φ, we say that $\mathcal{F} = \{\mathsf{F}_1, \ldots, \mathsf{F}_v\}$ is a key-forest of Φ if*

1. *F_i is a rooted directed tree for $i = 1, \ldots, k$ following the superset relation. Specifically if S_1 is a descendant of S_2 in F_i, it holds that $\mathsf{S}_1 \subset \mathsf{S}_2$; further, F_i is connected and there are no cycles in F_i, i.e., there are no nodes $\mathsf{S}_1, \mathsf{S}_2, \mathsf{S}_3, \mathsf{S}_4 \in \mathsf{F}_i$ with $\mathsf{S}_1 \subset \mathsf{S}_2$, $\mathsf{S}_1 \subset \mathsf{S}_3$ and $\mathsf{S}_2 \subset \mathsf{S}_4$, $\mathsf{S}_3 \subset \mathsf{S}_4$.*
2. *Any subset $\mathsf{S} \in \Phi$ belongs to exactly one tree F_i for some $i \in \{1, \ldots, k\}$.*

We say the key-forest $\mathcal{F}_\Phi$ is of degree c if it consists of trees whose nodes have degree strictly less than c.

We next give the definition for the broadcast encryption scheme $\mathtt{BE}^{\Phi}_{\mathcal{F},\mathsf{f}_c}$ that demonstrates the key compression strategy that employs a key-forest $\mathcal{F}$ of degree c and a key-extender $\mathsf{f}_c : \mathsf{K} \mapsto \mathsf{K}^c$. We use the notation $\mathsf{f}^i_c(k)$ to denote the i-th block of $\mathsf{f}_c(k)$ for $0 < i \leq c$.

Definition 2.21. *The scheme $\mathtt{BE}^{\Phi}_{\mathcal{F},\mathsf{f}_c}$ where $\mathcal{F}$ is a key-forest $\langle \mathsf{F}_1, \ldots, \mathsf{F}_v \rangle$ and f_c is a key-extender follows the template of Figure 2.2 with the following modifications:*

1. *Each subset $\mathsf{S}_\mathsf{j} \in \Phi$ is associated with a local $l_\mathsf{j} \in \mathsf{K}$ as well as a key k_j. It holds that $k_\mathsf{j} = \mathsf{f}^c_c(l_\mathsf{j})$.*
2. *If it happens that $\mathsf{S}_{\mathsf{j}_1}$ is the v-th child of $\mathsf{S}_{\mathsf{j}_0}$ in a tree F_i it holds that $l_{\mathsf{j}_1} = \mathsf{f}^v_c(l_{\mathsf{j}_0})$.*
3. *During the execution of* **KeyGen** *only the values l_j for which S_j is a tree-root in are selected independently at random from K. The remaining values determined based on f_c as defined in item 2.*
4. *The secret-key sk_u is defined as $(\mathcal{J}^{\mathcal{F}}_u, \mathsf{K}^{\mathcal{F}}_u)$ where $\mathcal{J}^{\mathcal{F}}_u$ contains those indices j that are tree roots in the intersection forest $\mathcal{F} \cap \mathsf{F}(u)$. $\mathsf{K}^{\mathcal{F}}_u = \{k_\mathsf{j} \mid \mathsf{j} \in \mathcal{J}^{\mathcal{F}}_u\}$.*
5. *During decryption a user will need to recover the key of some subset $\mathsf{S} \in \mathsf{F}(u)$. If it happens that S is a tree-root in $\mathcal{F} \cap \mathsf{F}(u)$ then u has the key. Otherwise it can derive it as follows. Denote the path to S from the root as follows $(\mathsf{S}_{\mathsf{j}_0}, \mathsf{S}_{\mathsf{j}_1}, \ldots, \mathsf{S}_{\mathsf{j}_m} = \mathsf{S})$ for some integer m. Assume that $\mathsf{S}_{\mathsf{j}_i}$ is the c_i-th child of the subset $\mathsf{S}_{\mathsf{j}_{i-1}}$ for $1 \leq i \leq m$. The local l_{j_m} assigned to the subset S can be derived from l_{j_0} by successive applications of $\mathsf{f}_c(\cdot)$. More specifically, $l_{\mathsf{j}_m} = \mathsf{f}^{c_m}_c(\mathsf{f}^{c_{m-1}}_c \ldots \mathsf{f}^{c_1}_c(l_{\mathsf{j}_0}))$. Finally the key k_{j_m} can be computed as $\mathsf{f}^c_c(l_{\mathsf{j}_m})$.*

Obviously the above methodology will impact the time required to derive any key that is assigned to a subset that a user belongs to. The compression technique defined above has a computational overhead that is bounded by the height of the underlying key-forest. Indeed, in the worst case, any user would have to compute as many applications of f_c as the height of the tallest tree to derive a key at the leaf-level. The maximum key-storage on the other hand is the number of trees in the forest $\max_{u\in[n]}(|\mathcal{F} \cap \mathsf{F}(u)|)$ and thus it depends on the structure of the key-poset and the choice of $\mathcal{F}$.

Note that, if the key-forest of a set system consists of single nodes, i.e. any node in the key-poset defines a tree of size 1, the key compression procedure defined above would yield an independent key for each subset, i.e. all subsets are assigned unique keys randomly and independently from each other (and in essence no compression would be achieved in this case).

Illustration.

Below we present an illustration of our key-compression strategy. Consider the set system Φ of figure 2.6(a). In this system that applies to 4 users there is a total of 11 encryption key (one for each subset) and it is possible to transmit to any subset of users using just two ciphertexts. Following an independent key assignment as the one suggested in the basic scheme $\mathtt{BE}^{\Phi}_{\mathtt{basic}}$ we will require from each user the storage of 5 keys. Consider now the forest $\mathcal{F}$ given in figure 2.6(b). This forest has 5 trees (two of height 2 and 3 of height 0); it is a degree 3 forest. Using the key assignment of $\mathtt{BE}^{\Phi}_{\mathcal{F},\mathsf{f}_c}$ we have that each user would need to store at most 3 keys.

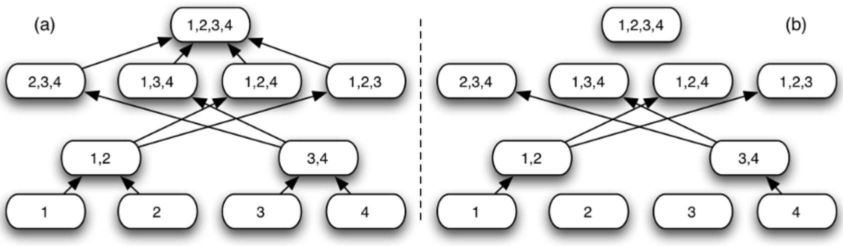

Fig. 2.6. An illustration of the key-compression strategy.

More specifically, observe that based on the key compression strategy on the given forest, we will choose at random the labels $l_{\{1\}}, l_{\{2\}}, l_{\{3\}}, l_{\{4\}}, l_{\{1,2,3,4\}}$. The remaining labels will be calculated based on a key-extender function f_3. User 2, for example, will receive the following data as her key material $l_{\{2\}}, l_{\{1,2\}} = f_3^1(l_{\{1\}}), l_{\{1,2,3,4\}}$. It is easy to see that user 2 based on her information can recover all necessary keys. For example the key of the subset $\{1, 2, 3\}$ is calculated as $k_{\{1,2,3\}} = f_3^3(f_3^2(l_{\{1,2\}}))$.

We will next prove the key-indistinguishability property (see Definition 2.6) of the key compression technique as given above in Definition 2.21.

Theorem 2.22. *The broadcast encryption* $\mathtt{BE}^{\Phi}_{\mathcal{F},\mathsf{f}_c}$ *satisfies the key indistinguishability property with distinguishing probability* $h_{\mathcal{F}} \cdot \varepsilon$ *where* $\mathcal{F}$ *is a key* *forest over Φ and f_c is a key extender with insecurity ε and $h_{\mathcal{F}}$ is the height of the forest $\mathcal{F}_\Phi$ plus one.*

Proof. We will first show the existence of a family of key generation procedures $\{\mathbf{KeyGen}^{\mathsf{j}}\}_{\mathsf{j}\in\mathcal{J}}$ with the property that for all j, $\mathbf{KeyGen}^{\mathsf{j}}$ selects the j-th key independently at random and it holds that for any probabilistic polynomial-time adversary $\mathcal{A}$, $\mathbf{Adv}_{\mathcal{A}}^{key-ind}(1^n) = |\mathbf{Prob}[\mathbf{Exp}_{\mathcal{A}}^{key-ind}(1^n) = 1] - \frac{1}{2}| \leq h_{\mathcal{F}} \cdot \varepsilon$, where the experiment is defined as in figure 2.4.

We consider a sequence of key generation $\mathbf{KeyGen}_i^{\mathsf{j}}$ procedures that are successive modifications of the way key generation works in the scheme $\mathtt{BE}_{\mathcal{F},\mathsf{f}_c}^{\Phi}$ for $i = 0, \ldots, l$. We first set $\mathbf{KeyGen}_0^{\mathsf{j}}$ to be exactly the key generation $\mathbf{KeyGen}$. The procedure $\mathbf{KeyGen}_1^{\mathsf{j}}$ modifies $\mathbf{KeyGen}_0^{\mathsf{j}}$ by substituting the labels of all level 1 children of the key tree F_j with random values from K while respecting the key derivation method for all remaining keys. Based on the property of the key extender f_c this incurs a distance of at most ε in distinguishing these two key generation algorithms. We continue in the same fashion for procedures $i = 2, \ldots, l, l+1$, following the path from the tree root of F_j to S_j. At each level we incur a distance of at most ε. At procedure $\mathbf{KeyGen}_l^{\mathsf{j}}$ the key k_{j} has been substituted by a random element of K. At last in the final modification, for $l+1$, we substitute the labels of all children of the subset S_j with random values to obtain $\mathbf{KeyGen}^{\mathsf{j}}$. Finally, we have the value k_{j} selected independently at random.

Based on this we derive that the distribution $\mathbf{KeyGen}$ of the scheme $\mathtt{BE}_{\mathcal{F},\mathsf{f}_c}^{\Phi}$ has a distance of at most $h_{\mathcal{F}} \cdot \varepsilon$, from the distribution $\mathbf{KeyGen}^{\mathsf{j}}$ for any j. This proves the fact that for any key indistinguishability adversary $\mathcal{A}$ against the broadcast encryption $\mathtt{BE}_{\mathcal{F},\mathsf{f}_c}^{\Phi}$ where $\mathcal{F}$ is a key forest over Φ and f_c is a key extender with insecurity ε, It holds that $\mathbf{Adv}_{\mathcal{A}}^{key-ind} \leq h_{\mathcal{F}} \cdot \varepsilon$ where $h_{\mathcal{F}}$ is the height of the forest $\mathcal{F}_\Phi$ plus one and $\mathbf{KeyGen}^{\mathsf{j}}$ is defined as above. ■

2.4 Revocation in the Key-Poset Framework

In order for a set system to be suitable for the design template of Figure 2.2 it should have an efficient covering algorithm that can be used for revocation. It follows that any proposal for a set system should be accompanied by a description of how the revocation algorithm works. Instead of treating the design of a fully-exclusive set systems with an efficient revocation algorithm on a case-by-case basis, in this chapter we will introduce an algebraic property that implies the existence of a revocation algorithm that optimally works for any set system that satisfies the property.

This section is structured as follows. In section 2.4.1 we will formalize the revocation problem in the key-poset framework and we will show its relation to a basic problem for a poset called `PatternCover`. We show that revocation is interreducible to `PatternCover` through an algebraic manipulation of the underlying key poset that we introduce called *chopping filters*. This observation is advantageous since in case we can solve `PatternCover` optimally we also have an optimal solution for the revocation problem.

In section 2.4.2, we introduce the property of *factorizability* for posets that characterizes a wide class of key posets for which we can optimally solve `PatternCover`. Specifically, we first observe that if a poset satisfies a property we call *separable* then `PatternCover` can be solved optimally. Building on this simple observation we then define factorizability for posets and we show that the operation of chopping filters when applied to a factorizable poset results to a separable poset. This provides the design of an optimal and efficient revocation algorithm.

2.4.1 Revocation in the key-poset framework: Definitions

Let Φ be a set system over the set $[n]$. Note that since Φ is a fully-exclusive set system, there exists a trivial revocation algorithm that searches for a broadcast pattern that covers $[n]\setminus \mathsf{R}$ in a brute-force manner. Moreover, if one does not wish to spend the time required to brute-force the optimal solution, there is a way to find a feasible cover that consists simply of all the individual elements in the set $[n]\setminus \mathsf{R}$; this is due to the fact that Φ is fully exclusive and thus all singletons should be present in Φ. Clearly these two approaches are undesirable as either inefficient or grossly wasteful in terms of communication overhead. In this section we will be interested in finding the shortest pattern in an efficient way. We will next define the revocation problem formally as the problem of finding a broadcast pattern for a set of enabled users that is minimal in size.

Definition 2.23. `Revocation` *Problem (Optimization Version).*
Input: A fully-exclusive set system Φ over set $[n]$ and a set of users R
Output: A minimal sized set $\mathcal{P} = \{\mathsf{S}_{\mathsf{j}_1}, \ldots, \mathsf{S}_{\mathsf{j}_m}\}$ of disjoint subsets in Φ, such that $[n]\setminus \mathsf{R} = \bigcup_{i=1}^{m} \mathsf{S}_{\mathsf{j}_i}$.

As we will show the revocation problem is a natural extension to the problem of `PatternCover` that we define next. `PatternCover` is the problem of finding an optimal cover for a family Φ.

Definition 2.24. `PatternCover` *Problem (Optimization Version).*
Input: A fully-exclusive set system Φ over set $[n]$
Output: A minimal sized set $\mathcal{P} = \{\mathsf{S}_{\mathsf{j}_1}, \ldots, \mathsf{S}_{\mathsf{j}_m}\}$ of disjoint subsets in Φ, such that $[n] = \bigcup_{i=1}^{m} \mathsf{S}_{\mathsf{j}_i}$.

Hardness of Pattern Cover for General Families.

We next discuss the hardness of `PatternCover` problem in the general case, something that shows that without imposing any special structure on the family there is no way to solve this problem efficiently (and the same would be true for the revocation problem). Specifically we show that `PatternCover` is NP-hard with a reduction from the the Set Cover problem. The difference between the two problems lies on the requirement that the instances of `PatternCover` are fully-exclusive (as opposed to arbitrary in the case of the Set Cover problem) and that they should admit the data structure specification of definition 2.18.

Theorem 2.25. `PatternCover` *is* NP*-hard.*

Proof. We recall that `SetCover` problem is NP-hard, because it is a special case of the `VertexCover` problem which itself is dual to `Clique`. It should be noted that `VertexCover` remains NP-complete even in graphs with node degree at most 3, see [48]. We next outline a reduction of `VertexCover` with node degree at most 3 to `PatternCover` thereby establishing its NP-hardness. Given a graph $G = (V, E)$ with node degree at most 3, Vertex Cover requires a minimal set of vertices $V' \subseteq V$ such that each edge in E is incident to at least one vertex in V'. The decisional version of Vertex Cover is a set of instances of the form $\langle G(V, E), b\rangle$ such that G is a graph with node degree at most 3 and there exists b vertices in V for which any edge in E is incident to one of these b vertices. From such a graph, we construct a fully-exclusive set system Φ over the set of edges E. A subset in Φ consists of edges for which there exists a vertex such that all edges are incident to it. Since nodes in G have degree at most 3, the maximum possible number of subsets in Φ would be $4|V| + |E|$ where each edge itself defines a subset and for any vertex we have at most 4 additional subsets, i.e. all three edges (if exist) incident to this vertex constitute a subset and all three combinations of two out of three edges define another three subsets. It is easy to see that Φ is a fully-exclusive set system. Further, we can describe an algorithmic specification according to Definition 2.18 by enumerating the vertices and defining a $\leq$ relation between the edges according to the vertex enumeration. Finally, we conclude that an optimal solution of the `VertexCover` problem would be interchangeable to an optimal cover in Φ over the set of users E. ■

Chopping Filters.

Despite the fact that revocation appears to be more general than `PatternCover`, we show that the two problems are interreducible. A reduction of `PatternCover` to `Revocation` is immediate (by simply considering $\mathsf{R} = \emptyset$). The other direction is the interesting one. Let us say a user u should be revoked; observe that any subset of Φ that contains u should not be in the cover of the set $\mathsf{N} \setminus \{u\}$. In other terms, finding a cover in Φ for $\mathsf{N} \setminus \{u\}$ is equivalent to finding a cover for $\mathsf{N}' = \mathsf{N} \setminus \{u\}$ in a new family Φ' where Φ' is constructed by deleting all supersets of $\{u\}$ in Φ which amounts to removing from Φ the filter $\mathsf{F}(u)$. To capture this we introduce formally the operation of chopping filters:

Definition 2.26. *Given the family $\Phi = \{\mathsf{S}_\mathsf{j}\}_{\mathsf{j}\in\mathcal{J}}$ and $\mathsf{j}_1, \ldots, \mathsf{j}_m \in \mathcal{J}$, we define the new family* **Chop**$(\Phi, \{\mathsf{j}_1, \ldots, \mathsf{j}_m\}) = \Phi \setminus \cup_{i=1}^{m} \mathsf{F}(\mathsf{S}_{\mathsf{j}_i})$*; the specification of its six algorithms is given in Figure 2.7.*

```
Chop(Subset Collection $\Phi$, list of encodings $\{\mathsf{j}_1, \ldots, \mathsf{j}_m\}$)
1.  set $\mathbf{cvm} = \Phi.\mathbf{cvm}$
2.  return descriptions of $(\mathbf{tst}, \mathbf{mbr}, \mathbf{slo}, \mathbf{sbs}, \mathbf{cvr}, \mathbf{spt})$ as follows:

Ctst($s$)
1.  if ($\Phi.\mathbf{tst}(s)$=1)
2.     if ($\Phi.\mathbf{sbs}(\mathsf{j}_1, s) = 1 \vee \ldots \vee \Phi.\mathbf{sbs}(\mathsf{j}_m, s) = 1$)
3.        return 0
4.     else return 1
5.  else 0

Cmbr($s, u$)
1.  return $\Phi.\mathbf{mbr}(s, u)$

Cslo($s_1, \ldots, s_r$)
1.  Let $U = \{u_1, \ldots, u_t\}$ such that $u \in U$ if and only if
      $u$ is an atom and there is some $1 \le i \le k$, such that $\mathsf{S}_{\mathsf{j}_i} = \{u\}$ .
2.  return $\Phi.\mathbf{slo}(s_1, \ldots, s_r, u_1, \ldots, u_t)$

Csbs($s, s'$)
1.  return $\Phi.\mathbf{sbs}(s, s')$

Ccvr($s, i$)
1.  $s' = \Phi.\mathbf{cvr}(s, i)$
2.  if ($\mathbf{Ctst}(s') = 1$)
3.    then output $s'$
4.  else return $\perp$

Cspt($s$)
1.  return $\Phi.\mathbf{spt}(s)$
```

Fig. 2.7. The computational description of a chopped family Φ.

It is easy to verify that the transformation pointed above from the family Φ to Φ' can be done in polynomial-time. Moreover if it happens that $\mathsf{S}_{\mathsf{j}_1}, \ldots, \mathsf{S}_{\mathsf{j}_m}$ are atoms, i.e., $\mathsf{S}_{\mathsf{j}_i} = \{u_i\}$ for some $u_i \in [n]$ for each $i = 1, \ldots, m$, it is easy to see that Φ' is fully exclusive over $[n] \setminus \mathsf{R}$ where R is equal to $\{u_1, \ldots, u_m\}$. Consider now an instance $\langle \Phi, [n], \mathsf{R} \rangle$ of the `Revocation` problem; note first that the optimal solution would be covering the set $[n] \setminus \mathsf{R}$. Note also that any subset in the optimal solution is also an element of the subset collection $\Phi' = \Phi \setminus \{\mathsf{S} \in \Phi \mid \mathsf{S} \cap \mathsf{R} \neq \emptyset\}$. Thus, the optimal solution is an optimal solution for the instance $\langle \Phi', [n] \setminus \mathsf{R} \rangle$ of `PatternCover` for the chopped collection Φ'.

Lemma 2.27. *The* `Revocation` *problem is polynomial-time interreducible to the* `PatternCover` *problem.*

Proof. Lemma 2.27 First consider an instance of the `PatternCover` problem, i.e., a fully exclusive set system Φ. We can form an instance of the `Revocation` problem by setting $\mathsf{R} = \emptyset$. It is clear that an optimal solution of this instance translates immediately to an optimal solution of the original `PatternCover` problem.

On the other hand, suppose that we have an instance of the `Revocation` problem, i.e., a fully exclusive set system Φ as well as a set of users R for revocation. We define the chopped family $\Phi' = \Phi \setminus \{\mathsf{S} \in \Phi \mid \mathsf{S} \cap \mathsf{R} \neq \emptyset\}$ as in figure 2.7. The set system Φ' is fully exclusive over $[n] \setminus \mathsf{R}$. An optimal solution of the `PatternCover` instance Φ' would immediately provide an optimal solution of the original `Revocation` problem.

In the next section, we will characterize sufficient algebraic properties of the underlying family Φ for solving the revocation problem optimally.

2.4.2 A sufficient condition for optimal revocation

In this subsection, we will characterize a simple sufficient algebraic property for the underlying set system Φ to have an efficient optimal revocation solution.

We will first focus on solving the `PatternCover` problem. It is easy to see an optimal solution of `PatternCover` problem defined for a set system Φ would consist of maximal subsets in the collection. The set of all maximal subsets might be overlapping, but if the maximal subsets of Φ are disjoint then an optimal solution could be easily found for such set systems. Since each subset has a corresponding principal ideal, we want to formalize the case when there are disjoint maximal subsets in Φ in terms of ideals. We introduce the following notion:

Definition 2.28. *We say a lower set* A *of a poset* Φ *is* separable *if in the lower-maximal partition* $\langle \mathsf{M}_1, \ldots \ \mathsf{M}_k \rangle$ *of* A *it holds that* M_i *is ideal for* $i = 1, \ldots, k$.

We show next that in a separable family Φ we can easily define an optimal solution for `PatternCover`. Indeed, since all ideals of Φ are principal, finding ideals is equivalent to finding the maximal subsets in Φ. In Figure 2.8 we present an optimal solution for a separable collection Φ; the intuition behind the algorithm is to build one chain for each ideal and use the basic algorithms of the collection to recover the ideals in the lower-maximal partition of Φ.

Theorem 2.29. *Assuming that* Φ *is separable, the algorithm in Figure 2.8 outputs an optimal solution to* `PatternCover` *problem which is of size* $\mathsf{ord}(\Phi)$.

```
PatternCover(Subset Collection $\Phi$)
1.  $\mathcal{P} = \emptyset$
2.  Repeat the following steps until break.
3.    $u = \Phi.\mathbf{slo}(\mathcal{P})$, let $s = \{u\}$
4.    if $u = \perp$ then output $\mathcal{P}$ and break.
5.    Repeat the following steps until $s$ does not change.
6.      Find $y = \Phi.\mathbf{cvr}(s, i)$ for minimal $i \leq \Phi.\mathbf{cvm}$ such that $y \neq \perp$
7.       if it exists.
8.      if $y \neq \perp$ then $s = y$
9.    $\mathcal{P} = \mathcal{P} \cup s$
```

Fig. 2.8. A `PatternCover` algorithm that works optimally for separable set systems.

Proof. Theorem 2.29 Note that the algorithm in Figure 2.8, starts from an atom in the poset of subset collection Φ and "grows" the current element until it hits the top of the ideal the selected atom was in. In the next iteration, another atom is selected outside the ideals so far included in the broadcast pattern. These ideals are actually the ones that make up the lower-maximal partition of the set system Φ. Overall, the algorithm will return the set of upper bounds from each ideal, i.e., their principal elements, which result in, of course, the optimal solution with size $\mathsf{ord}(\Phi)$. This is because, the optimal solution requires at least one element from each ideal necessarily. ■

The running time of the algorithm in Figure 2.8 is bounded by the sum of the chains followed by $\Phi.\mathbf{cvr}()$ in each of the ideals that comprise the lower maximal partition of Φ. Typical collections (see next section) have chains of polylogarithmic length in the number of users and thus the running time of the algorithm is quite efficient. The parent finding procedure $\mathbf{cvr}(\cdot,\cdot)$ of the set system plays an important role in this algorithm. Note that the `PatternCover` algorithm looks for a parent with a maximal size out of at most $\Phi.\mathbf{cvm}$ possible choices.

Factorizable Families.

We next introduce a basic property regarding the ideals of a family Φ that will play an important role in solving optimally the revocation problem. Intuitively the factorizable property mandates that the complements of the atomic filters in $(\Phi, \subseteq)$ share some similar properties to prime ideals over rings. We use the notation (S) to denote the principal ideal of $\mathsf{S} \in \Phi$.

Definition 2.30. *A fully-exclusive set system Φ is called* factorizable *if the poset, itself, is separable and for any subset $\mathsf{S} \subseteq \Phi$ and any atom u, it holds that $(\mathsf{S}) \cap \mathsf{P}_u$ is a separable set.*

In the following lemma we present some useful facts about the P_u sets and the factorizable property.

Lemma 2.31. *Let Φ be a factorizable fully exclusive set system defined over $[n]$.*

1. *For any $\mathsf{S} \in \Phi$, it holds that $(\mathsf{S}) = \cap_{u\in[n]\setminus\mathsf{S}}\mathsf{P}_u$.*
2. *For any $\mathsf{S} \in \Phi$, it holds that $\Phi \setminus \mathsf{F}(\mathsf{S}) = \cup_{u\in\mathsf{S}}\mathsf{P}_u$.*
3. *For any $\mathsf{S} \in \Phi$, (S) is a fully exclusive factorizable set system over S.*
4. *For any $u \in [n]$, and a separable set $\mathsf{A} \subseteq \Phi$, the set $\mathsf{P}_u \cap \mathsf{A}$ is separable.*

Proof. Lemma 2.31

1. If a subset $\mathsf{S}_1 \in (\mathsf{S})$, then it holds for any $u \in [n] \setminus \mathsf{S}$ that $\mathsf{S}_1 \in \mathsf{P}_u$; hence $\mathsf{S}_1 \in \cap_{u\in[n]\setminus\mathsf{S}}\mathsf{P}_u$. For the opposite direction, if $\mathsf{S}_2 \in \cap_{u\in[n]\setminus\mathsf{S}}\mathsf{P}_u$, then it holds that $\mathsf{S}_2 \cap ([n] \setminus \mathsf{S}) = \emptyset$; hence $\mathsf{S}_2 \subseteq \mathsf{S}$.
2. If $\mathsf{S}_1 \in \Phi \setminus \mathsf{F}(\mathsf{S})$, then it holds that there exists $u \in \mathsf{S}$ such that $u \notin \mathsf{S}_1$; hence $\mathsf{S}_1 \in \cup_{u\in\mathsf{S}}\mathsf{P}_u$. For the opposite direction, if $\mathsf{S}_2 \in \cup_{u\in\mathsf{S}}\mathsf{P}_u$, then it holds that there exists $u \in \mathsf{S}$ such that $u \notin \mathsf{S}_2$; hence $\mathsf{S}_2 \notin \mathsf{F}(S)$, i.e. $\mathsf{S}_2 \in \Phi \setminus \mathsf{F}(\mathsf{S})$.
3. It is easy to observe that (S) is fully-exclusive over the elements of S. Now consider a subset $\mathsf{S}_1 \in (\mathsf{S})$ and an atom $u \in \mathsf{S}$; we will check the set $(\mathsf{S}_1) \cap \mathsf{P}'_u$, where $\mathsf{P}'_u = (\mathsf{S}) \setminus \mathsf{F}(u)$. First if $u \notin \mathsf{S}_1$ then $(\mathsf{S}_1) \cap \mathsf{P}'_u = (\mathsf{S}_1)$ and we are done since (S_1) is ideal. Now suppose $u \in \mathsf{S}_1$. We will show that $(\mathsf{S}_1) \cap \mathsf{P}'_u = (\mathsf{S}_1) \cap \mathsf{P}_u$ where $\mathsf{P}_u = \Phi \setminus \mathsf{F}(u)$; this will establish that $(\mathsf{S}_1) \cap \mathsf{P}'_u$ is separable. If $\mathsf{S}_2 \in (\mathsf{S}_1) \cap \mathsf{P}'_u$, then it follows that $\mathsf{S}_2 \in (\mathsf{S}_1) \cap \mathsf{P}_u$ as $\mathsf{P}'_u \subseteq \mathsf{P}_u$. For the opposite direction, consider a subset $\mathsf{S}_3 \in (\mathsf{S}_1) \cap \mathsf{P}_u$, then it holds that $\mathsf{S}_3 \in (\mathsf{S}_1) \setminus \mathsf{F}(u) \subseteq \mathsf{P}'_u$; hence $\mathsf{S}_3 \in (\mathsf{S}_1) \cap \mathsf{P}'_u$.
4. Given that A is separable, by definition, the lower-maximal partition of A consists of ideals. We have $\mathsf{A} = \cup_{j=1}^{k}\mathsf{I}_j$. By Lemma 2.17 u belongs to only one of these ideals, say $\{u\} \in \mathsf{I}_s$. Thus, for any $j \neq s, 1 \leq j \leq k$, $\mathsf{I}_j \cap \mathsf{P}_u = \emptyset$. On the other hand, since Φ is factorizable, the lower maximal partition of $\mathsf{I}_s \cap \mathsf{P}_u$ consists of disjoint ideals and thus $\mathsf{P}_u \cap \mathsf{A}$ is a separable set.

■

Interestingly, the property of being factorizable has an alternative characterization that is of a more local nature:

Definition 2.32. *We say a fully-exclusive set system Φ has the diamond property if for any $\mathsf{S}_1, \mathsf{S}_2 \in \Phi$ such that $\mathsf{S}_1 \cap \mathsf{S}_2 \neq \emptyset$, then it holds that $\mathsf{S}_1 \cup \mathsf{S}_2 \in \Phi$.*

We prove the following:

Theorem 2.33. *A fully-exclusive set system Φ that is an ideal satisfies the following: Φ is factorizable if and only if Φ has the diamond property.*

Proof. Theorem 2.33 Suppose that Φ has the diamond property. We will prove that Φ is factorizable by showing that for any ideal I and any atom $u \in \mathsf{I}$, the lower set P_u inside I, i.e. $\mathsf{I} \cap \mathsf{P}_u$, is a separable set. Consider the lower-maximal partition $\langle \mathsf{M}_1, \ldots, \mathsf{M}_k \rangle$ of $\mathsf{I} \cap \mathsf{P}_u$ and suppose the converse is true, $\mathsf{I} \cap \mathsf{P}_u$

is not separable, i.e., there exists $l \in \{1,\ldots,k\}$ for which M_l is not ideal while it holds that $\mathsf{ord}(\mathsf{M}_l) = 1$. In such case M_l fails the directed property i.e., there are at least two elements a, b for which there is no upper bound, i.e., $\mathsf{maxnum}(\mathsf{M}_l) > 1$. Consider the set of maximal elements $\langle \mathtt{m}_1,\ldots,\mathtt{m}_s\rangle$, with $s = \mathsf{maxnum}(\mathsf{M}_l)$, that satisfies the properties (1-2) of Definition 2.14. Lemma 2.15 implies that there exist an index-pair i, j for which $\mathtt{m}_i$ and $\mathtt{m}_j$ share a common atom, i.e. the subsets corresponding to the maximal elements are intersecting. Denoting the subsets by S_i and S_j, we obtain $\mathsf{S} = \mathsf{S}_i \cup \mathsf{S}_j \in \Phi$ due to the diamond property. Observe now that, the node $\mathtt{m}$ corresponding to the subset S dominates exactly the atoms of $\mathtt{m}_i$ and $\mathtt{m}_j$ and hence we have $u \not\leq \mathtt{m}$; as a result $\mathtt{m} \in \mathsf{I} \cap \mathsf{P}_u$. Due to the disjointness of the lower sets in a lower maximal partition, as it is proven in Proposition 2.12, it holds that $\mathtt{m} \in \mathsf{M}_l$. However, this contradicts with the maximality of the elements $\mathtt{m}_i$ and $\mathtt{m}_j$ in M_l since it holds that $\mathtt{m}_i \leq \mathtt{m}$ and $\mathtt{m}_j \leq \mathtt{m}$. This concludes that M_l is an ideal for any $l \in \{1,\ldots,k\}$, hence $\mathsf{I} \cap \mathsf{P}_u$ turns out to be a separable set.

For the opposite direction, suppose that Φ is factorizable but does not satisfy the diamond property. There exists, then, two subsets $\mathsf{S}_1, \mathsf{S}_2$ that are intersecting and $\mathsf{S} = \mathsf{S}_1 \cup \mathsf{S}_2$ does not exist in Φ. Let us denote the nodes within the poset description corresponding to the subsets S_1 and S_2 by $\mathtt{m}_1$ and $\mathtt{m}_2$ respectively. Consider the minimal ideal I in size that contains the set of atoms dominated by $\mathtt{m}_1$ and $\mathtt{m}_2$, i.e., I is the smallest ideal (S') in size such that $(\mathtt{m}_1 \in (\mathsf{S}') \wedge \mathtt{m}_2 \in (\mathsf{S}'))$; denote the principal element of I by $\mathtt{m}$. Since $\mathsf{S} \notin \Phi$, there exists an atom $u \leq \mathtt{m}$ for which we have $u \not\leq \mathtt{m}_1$ and $u \not\leq \mathtt{m}_2$. On the other hand, due to factorizability, $\mathsf{I} \cap \mathsf{P}_u$ is separable, i.e. the lower maximal partition $\langle \mathsf{I}_1,\ldots,\mathsf{I}_s\rangle$, for some $s \in \mathbb{Z}$ of $\mathsf{I} \cap \mathsf{P}_u$ consists of ideals that are strict subsets of I. Recall that the sets in the above lower-maximal partition are disjoint and their union covers the set $\mathsf{I} \cap \mathsf{P}_u$, (see Proposition 2.12); hence given that $\mathsf{S}_1 \cap \mathsf{S}_2 \neq \emptyset$, the nodes $\mathtt{m}_1$ and $\mathtt{m}_2$ are contained in the same ideal I_j for some $j \in \{1,\ldots,s\}$, i.e. $\mathtt{m}_1 \in \mathsf{I}_j \wedge \mathtt{m}_2 \in \mathsf{I}_j$. This contradicts with the assumption on minimality of I. ■

Optimal Solution of Revocation for Factorizable Families.

We are now ready to describe an algorithmic solution for the revocation problem over any factorizable family Φ. Recall the reduction of the revocation problem to the `PatternCover` problem given in lemma 2.27 that was based on chopping a series of atomic filters from Φ. We next prove the following interesting result about factorizable families: the action of chopping atomic filters splits a factorizable family into ideals.

Theorem 2.34. *Given a factorizable set system Φ, let $\Phi' = \mathbf{Chop}(\Phi, \{\mathsf{j}_1, \mathsf{j}_2, \ldots, \mathsf{j}_r\})$ for a set of elements $\{\mathsf{j}_i\}_{i\in[r]}$ that all encode singleton subsets, i.e., j_i encodes the subset $\{\ell_i\}$ with $\ell_i \in [n]$ for $i = 1,\ldots,r$. Then, it holds that $\Phi' = \cap_{i=1}^r \mathsf{P}_{\ell_i}$. Futher, Φ' is separable.*

Proof. Theorem 2.34 We first note that, the set system $\Phi' = \mathbf{Chop}(\Phi, \{\mathsf{j}_1, \mathsf{j}_2, \ldots, \mathsf{j}_r\})$ corresponds to the partially ordered set $\cap_{i=1}^r P_{\ell_i}$ where P_{ℓ_i} is the complement of the atomic filter $\mathsf{F}(\ell_i)$ within the set system Φ. Indeed, if a subset S is an element of Φ', then $\mathsf{S} \cap \{\ell_1, \ldots, \ell_r\} = \emptyset$ which implies that $\mathsf{S} \in \mathsf{P}_{\ell_i}$ for any $1 \leq i \leq r$. Similarly, for any subset S of $\cap_{i=1}^{r} \mathsf{P}_{\ell_i}$ it holds that $\mathsf{S} \in \Phi'$.

We will now prove by induction on r that $\cap_{i=1}^{r} \mathsf{P}_{\ell_i}$ is a separable set. This will show that **Chop**(Φ, $\{\mathsf{j}_1, \mathsf{j}_2, \ldots, \mathsf{j}_r\}$) constitutes a separable set system. The $r = 1$ case can be proved as follows: given that Φ is factorizable, i.e. itself is separable, we obtain that the set $\mathsf{P}_{\ell_1} = \Phi \cap \mathsf{P}_{\ell_1}$ is separable as a consequence of Lemma 2.9 and 2.31(4). Consider now the induction hypothesis that for any $1 \leq s < r$ and $\ell_1, \ldots, \ell_s$, it holds that $\cap_{i=1}^{s} \mathsf{P}_{\ell_i}$ is a separable set. Now, consider r atoms over Φ.

Using the induction hypothesis, the intersection of the complement of atomic filters of the first $r-1$ atoms will be a separable set, i.e., $\cap_{i=1}^{r-1} \mathsf{P}_{\ell_i} = \mathsf{A}$ where A is a separable set. The Lemma 2.31(4) implies that the intersection of A with P_{ℓ_r} is separable which concludes the statement of the theorem. ■

In light of the Theorems 2.29 and 2.34, the algorithm in Figure 2.9 outputs an optimal solution for the revocation problem for a factorizable set system Φ using the chopping filters operation in conjunction to the **PatternCover** algorithm for separable families.

```
Revoke(Φ, R)
1.  let $\mathsf{R} = \{\ell_1, \ldots, \ell_r\}$ and $\mathsf{j}_i$ be the encoding for $\{\ell_i\}$
2.  define $\Phi' = $ Chop($\Phi, \{\mathsf{j}_1, \ldots, \mathsf{j}_r\}$)
3.  output PatternCover($\Phi'$)
```

Fig. 2.9. Optimal Solution for the revocation problem in a factorizable set system.

Theorem 2.35. *Given that the fully-exclusive set system Φ is factorizable, the algorithm in Figure 2.9 outputs an optimal solution for the revocation problem with respect to a set of revoked users $\mathsf{R} = \{\ell_1, \ell_2, \ldots, \ell_r\}$. The number of subsets in the solution will be $\mathsf{ord}(\cap_{i=1}^{r} \mathsf{P}_{\ell_i})$.*

Proof. Theorem 2.35 The proof follows from Theorem 2.34 and the correctness of the `PatternCover` algorithm given in Figure 2.8. As shown in Theorem 2.34, **Chop**(Φ, $\{\mathsf{j}_1, \ldots, \mathsf{j}_r\}$), where j_i is the encoding for the singleton $\{\ell_i\}$, will be set of disjoint principal ideals, thus the optimal solution will consist of those principal elements. Their number will be as many as the number of ideals in $\cap_{i=1}^{r} \mathsf{P}_{\ell_i}$ which is equal to $\mathsf{ord}(\cap_{i=1}^{r} \mathsf{P}_{\ell_i})$. ■

As shown above, for a given factorizable family Φ and a set of users, $\mathsf{R} = \{\ell_1, \ell_2, \ldots, \ell_r\}$, the intersection $\cap_{i=1}^{r} \mathsf{P}_{\ell_i}$ has a lower-maximal partition consisting of disjoint ideals. Their number would equal to the transmission overhead for the ciphertext to be transmitted. We can provide an explicit upper bound for the transmission overhead by providing an upper bound on the size of the lower-maximal partition of $\mathsf{I} \cap \mathsf{P}_u$ for any ideal I and atom u of Φ in terms of the size of ideal I.

Theorem 2.36. *Suppose that a fully-exclusive set system Φ over $[n]$ is factorizable. If for any subset $\mathsf{S} \in \Phi$ and an atom u, it holds that $\mathsf{ord}((\mathsf{S}) \cap \mathsf{P}_u) \leq f$ then for any $\ell_1, \ldots, \ell_r \in [n]$, $\mathsf{ord}(\cap_{i=1}^{r} \mathsf{P}_{\ell_i}) \leq r(f-1)+1$.*

Proof. Theorem 2.36 We consider the successive chopping of filters $\mathsf{F}_{\ell_1}, \ldots, \mathsf{F}_{\ell_r}$ from Φ. Given the factorizability of the family, each chopping operation applies to a specific ideal in the lower-maximal partition. Initially this ideal is Φ, in the second step it is one of the ideals of P_{ℓ_1} and so on. Given that $\mathsf{ord}((\mathsf{S}) \cap \mathsf{P}_u) \leq f$ for any $\mathsf{S} \in \Phi$, we have that at each successive chopping, the ideal that is selected will be removed from the list and at most f ideals would be added. Thus, the total number of ideals at the end will be bounded by $f + (r-1)(f-1) = r \cdot f - r + 1$. ■

2.5 Constructions

In this section we present a series of constructions of broadcast encryption schemes in the key poset framework. In each case we give first a combinatorial description and then we provide the key poset description.

2.5.1 Complete Subtree

The Complete Subtree (CS) is a method that defines a set system over a binary tree where the users are located on the leaves of the tree, and any intermediate node defines a subset in the set system which contains the users placed on the leaves rooted at this node. It is assumed that the number of users n is a power of 2. In this way a set system Φ^{CS} is defined over a user population $[n] = \{1, \ldots, n\}$ that consists of $2n-1$ subsets each corresponding to a node in the full binary tree with n leaves.

Set system description in the KP framework.

We next provide the description of the complete subtree method in the key poset framework. Let $\mathcal{J}^{CS}$ be the set of encodings of subsets in the set system Φ^{CS} defined over the set $\mathsf{N} = \{1, \ldots, n\}$. Recall that for simplicity, we assume that $\log n$ is an integer.

An encoding $\mathsf{j} \in \mathcal{J}^{CS}$ is a binary string of length at most $\log n$. Each such encoding corresponds to an index of a node in a full binary tree where the indices are constructed in a top-down manner: the root of the binary tree is encoded by ϵ (the empty string by ϵ), an index of a left child is constructed by appending '0' to its parent index, while an index of a right child is constructed by appending '1' to its parent index.

1. *Encoding Testing:* $\mathbf{tst}(\mathsf{j})$*; return* 1 *if* $\mathsf{j} = \{0,1\}^s$ *for some* $s \in \{0, 1 \ldots, \log n\}$.
2. *Membership Testing:* $\mathbf{mbr}(\mathsf{j}, u)$*; if* $\mathsf{j} = \epsilon$ *or* j *is a prefix of* u*, then return 1.*
3. *Outside Selection:* $\mathbf{slo}(\mathsf{j}_1, \mathsf{j}_2, \ldots, \mathsf{j}_r)$*; if* $\mathsf{j}_i = \epsilon$ *for an* $i \leq r$ *return* $\perp$*, otherwise compute the mimimal string* s*, that does not have any of* j_i*'s as a prefix. This can be done quite efficiently starting from the most significant bits of the input strings. Then pad the string* s *with* 0 *and* 1*'s arbitrarily to prepare a string* u *of length* $\log n$*; output* u.
4. *Inclusion Testing:* $\mathbf{sbs}(\mathsf{j}, \mathsf{j}')$*; if* j' *is a prefix of* j *then return 1.*
5. *Parent Finding:* $\mathbf{cvr}(\mathsf{j}, i)$*; if* $i \neq 1$ *then return* $\perp$*. Otherwise, given that* $\mathsf{j} = \mathsf{j}'b$ *for* $b \in \{0, 1\}$*, output* j'.
6. *Split Finding:* $\mathbf{spt}(\mathsf{j})$*; if* $|\mathsf{j}| < \log n$ *then return the subset pair* $(\mathsf{j}0, \mathsf{j}1)$ *else return* $\perp$.

It is straightforward to see that the complete subtree family as defined above is efficient.

The key-forest for compression.

The key forest $\mathcal{F}_{CS}$ used for compression is defined trivially by having trees of size one, i.e., each node constitutes a tree by itself and thus there is no compression. Hence, the key-handling procedure of Definition 2.21 would essentially yield a information theoretical key-assignment where each subset is assigned a unique key randomly and independently. It is not hard to observe that there is no possible way to derive a better strategy in this case : indeed any upward looking tree in the complete subtree poset needs to be a single node.

The factorizability of the complete subtree set system.

We see next that the complete subtree set system satisfies the factorizability property and thus it inherits the revocation algorithm of Figure 2.9.

Proposition 2.37. *The set system* Φ^{CS} *is a factorizable set system.*

Proof. Our proof will take advantage of the alternative characterization for factorizability, that is the "diamond property" given in definition 2.32 which is shown to be equivalent with factorizability (see Theorem 2.33).

Indeed, it is immediate to see that if any two subsets in Φ^{CS} are intersecting, then one of them is actually completely containing the other and hence their union is also in the set system. ■

Since the Complete Subtree method satisfies the factorizability property the revocation algorithm given in Figure 2.9 finds the optimal cover.

Transmission Overhead

As the theorem 2.36 illustrates the transmission overhead depends on the order of the lower maximal partition of an intersection $(S) \cap \mathsf{P}_u$. For the set system Φ^{CS} we have the following:

Proposition 2.38. *For any subset* $\mathsf{S} \in \Phi^{CS}$ *and an atom* u*, it holds that* $\mathsf{ord}((\mathsf{S}) \cap \mathsf{P}_u) \leq \log |\mathsf{S}|$

Proof. In the key-poset of the complete subtree set system, $(\mathsf{S}) \cap \mathsf{P}_u$ consists of the subtrees hanging from the path from u to the node corresponding to the subset S. Hence, there would be $\log |\mathsf{S}|$ ideals in the lower-maximal partition of the intersection $(\mathsf{S}) \cap \mathsf{P}_u$. ■

The above result combined with Theorem 2.36 gives an upper bound for the transmission overhead that equals $r(\log n - 1) + 1 \leq r \log n$. We will show next a slightly more refined analysis that brings this bound further down.

Theorem 2.39. *The transmission overhead of the optimal revocation algorithm given in Figure 2.9 for the factorizable set system* Φ^{CS} *is bounded by* $r(\log n/r - 1)$ *where* r *is the number of revoked users.*

Proof. Observe that the chopped set system $\mathbf{Chop}(\Phi, \{\mathsf{j}_1, \ldots, \mathsf{j}_r\})$ is constructed by chopping the filter of the subset $\{\ell_i\}$ encoded by j_i one by one starting from $i = 1$. For each user, the chopping takes an intersection of the filter of the user with an ideal containing it (recall Theorem 2.35 for more details on how the revocation algorithm operates). In the light of proposition 2.38, a subset contributes on the size of the broadcast pattern (which is transmission overhead) as many as elements as its height. It follows that in each revocation the subset S that contains the revoked user in question contributes $\log |\mathsf{S}| - 1$ subsets in the broadcast pattern.

It remains to find the maximum size that a subset in each revocation can have. Let $i = 1, \ldots, r$ denote the revocation index and m_i be the maximum size a subset can have when revoking the i-th user. We have that $m_1 = n$, $m_2 = n/2$, $m_3 = m_4 = n/4$ and in general $m_i = n/2^l$ where $2^{l-1} < i \leq 2^l$.

Using this series, the upper bound on the broadcast pattern would then be given by the summation

$$\sum_{i=1}^{r} \log m_i - (r - 1)$$

where we subtract $r - 1$ since at each revocation we will be picking up an existing subset and substituting with the ones that result from the intersection with the subsets in the atomic filter of the user.

Let k be such that $2^k \leq r < 2^{k+1}$. We analyze the above summation to obtain the following upper bound

$$\log n + \sum_{l=1}^{k} \sum_{i=2^{l-1}+1}^{2^l} (\log n - l) + (r - 2^k)(\log n - k - 1) - (r-1) =$$

$$= \log n + \sum_{l=1}^{k} 2^{l-1}(\log n - l) + (r - 2^k)(\log n - k - 1) - (r-1)$$

Now based on the identities $\sum_{l=1}^{k} 2^{l-1} = 2^k - 1$ and $\sum_{l=1}^{k} l2^{l-1} = (k+1)2^k - 2^{k+1} + 1$ we simplify the expression to

$$2^k \log n - (k-1)2^k - r + (r - 2^k)(\log n - k - 1)$$

After the calculations and using the fact $k > \log r - 1$ we get

$$r \log n - r(k+1) + 2^{k+1} - r \leq r \log n - r(k+2) + 2r \leq r(\log n - \log r + 1)$$

This completes the proof. ■

A direct revocation algorithm.

We will next discuss a direct revocation algorithm for the CS method that has the same performance as the generic algorithm that is inherited due to factorizability. Given a set of users R to be revoked, the broadcast pattern that covers $[n] \setminus \mathsf{R}$ can be determined by computing first the Steiner tree $\mathsf{Steiner}(\mathsf{R})$. Recall that the Steiner tree is the minimal subtree of the full binary tree that connects all the leaves in R. Consider now the nodes that are "hanging" from the tree $\mathsf{Steiner}(\mathsf{R})$. Assume there are m such nodes corresponding to the subsets $\mathsf{S}_1, \mathsf{S}_2, \ldots, \mathsf{S}_m$. We say a node is hanging from the Steiner tree if its sibling is in the Steiner tree while itself is not. See Figure 2.10 for the hanging nodes for a simple revocation instance in the case of 16 users; ; the nodes hanging from the Steiner Tree constitute the broadcast pattern. It holds that $[n] \setminus \mathsf{R} = \cup_{i=1}^{m} \mathsf{S}_i$; indeed for any $i \in \{1, \ldots, m\}$, S_i would not contain any revoked user. On the other hand, if $u \notin \mathsf{R}$ then, there exists some node S that is on the path from u to the root of the binary tree and hangs from the Steiner tree. It follows that the broadcast pattern that is revoking R would be $\{\mathsf{S}_1, \ldots, \mathsf{S}_m\}$. The transmission overhead of the broadcast encryption scheme that is using the CS as the underlying set system would be upper-bounded by the number of inner nodes in the Steiner tree $\mathsf{Steiner}(\mathsf{R})$. An analysis on the number of nodes in a Steiner tree with $|\mathsf{R}| = r$ leaves would yield a transmission overhead of size $r \log(n/r)$. We argue about this next.

Theorem 2.40. *The size of the broadcast pattern output resulting from the revocation algorithm outlined above is at most $r \log(n/r)$ where $0 < r < n$ is the size of the revoked set R.*

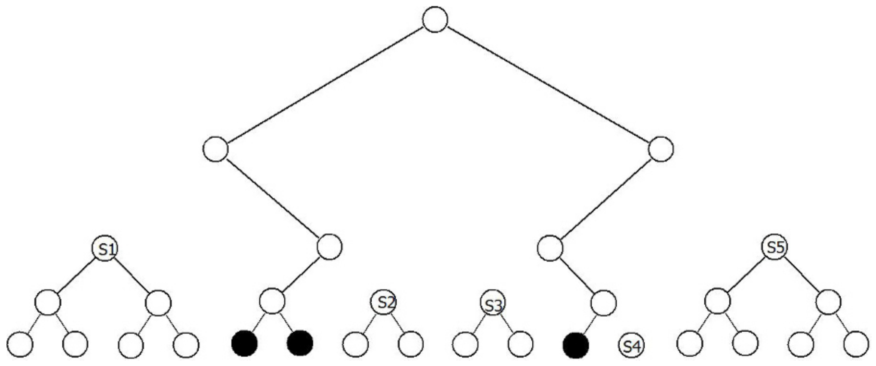

Fig. 2.10. Steiner tree that is connecting the revoked leaves.

Proof. Observe that the number of subsets hanging from the Steiner Tree $\mathsf{Steiner}(\mathsf{R})$ is exactly the number of nodes in $\mathsf{Steiner}(\mathsf{R})$ that have degree 1. We will now prove the above claim by induction on the height of the tree, i.e. induction on $\log n$.

Base Case $\log n = 1$: This case corresponds to a set system with two users only, and the claim can be easily seen to hold for any r, $r = 1, 2$.

Induction Assumption for $\log n = s$: Suppose that the number of subsets hanging from the Steiner Tree $\mathsf{Steiner}(\mathsf{R})$ is at most $r(s - \log r)$ for any subset $\mathsf{R} \subseteq \mathsf{N}$ with $|\mathsf{R}| = r$ and $0 < r < n$.

Induction Step for $\log n = s + 1$: We have two cases: either all the revoked users are located on the same subtree of one of the children of the root, or $r = r_1 + r_2$ so that r_1 users are located on the left child of the root, and r_2 users are located on the right child.

In the first case, due to induction assumption, $\mathsf{Steiner}(\mathsf{R})$ would yield at most $r(s - \log r) + 1$ nodes of degree 1, where the extra node is because of the root, and the remaining are within the child contains all revoked users. The induction step is proven in this case since $r(s - \log r) + 1 \leq r(s + 1 - \log r)$.

In the second case, due to the induction assumption $\mathsf{Steiner}(\mathsf{R})$ would yield at most $r_1(s - \log r_1) + r_2(s - \log r_2)$ nodes of degree 1.

$$\begin{aligned} r_1(s - \log r_1) + r_2(s - \log r_2) &= rs - (r_1 \log r_1 + r_2 \log r_2) \\ &\leq rs - (-r + r \log r) \\ &= r(s + 1) - r \log r \\ &= r(s + 1 - \log r) \end{aligned}$$

The above, indeed, holds, since it holds that $r(\log r - 1) \leq r_1 \log r_1 + r_2 \log r_2$ for any possible r, r_1, r_2 with $r = r_1 + r_2$. ■

2.5.2 Subset Difference

We will next discuss the subset difference (SD) set system that outperforms the complete subtree set system in terms of transmission overhead at the expense of increasing the size of the set system. In fact the increase is such that the number of keys required to be stored by each user become uneconomical as it is linear in the number of users. This downside will be mitigated by a non-trivial application of the key compression technique of Section 2.3.3.

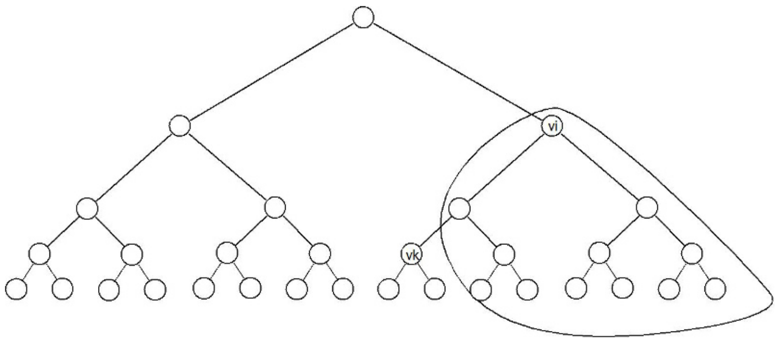

Fig. 2.11. The subset encoded by a pair of nodes (v_i, v_k) in the subset difference method.

We consider again a binary tree whose leaves correspond to the receivers in the user population $[n] = \{1, \ldots, n\}$ where n is a power of 2. A subset $\mathsf{S} \in \Phi^{SD} = \{\mathsf{S}_\mathsf{j}\}_{\mathsf{j} \in \mathcal{J}}$ can be denoted by a pair of nodes $\mathsf{j} = (v_i, v_k)$ in the binary tree where v_i is an ancestor of v_k. S_j is the set of all leaves in the subtree rooted at v_i excluding the leaves in the subtree rooted at v_k. See Figure 2.11 for an example description of a subset.

Set system description in the KP framework.

The key poset of the subset difference method is, as expected, more complex compared to the complete subtree set system. We refer reader to the Figure 2.12 for a depiction of the key poset of SD method for 8 users.

Let $\mathcal{J}^{SD}$ be the set of encodings of subsets in the set system Φ^{SD} defined over a set $[n]$. An element in $\mathcal{J}^{SD}$ is a pair of binary strings so that the sum of their length is less than or equal to $\log n$. Recall that for the sake of simplicity we assume that $\log n$ is an integer.

We define an index for each node in the binary tree in a top-down manner similar indexing as in the complete subtree set system : the root of the binary tree is encoded by ϵ, the index of a left child is constructed by appending '0' to its parent's index while index of a right child is constructed by appending '1' to its parent's index. An encoding $\mathsf{j} \in \mathcal{J}^{SD}$ now is a *pair* of strings (L, R)

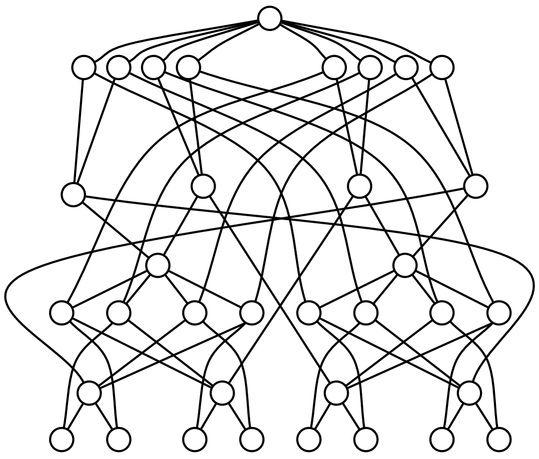

Fig. 2.12. A graphical depiction of the subset difference key poset for 8 users.

where the concatenated string LR has a length at most $\log n$. We refer reader to the Figure 2.13 for the computational specification of Subset Difference in the KP framework.

The key-forest for compression.

Consider the set of trees of the following form : let $L \in \{0,1\}^x$ with $0 \leq x \leq \log n - 1$ and $b \in \{0,1\}$. We have that F_{Lb} is a binary tree constructed as follows: it includes as root the node that corresponds to the subset j where $\mathsf{j} = (L, b)$. In addition, F_{Lb} includes all the nodes that can be added recursively following this rule : for any node j' in F_{Lb}, the first child of j' is $\mathbf{cvr}(\mathsf{j}', 2)$ and the second child of j' is $\mathbf{cvr}(\mathsf{j}', 3)$ (see Figure 2.13 for computational specification of the Subset Difference). The subset encoded by ϵ that contains the whole user population also forms a tree by itself as a single node that is denoted by F_ε. The family of trees is denoted by $\mathcal{F}_{SD}$.

Proposition 2.41. *The set $\mathcal{F}_{SD} = \{\mathsf{F}_s \mid s \in \{0,1\}^x, x = 0, \ldots, \log n\}$ is a key-forest of the set system Φ^{SD} with degree* 3.

Proof. Due to the definition of F_{Lb} for any $L \in \{0,1\}^x$ with $0 \leq x \leq \log n - 1$ and $b \in \{0,1\}$, it is easy to observe that the tree edges follow the superset relation, i.e. it holds that a parent of a node is a subset of that node. Now consider any four nodes in the tree F_{Lb} denoted by $(L_1, R_1), (L_2, R_2), (L_3, R_3), (L_4, R_4)$ for which it holds that R_1 is a prefix of R_2, R_3 and R_2, R_3 are both prefixes of R_4. While this condition is necessary for a cycle (albeit not sufficient) it also suggests that R_2, R_3 must satisfy that one is a prefix of the other and thus the four nodes cannot be in a cycle.

Computational Specification of Subset Difference in the KP Framework.

1. Encoding Testing: $\mathbf{tst}(\mathsf{j})$; if $\mathsf{j} = \epsilon$ return 1, otherwise parse j as $(\{0,1\}^x, \{0,1\}^y)$. If $x + y \leq \log n$ provided that $y \geq 1$, then return 1.
2. Membership Testing: $\mathbf{mbr}(\mathsf{j}, u)$; let $\mathsf{j} = (L, R)$ and $u \in \{0,1\}^{\log n}$. if L is a prefix of u whereas LR is not, then return 1.
3. Outside Selection: $\mathbf{slo}(\mathsf{j}_1, \mathsf{j}_2, \ldots, \mathsf{j}_r)$; if $\mathsf{j}_i = \epsilon$ for any $i \leq r$ then return $\perp$. Let $\mathsf{j}_i = (L_i, R_i)$ for $i = 1, \ldots, r$. We have two possible cases:
   (i) $\{s \in \{0,1\}^{\log n} \mid \exists i \in [r], R \text{ s.t. } s = L_i R\} \neq \{0,1\}^{\log n}$; this case implies that there exists a $u \in \{0,1\}^{\log n}$ such that none of L_i's are prefix of u; in this case return u. This string can be determined easily by computing the tree connecting the nodes corresponding to strings L_i.
   (ii) $\{s \in \{0,1\}^{\log n} \mid \exists i \in [r] \text{ s.t. } s = L_i R\} = \{0,1\}^{\log n}$; observe that for any u that is selected outside the subsets encoded by $\{\mathsf{j}_1, \ldots, \mathsf{j}_r\}$, it holds that if L_i is a prefix of u for any $i \leq r$ then $L_i R_i$ is also a prefix of u. We will compute such u, if it exists, in the following way: for all $i = 1, \ldots, r$, set $p = L_i R_i$ and test whether it holds that for all $j \neq i$, either $L_j R_j$ is a prefix of p or L_j is not a prefix of p. If such an i is found, $p = L_i R_i$ is padded arbitrarily to length $\log n$ to obtain a string u and return it. Otherwise return $\perp$.
4. Inclusion Testing: $\mathbf{sbs}(\mathsf{j}, \mathsf{j}')$; Let $\mathsf{j} = (L, R), \mathsf{j}' = (L', R')$, if L' is not prefix of L then return 0. We have two possible cases:
   (i) L is a prefix of $L'R'$: if LR is also prefix of $L'R'$ return 1, otherwise return 0.
   (ii) L is not prefix of $L'R'$: if $L'R'$ is prefix of L then return 0; else return 1.
5. Parent Finding: $\mathbf{cvr}(\mathsf{j}, i)$; We define for $i = 1, 2, 3, 4 = \mathbf{cvm}$.
   $i = 1$: if j is defined such that $\mathsf{j} = (Lb, R)$, where $b \in \{0,1\}$ and $|L| \geq 0$, then return (L, bR); otherwise return $\perp$.
   $i = 2$: if j is defined such that $\mathsf{j} = (L, R)$, where $|LR| < \log n$, then return $(L, R0)$; otherwise return $\perp$
   $i = 3$: if j is defined such that $\mathsf{j} = (L, R)$, where $|LR| < \log n$, then return $(L, R1)$; otherwise return $\perp$
   $i = 4$: if j is defined such that $\mathsf{j} = (Lb, R)$, where $b \in \{0,1\}$ and $|LbR| = \log n$, then return $(L, \overline{b})$; otherwise return $\perp$
6. Split Finding: $\mathbf{spt}(\mathsf{j})$; if j is defined such that $\mathsf{j} = (L, bR)$, where $b \in \{0,1\}$ and $|R| > 0$, then return $\{(L, b), (Lb, R)\}$; otherwise if $R = \epsilon$ then return $\{(L\overline{b}, 0), (L\overline{b}, 1)\}$ where $\overline{b} = 1 - b$.

Fig. 2.13. The computational specification of subset difference set system in the Key-Poset framework.

Regarding the coverage of the set family by the trees, consider a subset encoded as $\mathsf{j} = (L, R)$ where R is a string of length at least 1 starting with $b \in \{0, 1\}$. It is easy to see that this node is located in the binary tree F_{Lb} and not in any other binary tree.

Finally, the degree of the key-forest is 3 since the key forest consists of only binary trees. ■

Since the above key-forest has a degree of three, we can apply the key-assignment of Definition 2.21 by employing 3-fold key extender $\mathsf{f}_3 : \mathsf{K} \mapsto \mathsf{K}^3$.

It is easy to see that each of trees in the forest has height less than $\log n$ and hence the computation overhead for key decompression would be bounded by $\log n$. The key storage for a user u is the number of trees in the forest $|\mathcal{F}_{SD} \cap \mathsf{F}(u)|$ where $\mathsf{F}(u)$ refers to the filter of the user u in the key-poset (see Figure 2.14 for graphical illustration). Recall that user u is given a local value for the root of each tree in the intersection forest $|\mathcal{F}_{SD} \cap \mathsf{F}(u)|$. In order to find the number of trees in the intersection forest, we need to consider for each $L \in \{0, 1\}^x, b \in \{0, 1\}$ the number of trees that are produced when intersecting the tree F_{Lb} with the nodes of the filter $\mathsf{F}(u)$. We will denote this forest by $\mathsf{F}_{Lb} \cap \mathsf{F}(u)$. We distinguish the following three cases.

Case 1: if L is not a prefix of u; then $\mathsf{F}_{Lb} \cap \mathsf{F}(u)$ is empty.

Case 2: if L is a prefix of u, but Lb is not; then the intersection $\mathsf{F}_{Lb} \cap \mathsf{F}(u)$ is the tree F_{Lb} itself, meaningly all nodes in that tree is already included in the filter.

Case 3: If Lb is a prefix of u; then the intersection would have $\log n - |Lb|$ disjoint trees.

The second case happens for $\log n$ different choices of L as a prefix of u, (L can not be equal to u). The third case also happens for $\log n$ different choices of Lb as a prefix of u, with each case resulting $\log n - |Lb|$ disjoint trees. Also the user would need the one key for the unit-size tree F_ϵ. Based on the above, the key storage required by a user would be equal to

$$1 + \log n + \sum_{i=1}^{\log n} (\log n - i) = 1 + \frac{\log n \cdot (\log n + 1)}{2}$$

Factorizability of the subset difference set system.

We now prove that the subset difference set system satisfies the factorizability property and hence it inherits the optimal revocation algorithm of Figure 2.9.

Proposition 2.42. *The set system Φ^{SD} is a factorizable set system.*

Proof. Our proof will take advantage of the alternative characterization for factorizability, the diamond property given in definition 2.32 which is shown to be equivalent with factorizability (see Theorem 2.33).

Consider two subsets encoded by $\mathsf{j}_1 = (L_1, R_1)$ and $\mathsf{j}_2 = (L_2, R_2)$ that are intersecting. We will have to show that their union is in the set system. Since

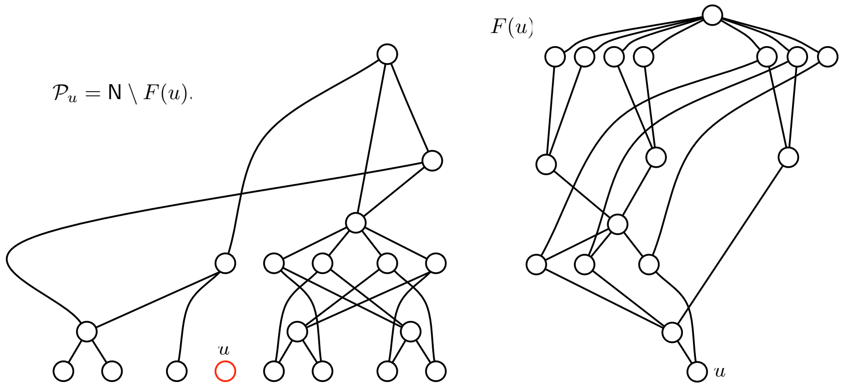

Fig. 2.14. The $\mathsf{P}(u)$ and $\mathsf{F}(u)$ sets for a user u in the subset difference key poset for 8 receivers.

they are intersecting there is a common element u for which it holds that L_1 and L_2 are prefixes of u. If $L_1 = L_2$, then it is easy to see that either of the following cases holds : (i) one is covering the other entirely, (ii) their union is equal to the subset $(M, \bar{b})$ where $L_1 = L_2 = Mb$ and $|b| = 1$, (iii) their union is equal to the subset of all users. In all three cases the union belongs to the set systems.

In the other case we assume without loss of generality that $|L_1| < |L_2|$. Then it holds that L_1 is a prefix of L_2. Moreover, $\mathbf{mbr}(\mathsf{j}_1, u) = 1$ implies that L_1R_1 is not a prefix of u while L_2 is a prefix of u, hence L_1R_1 can not be a prefix of L_2. Then we have two cases: (i) either the common prefix of L_1R_1 and L_2 is shorter than L_2, or (ii) L_2 is prefix of L_1R_1.

For case (i), since L_1R_1 is not a prefix of L_2, this case implies that the common prefix is also shorter than L_1R_1. Observe now that for any user u that satisfies $\mathbf{mbr}(\mathsf{j}_2, u) = 1$, it also holds that $\mathbf{mbr}(\mathsf{j}_1, u) = 1$. Hence one is covering the other, and their union, the superset among the two, is included in the set system.

For case (ii) observe that for any user u with a prefix L_1 either satisfies $\mathbf{mbr}(\mathsf{j}_1, u) = 1$ or satisfies $\mathbf{mbr}(\mathsf{j}_2, u) = 1$. Hence, the union of these two subsets would either correspond to the subset $(M, \bar{b})$ where $L_1 = Mb$ with $|b| = 1$ or the set of all users if $L_1 = \epsilon$. ■

Given that the subset difference set system satisfies the factorizability property, the revocation algorithm given in Figure 2.9 provides the optimal solution.

Transmission overhead.

As the theorem 2.36 illustrates the transmission overhead depends on the order of the lower maximal partition of an intersection $(S) \cap P_u$. The superiority of the set system Φ^{SD} compared to the complete subtree is mainly in that it satisfies the following:

Proposition 2.43. *For any subset* $\mathsf{S} \in \Phi^{SD}$ *and an atom* u*, it holds that* $\mathsf{ord}((\mathsf{S}) \cap P_u) \leq 3$.

Proof. Suppose S has the encoding $\mathsf{j} = (L, R)$. If $\mathtt{mbr}(\mathsf{j}, u) = 0$, then the intersection itself equals to the ideal (S), hence the proposition holds. If not, then L would be a prefix of u while LR is not a prefix of u. Then we can write $u = LL'bU$ where LL' is the longest common prefix of LR and u so that $LR = LL'\bar{b}R'$ and $|LL'bU| = \log N$ for some U, R'. Observe that $b \in \{0, 1\}$ necessarily exists, but U and R' may possibly be empty strings.

Consider now the subsets, $\mathsf{j}_1 = (L, L'), \mathsf{j}_2 = (LL'b, U), \mathsf{j}_3 = (LL'\bar{b}, R')$ where $\mathsf{j}_2, \mathsf{j}_3$ are well defined in case U, R' exist respectively. It is easy to observe that none of these subsets are intersecting. Indeed if $\mathtt{mbr}(\mathsf{j}_1, v) = 1$ for some v, then it holds that LL' is not a prefix of v, so $\mathtt{mbr}(\mathsf{j}_2, v) = \mathtt{mbr}(\mathsf{j}_3, v) = 0$. Similarly, if $\mathtt{mbr}(\mathsf{j}_2, v) = 1$, then it holds that $LL'b$ is a prefix of v so $LL'\bar{b}$ not while v is explicitly excluded from j_1; the same reasoning applies to j_3.

On the other hand, it is easy to verify that any $v \in \mathsf{S} \setminus \{u\}$ would be included in one of the above subsets. Indeed, $v \in \mathsf{S}$ implies that L is a prefix of v while LR is not. There are two mutually exclusive cases : (i) either LL' is not a prefix of v or (ii) $LL'c$ is a prefix of v for some $c \in \{0, 1\}$:

Case (i) implies that $\mathtt{mbr}(\mathsf{j}_1, v) = 1$. On the other hand, for case (ii) if $c = b$, then it holds that $\mathtt{mbr}(\mathsf{j}_2, v) = 1$; otherwise it holds that $\mathtt{mbr}(\mathsf{j}_3, v) = 1$.

Next we observe that for $\mathsf{j}' \in \{\mathsf{j}_1, \mathsf{j}_2, \mathsf{j}_3\}$ and $i = 1, 2, 3, 4$, it holds that $\mathtt{cvr}(\mathsf{j}', i)$ would either contain u or are outside (S). This establishes the fact that the above subsets correspond to the principal ideals of the lower-maximal partition of $(\mathsf{S}) \cap P_u$. ■

Applying Theorem 2.36 provides an upper bound on the transmission overhead for the subset difference set system, resulting in an overhead of $2r + 1$. A simple observation would refine this bound: in particular if S is the subset for whole receiver population, i.e. the encoding of S is ϵ, then j_2 and j_3 would not exist in the above formulation, in such case the first chopping for Theorem 2.36 will result a lower-maximal partition of order 1. Hence, the overall transmission overhead will amount to $2r - 1$.

A direct revocation algorithm.

We also describe a direct revocation algorithm for the subset difference set system that has the same performance as the generic algorithm described above. This is useful for historical reasons and in order to provide further intuition for the way the set system works. The algorithm utilizes the Steiner Tree induced by the set of revoked users R and the root as well. In this case, $\mathsf{Steiner}(\mathsf{R})$ is the minimal subtree that connects all the leaves corresponding the user set R. We compute the "broadcast pattern" that is covering the users in $[n]\setminus \mathsf{R}$ iteratively by modifying the Steiner Tree at each step. We set initially $\mathsf{T} = \mathsf{Steiner}(\mathsf{R})$ and repeat the following until the tree is empty.

1. Find a node v in the tree T that has two children v_L and v_R with each one being an ancestor of a single leaf. Denote the leaf that is a descendent of v_L by v_i and the leaf that is a descendent of v_R by v_j. If no such node exists, i.e., there is only one leaf left in the tree, then set the nodes $v_i = v_j$ to the leaf, set v to be the root and $v_L = v_R = v$.
2. If $v_L \neq v_i$, then add the subset $\mathsf{S}_{\mathsf{j}_1}$ with encoding $\mathsf{j}_1 = (v_L, v_i)$ to the broadcast pattern. Likewise, if $v_R \neq v_j$, then add the subset $\mathsf{S}_{\mathsf{j}_2}$ with encoding $\mathsf{j}_2 = (v_R, v_j)$ to the broadcast pattern.
3. Remove from T all the descendents of v and make v a leaf.

We next show that the the size of the broadcast pattern is at most $2r - 1$ in the worst case scenario and $1.38r$ in the average case.

Theorem 2.44. *The size of the broadcast pattern output by the above revocation algorithm is at most* $2r - 1$ *and* $1.38r$ *on average where* r *is the size of the revoked set* R.

Proof. In each step of the above algorithm, the number of leaves is decreasing by 1. Indeed, v_i and v_j are replaced by a single node v, and at most two new subsets are included in the broadcast pattern. This continues until the last leaf which yields a single subset. Hence, it is quite easy to observe that the transmission overhead is bounded by $2(r - 1) + 1 = 2r - 1$.

Note that it is possible that v_i is left child of v, or v_j is right child of v. These cases do not generate a subset to be included in the broadcast pattern. As a result the transmission overhead can be much smaller than $2r - 1$.

We will now discuss the average-case for randomly chosen r users to be revoked, i.e. the expected number of subsets generated by the above revocation algorithm. First, we mark the nodes that are set as leaves during the revocation algorithm. As mentioned before, in each step two leaves are revoked and an ancestral node is set as a new leaf. Hence, there are $r - 1$ of those nodes that are marked and both left and right children of these nodes are included in the $\mathsf{Steiner}(\mathsf{R})$. Denoting the children of the marked leaves as $v_1, \ldots, v_{2r-2}$, a possible subset is generated for each of them depending on how the revoked users are located beneath each node v_i, for $i = 1, \ldots, 2r - 2$.

If v_i has an outdegree 2 in the Steiner Tree, then no subset will be generated, otherwise, if the outdegree is 1, a single subset will be generated. Assuming that there are k_i revoked users rooted at this node the probability that a subset is generated is $1/2^{k_i - 1}$. This comes from the event that the users are either placed all on the left subtree of the node or on the right subtree of the node. The expected number of subsets over the values of k_i is thus given by the summation:

$$\sum_{i=1}^{2r-2} \frac{1}{2^{k_i - 1}}$$

We next observe that $|\{i : k_i = x\}| \leq \frac{r}{x}$, for $x \in \{1, \ldots, r\}$. This observation enables the following upper bound:

$$\sum_{x=1}^{r} \frac{r}{x} \cdot \frac{1}{2^{x-1}} \leq 2r \sum_{x=1}^{\infty} \frac{1}{k} \cdot \frac{1}{2^k} \leq 2r \ln 2 \approx 1.38 \cdot r$$

This completes the proof. ■

2.5.3 Key Chain Tree

In this section we will describe the key-chain tree set system Φ^{KCT}. Consider again a binary tree whose leaves correspond to the receivers in the user population $[n] = \{1, \ldots, n\}$ where n is considered a power of 2. For each internal node v_i of the binary tree, consider the sequence of consecutive leaves of the tree rooted at v_i. Any consecutive sequence starting from the leftmost or rightmost leaf of the tree rooted at v_i amounts to a subset in the set system. See the figure 2.15 for two examples of subsets in the set system. With this description, it is not hard to observe that there are lists of leaves in the binary tree starting from the leftmost (resp. rightmost) leaf of an intermediate node v_i that will also yield a subset due to the node v_j if v_j is a left (resp. right) child of v_i. Hence, each subset is being considered more than once depending on the geometry of its leaves, which results an unnecessary increase in key-storage unless some special care is taken.

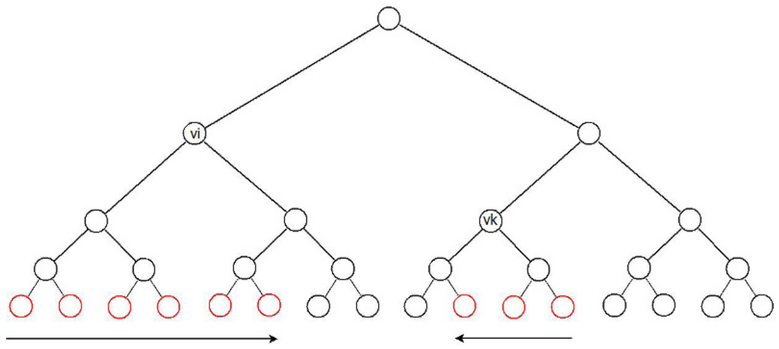

Fig. 2.15. An example of a subset in the Key Chain Tree method.

For the sake of avoiding the extra counting we define the following notion: we say a a pair of nodes (u_1, u_2) is *related* to an intermediate node v if the following hold: (i) v is the least-common ancestor of u_1 and u_2 and (ii) it holds that either u_1 is the leftmost leaf or u_2 is the rightmost leaf of the tree rooted at node v. We consider the pairs that can be related to an intermediate node as the subsets of the Key Chain Tree. This will be suffice to count each subset once.

Set System Description in the KP Framework

Let $\mathcal{J}^{KCT}$ be the set of encodings of subsets in the set system Φ^{KCT} defined over a set $[n] = \{1, \ldots, n\}$. An element in $\mathcal{J}^{KCT}$ is a pair of binary strings so that the sum of their length is equal to $\log n$. The algorithms, mostly, will be based on the integer representation of the encoding, hence we denote the procedure to convert a string into an integer by $\mathsf{str2int}$; specifically $\mathsf{str2int}(b_k \ldots b_0) = 1 + \sum_{i=0}^{k} 2^i \cdot b_i$. We also use the inverse function $\mathsf{int2str}$. As mentioned already, for the sake of simplicity, we assume that $\log n$ is an integer.

Regardless of the description of the set system, we can picture the key-poset of the basic set system in Figure 2.16. The figure reflects the the consecutive lists of users present in the set system as the long chains of nodes; especially observe the left and right sides of the triangle-looking key poset.

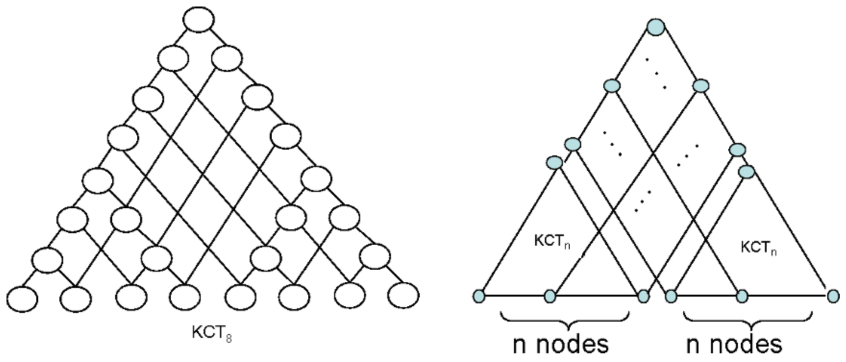

Fig. 2.16. (left) the key-poset of the key-chain tree method for 8 users. (right) the recursive definition of the key-poset for the key-chain tree for $2n$ users.

We define an index for each node in the binary tree in a top-down manner similar to the indexing of the complete subtree method: the root of the binary tree is encoded by ϵ (the empty string) and subsequently the index of a left child is constructed by appending '0' to its parent's index while the index of a right child is constructed by appending '1' to its parent's index. We denote the node corresponding to the string z by v_z for any $z \in \{0,1\}^i$ such that $0 \leq i \leq \log n$. An encoding $\mathsf{j} \in \mathcal{J}^{KCT}$ is a pair of strings $\mathsf{j} = (L, R)$ with $|L| + |R| = \log n$. The value L is the encoding of the root of the subtree that $\mathsf{S_j}$ is contained, i.e. the least common ancestor of all leaves in $\mathsf{S_j}$ will be the node v_L. In case $L = \epsilon$ then the least common ancestor of the leaves of the subset is the root of the binary tree; in case $|L| = \log n$ then the subset contains a single leaf corresponding to the encoding L. In case $0 < |L| < \log n$, observe that the value R is of the suitable length to identify a leaf in the subtree rooted by v_x. We determine the sequence of leaves that are in $\mathsf{S_j}$ as follows: (i) if R starts with 1 then S_j contains the sequence of leaves of the subtree rooted at v_x that start with the leftmost leaf and continue up to and including the leaf encoded by LR. (ii) if R starts with 0 then S_j contains the sequence of leaves of the subtree rooted at v_x that start with the rightmost leaf and continue towards the left up to and including the leaf encoded by LR.

The encoding that we put forth above is convenient as can be seen from the following facts. Given $\mathsf{j} = (L, 1R)$ we can represent the set $\mathsf{S_j}$ in the following way. Suppose that $a = \mathsf{str2int}(L0^{|R|})$ and $b = \mathsf{str2int}(L1R)$ then it holds that $\mathsf{S_j} = \{a, a+1, \dots, b\}$. On the other hand, if $\mathsf{j} = (L, 0R)$, suppose that $c = \mathsf{str2int}(L0R)$ and $d = \mathsf{str2int}(L1^{|R|})$ then it holds that $\mathsf{S_j} = \{c, c+1, \dots, d\}$.

Observe that the encoding scheme would cover all subsets of Φ^{KCT} exactly once with the only exception being the subsets that consist of the complete sets of leaves that are rooted at a node in the binary tree. These sets of leaves will be counted twice. For example, $(\epsilon, 0^{\log n})$ and $(\epsilon, 1^{\log n})$ are both referring to the set $\{1, \dots, n\}$. For this reason, for any L with $|L| < \log n$, we will consider invalid as encoding the pair $\mathsf{j} = (L, R)$ with $R = b^{\log n - |L|}$ where b is the last bit of L (and say $b = 0$ if $L = \epsilon$). In this way any $\mathsf{S} \in \Phi^{KCT}$ uniquely corresponds to an encoding $\mathsf{j} \in \mathcal{J}^{KCT}$.

To summarize the above for any L, R with $|L| + |R| + 1 = \log n$ we have

$$\mathsf{S}_{(L,0R)} = \{\mathsf{str2int}(L0R), \dots, \mathsf{str2int}(L1^{|R|+1})\}$$

and

$$\mathsf{S}_{(L,1R)} = \{\mathsf{str2int}(L0^{|R|+1}), \dots, \mathsf{str2int}(L1R)\}$$

while note that $\mathsf{S}_{(Lb, b^{\log n - |L| - 1})}$ and $\mathsf{S}_{(\epsilon, 0^{\log n})}$ are undefined.

The key forest for compression.

Let $L \in \{0,1\}^x$ with $0 \leq x \leq \log n - 2$. We define F_{L0} (resp. F_{L1}) as a chain of length $2^{\log n - x - 1} - 1$ which consists of subsets that are of the following form: a subset is included in chain F_{L0} (resp. F_{L1}) if it is a consecutive set of leaves rooted at node v_{L0} (resp. v_{L1}) including the rightmost (resp. leftmost) leaf rooted at node v_{L0} (resp. v_{L1}).

Observe that the leaf encoded by $(Lb\bar{b}^{\log n - x - 1}, \epsilon)$ with $\bar{b} =_{df} 1 - b$ for $b \in \{0,1\}$ either amounts to the rightmost leaf of the subtree rooted at v_{Lb} in case $b = 0$ or the leftmost leaf in case $b = 1$. The chain F_{Lb} can be formally described as follows: it contains the leaf that corresponds to the encoding $\mathsf{j} = (Lb\bar{b}^{\log n - x - 1}, \epsilon)$, and includes all the nodes in a chain recursively following this rule : for any node j' in F_{Lb}, if it holds that $\mathbf{cvr}(\mathsf{j}', 2) \neq (Lb, \bar{b}^{\log n - x - 1})$ (refer to the Figure 2.17 for the computational key assignment of Key Chain Tree) the next node (as a child of j') is assigned $\mathbf{cvr}(\mathsf{j}', 2)$, otherwise the chain ends as it reaches its maximum of length $2^{\log n - x - 1} - 1$.

In addition, we define F_{fl} to be the chain starting from $\mathsf{j} = (0^{\log n}, \epsilon)$ and F_{fr} be the chain starting from $\mathsf{j} = (1^{\log n}, \epsilon)$. In such case, the chain covers the consecutive set of leaves defined for the root of the Figure 2.15.

Computational Specification of Key Chain Tree in the KP Framework.

1. Encoding Testing: **tst**(j); Let the encoding j be (L,R) such that $|LR| = \log n$. Denote the last bit of L by $b \in \{0,1\}$, define $b = 0$ if $L = \epsilon$. If $R = b^{\log n - |L|}$ and $|x| < \log n$ then return 0 otherwise return 1.
2. Membership Testing: **mbr**(j, u); Let j be a valid encoding (L, bR) such that $|LbR| = \log n$, $b \in \{0,1\}$ and $u \in \{0,1\}^{\log n}$. Return 1 if and only if $\mathsf{str2int}(u) \in \mathsf{S}_\mathsf{j}$.
3. Outside Selection: **slo**($\mathsf{j}_1, \mathsf{j}_2, \ldots, \mathsf{j}_r$). Choose the smallest integer that lies outside the set $\mathsf{S}_{\mathsf{j}_1} \cup \ldots \cup \mathsf{S}_{\mathsf{j}_r}$ and if no such integer exists return $\perp$.
4. Inclusion Testing: **sbs**(j, j′); return 1 if and only if $\mathsf{S}_\mathsf{j} \subseteq \mathsf{S}_{j'}$.
5. Parent Finding: **cvr**(j, i), where $1 \le i \le$ **cvm** $= 2$.
   - $i = 1$: Let j be encoded by (L, bR) such that $|LbR| = \log n$ and $b \in \{0,1\}$. If $L = L'b\overline{b}^x$ for some L', x, then return $(L', b\overline{b}^x bR)$. Recall $\overline{b} =_{df} 1 - b$. Return $\perp$ if j is not of the form required.
   - $i = 2$: If $\mathsf{j} = (Lbb', \epsilon)$, then return $(Lb, \overline{b})$.
     Otherwise, let j be encoded by (L, bR) such that $|LbR| = \log n$ and $b \in \{0,1\}$:
     (i) $b = 0$; if $\mathsf{str2int}(bR) > 2$ then return (L, bR') such that $\mathsf{str2int}(bR') = \mathsf{str2int}(bR) - 1$, otherwise if $\mathsf{str2int}(bR) = 2$, return $(L'a, \overline{a}^{\log n - |L|})$ where $L = L'a$ for some $a \in \{0,1\}$ (return $\perp$ in case $L = \epsilon$), otherwise if $\mathsf{str2int}(bR) = 1$, we have that $bR = 0^{\log n - |L|}$ (and hence by the validity of encoding it holds that $L = L'1$) return $(L', 01^{\log n - |L|})$.
     (ii) $b = 1$; if $\mathsf{str2int}(bR) < 2^{|bR|} - 1$ then return (L, bR') such that $\mathsf{str2int}(bR') = \mathsf{str2int}(bR) + 1$, otherwise if $\mathsf{str2int}(bR) = 2^{|bR|} - 1$, return $(L'a, \overline{a}^{\log n - |L|})$ where $L = L'a$ for some $a \in \{0,1\}$ (return $(\epsilon, 1^{\log n})$ in case $L = \epsilon$), otherwise if $\mathsf{str2int}(bR) = 2^{|bR|}$, we have $bR = 1^{\log n - |L|}$ (and hence by the validity of encoding it holds that either $L = L'0$ or $L = \epsilon$) return $(L', 10^{\log n - |L|})$ if $|L| > 0$ and $\perp$ otherwise.
6. Split Finding: **spt**(j); There are two cases:
   (i) Let j be encoded by $(L, 0R)$ such that $|L0R| = \log n$. Suppose $R = 1^x 0R'$, then output $\{(L01^x, 0R'), (L1, 0^{|R|})\}$.
   (ii) Let j be encoded by $(L, 1R)$ such that $|L1R| = \log n$. Suppose $R = 0^x 1R'$, then output $\{(L10^x, 1R'), (L0, 1^{|R|})\}$.

Fig. 2.17. The computational specification of the key chain tree set system in the Key-Poset framework.

As an illustration, let us write the chains for the key forest of a key chain tree with 8 users given in Figure 2.16, i.e. the case $\log n = 3$.

$\mathsf{F}_{00} = (001, \epsilon)$	$: \{2\}$
$\mathsf{F}_{01} = (010, \epsilon)$	$: \{3\}$
$\mathsf{F}_{10} = (101, \epsilon)$	$: \{6\}$
$\mathsf{F}_{11} = (110, \epsilon)$	$: \{7\}$
$\mathsf{F}_{0} = (011, \epsilon) \to (01, 0) \to (0, 01)$	$: \{4\} \to \{4,3\} \to \{4,3,2\}$
$\mathsf{F}_{1} = (100, \epsilon) \to (10, 1) \to (1, 10)$	$: \{5\} \to \{5,6\} \to \{5,6,7\}$
$\mathsf{F}_{fl} = (000, \epsilon) \to (00, 1) \to (0, 10) \to (0, 11)$	$: \{1\} \to \{1,2\} \to \{1,2,3\}$
$\to (\epsilon, 100) \to (\epsilon, 101) \to (\epsilon, 110) \to (\epsilon, 111)$	$\to \{1,\ldots,4\} \to \ldots \to \{1,\ldots,8\}$
$\mathsf{F}_{fr} = (111, \epsilon) \to (11, 0) \to (1, 01) \to (1, 00)$	$: \{8\} \to \{7,8\} \to \{6,7,8\}$
$\to (\epsilon, 011) \to (\epsilon, 010) \to (\epsilon, 001)$	$\to \{5,\ldots,8\} \to \ldots \to \{2,\ldots,8\}$

Proposition 2.45. *The set* $\mathcal{F}_{KCT} = \{\mathsf{F}_s \mid s \in \{fl, fr, \{0,1\}^x\}, x = 0, \ldots, \log(n-2)\}$ *is a key-forest of the set system* Φ^{KCT} *with degree* 2.

Proof. Due to the definition of the sets F_{Lb} it is easy to observe that the key-forest consists of "upward" looking trees in the poset, i.e. it holds that a parent of a node is a subset of that node.

Consider now a subset $\mathsf{j} = (L, bR)$ for any $b \in \{0,1\}$ where $|LbR| = \log n$. We have two cases: (i) $L = \epsilon$; if $b = 0$ then S_j exists in the chain F_{fr}, if $b = 1$ then S_j exists in the chain F_{fl}.

(ii) Suppose that $L = L'\bar{b}^x$ for some $x \geq 0$ so that L' is either ϵ or ends with b. In case $L' = \epsilon$ holds, it is easy to see that the subset exists in the chain F_{fr} if $b = 0$ and it exists in the chain F_{fl} otherwise. We next claim that if $L' \neq \epsilon$, this subset belongs to the chain $\mathsf{F}_{L'}$ and not in any other chain. To prove the claim, we recall the following two facts:

- $\mathsf{F}_{L'}$ is a chain that starts with a leaf encoded by $(L'\bar{b}^{\log n - |L'|}, \epsilon)$ and grows one-by-one until all leaves are covered (except one) located in the subtree rooted at $v_{L'}$ in the following direction: (i) if $b = 0$ then from right to left, starting from $(L'1^{\log n - |L'|}, \epsilon)$ adding leaves whose string-indices represent smaller numbers. (ii) if $b = 1$ then from left to right, starting from $(L'0^{\log n - |L'|}, \epsilon)$ adding leaves whose string-indices represent larger numbers.
- Recall that $\mathsf{j} = (L, bR)$ is an encoding of a subset $\mathsf{S}_j = \{c, c+1, \ldots, d\}$ where (i) if $b = 0$ then $c = \mathsf{str2int}(LbR)$ and $d = \mathsf{str2int}(L1^{\log n - |L|}) = \mathsf{str2int}(L'1^{\log n - |L'|})$.
  (ii) if $b = 1$ then $c = \mathsf{str2int}(L0^{\log n - |L|}) = \mathsf{str2int}(L'0^{\log n - |L'|})$ and $d = \mathsf{str2int}(LbR)$.

Hence, we obtain the fact that the subset encoded as $\mathsf{j} = (L, bR)$ is contained in the tree $\mathsf{F}_{L'}$. Furthermore it can be easily verified that any other chain cannot contain j. The degree of the key-forest is 2 since the key-forest consists of chains. ■

Since the above key-forest has a degree of two, we can apply the key-assignment of Definition 2.21 by employing a 2-fold key extender $\mathsf{f}_2 : \mathsf{K} \mapsto \mathsf{K}^2$.

Observe that the trees with a maximum chain-length are F_{fl} and F_{fr}, hence the computation overhead is bounded by n function evaluations. The key storage for a user u is the number of trees in the forest $|\mathcal{F}_{KCT} \cap \mathsf{F}(u)|$ where $\mathsf{F}(u)$ refers to the filter of the user u in the key-poset. Recall that user u is given the label of the root of each tree in the intersection $|\mathcal{F}_{KCT} \cap \mathsf{F}(u)|$. We will count the number of trees by counting over each intersection $\mathsf{F}_{Lb} \cap \mathsf{F}(u)$:

1. if Lb is not a prefix of u; then $\mathsf{F}_{Lb} \cap \mathsf{F}(u)$ is empty.
2. if Lb is a prefix of u, then the intersection $\mathsf{F}_{Lb} \cap \mathsf{F}(u)$ is a subchain of F_{Lb}.

The second case happens for $\log n - 1$ different choices of $Lb \neq \epsilon$ as a prefix of u, (Lb can not be equal to u). Including also the subchains from F_{fl} and F_{fr}, the total key storage of a user would be $1 + \log n$.

Factorizability of the KCT set system.

We next show that the key chain tree set system satisfies the factorizability property and thus we ensure the existence of an efficient revocation algorithm.

Proposition 2.46. *The set system Φ^{KCT} is a factorizable set system.*

Proof. Our proof will take advantage of the alternative characterization for factorizability, the diamond property, given in definition 2.32 which is shown to be equivalent with factorizability (see Theorem 2.33)

Now consider two subsets $\mathsf{j}_1 = (L_1, b_1 R_1)$ and $\mathsf{j}_2 = (L_2, b_2 R_2)$ that are intersecting. We show that their union is in the set system. Since they are intersecting there is a common element u for which it holds that L_1 and L_2 are prefixes of u. If $L_1 = L_2$, then it is obvious that either one is covering the other in case $b_1 = b_2$, or their union corresponds to a subset in Φ^{KCT} that is containing all leaves rooted at node v_{L_1} in case $b_1 \neq b_2$.

Otherwise, without loss of generality say $|L_1| < |L_2|$ and we have that L_1 is a prefix of L_2. We have two cases: either (1) $L_1 b_1$ is not a prefix of L_2 or (2) $L_1 b_1$ is a prefix of L_2.

(1) This case implies that all leaves in the subtree rooted at v_{L_2} are included in the subset encoded by j_1, hence one is j_1 covers j_2.

(2) Suppose that $L_1 b_1$ is a prefix of L_2. We consider the cases (a) L_2 is a prefix of $L1 b_1 R_1$ and (b) L_2 is not a prefix of $L_1 b_1 R_1$.

- Case (a) : The union of $\mathsf{j}_1, \mathsf{j}_2$ can be expressed as the set $(L_1, b_1 R')$ where R' is defined by requiring that $(L_1 b_1 R', \epsilon)$ to be respectively the rightmost ($b_1 = 1$) or leftmost ($b_1 = 0$) leaf of the sets $\mathsf{j}_1, \mathsf{j}_2$.
- Case (b) : The only feasible arrangement of the sets $\mathsf{j}_1, \mathsf{j}_2$ is that j_2 is entirely contained in j_1.

With the above we showed that the union of the two subsets is always in the family. ■

Since the set system of Key Chain Tree method satisfies the factorizability property, our revocation algorithm given in Figure 2.9 applies and is optimal.

Transmission Overhead

Recall from the theorem 2.36 that the transmission overhead depends on the order of the lower maximal partition of an intersection $(S) \cap P_u$. The set system Φ^{KCT} satisfies the following:

Proposition 2.47. *For any subset* $\mathsf{S} \in \Phi^{KCT}$ *and an atom* u*, it holds that* $\mathsf{ord}((\mathsf{S}) \cap \mathsf{P}_u) \leq 3$.

Proof. First we give some intuition: any subset S encoded as $\mathsf{j} = (L, bR)$ corresponds to a consecutive sequence of leaves and will be splitted into two parts after removing a complement of an atomic filter P_u. One of the parts will contain the end leaf of the subset S, i.e., the leaf (LbR, ϵ), while the other part can be described as a union of two subsets in the set system. Totally, the ideal (S) will end-up with at most 3 ideals after removing the complement of an atomic filter. Let us see next a more detailed analysis of this.

Let S have an encoding $\mathsf{j} = (L, bR)$, if $\mathbf{mbr}(\mathsf{j}, u) = 0$, then the intersection $(\mathsf{S}) \cap \mathcal{P}_u$ itself equals to the ideal (S), hence the proposition holds. If not, then L would be a prefix of u. We have two cases: either (i) Lb is a prefix of u or (ii) Lb is not a prefix of u. In both of these cases, we will construct disjoint subsets $\mathsf{j}_1, \mathsf{j}_2, \mathsf{j}_3$ that are covering all the atoms in the intersection $(\mathsf{S}) \cap \mathsf{P}_u$, further it will be easy to observe that these subsets can not grow further, i.e. $\mathbf{cvr}(\mathsf{j}', i)$ would either contain u or goes outside the range of (S) for $\mathsf{j}' \in \{\mathsf{j}_1, \mathsf{j}_2, \mathsf{j}_3\}$ and $i = 1, 2$.

(i) In this case it holds that the leaves corresponding to u and (LbR, ϵ) are on the same half of the subtree rooted at v_L. Denote the longest common prefix of the string that encodes u and LbR by LbP where $P \in \{0, 1\}^x$ for some $x \geq 0$. Given that u belongs to (L, bR) it will hold that $LbP\bar{b}$ is prefix of u while $LbPb$ is prefix of LbR.

Suppose first that $b = 1$. Provided that u is not the leftmost leaf in the subtree rooted at v_L, the subset $\mathsf{S}_{\mathsf{j}_1} = \{\mathsf{str2int}(s_1), \ldots, u - 1\}$ (with $s_1 = L0^{\log n - |L|}$) contains all leaves located on the left of the leaf corresponding to u. To cover the remaining leaves we define the subsets $\mathsf{S}_{\mathsf{j}_2} = \{\mathsf{str2int}(s_2), \ldots, \mathsf{str2int}(L1R)\}$ where $s_2 = L1P10^{\log n - |LP| - 2})$ and $\mathsf{S}_{\mathsf{j}_3} = \{u + 1, \ldots, \mathsf{str2int}(s_3)\}$ where $s_3 = L1P01^{\log n - |LP| - 2}$. It is easy to verify that the subsets are not intersecting, they are members of the set systems and cover all leaves of (L, bR) except u. Note that it is possible that some of these subsets are empty; in such case there less than three subsets are needed to cover the leaves.

The case $b = 0$ is symmetric. Provided that u is not the rightmost leaf in the subtree rooted at v_L, the subset $\mathsf{S}_{\mathsf{j}_1}$ equal to $\{u + 1, \mathsf{str2int}(s_1)\}$ where $s_1 = L1^{\log n - |L|}$, contains all leaves located on the right of the leaf encoded by u. To cover the remaining leaves we define the subsets $\mathsf{S}_{\mathsf{j}_2} = \{\mathsf{str2int}(L0R), \ldots \mathsf{str2int}(s_2)\}$ where $s_2 = L0P01^{\log n - |LP| - 2}$ and $\mathsf{S}_{\mathsf{j}_3} = \{\mathsf{str2int}(s_3), \ldots, u - 1\}$ where $s_3 = L0P10^{\log n - |LP| - 2}$. As before it is easy to verify that these subsets are disjoint and cover all leaves of (L, bR) except u.

(ii) In this case it holds that the leaves corresponds to the indices u and LbR are on the different half of the subtree rooted at v_L.

Suppose first that $b = 1$. If u is not the leftmost leaf in the subtree rooted at v_L, the subset $\mathsf{S}_{\mathsf{j}_1}$ with an interval $[\mathsf{str2int}(s_1), u-1]$ (with $s_1 = L0^{\log n - |L|}$) contains all leaves located on the left of the leaf encoded by u. If $u+1$ has a prefix of $L1$ then the subset $\mathsf{S}_{\mathsf{j}_2} = \{\mathsf{str2int}(s_2), \ldots \mathsf{str2int}(LbR)\}$ where $s_2 = L10^{\log n - |L| - 1}$, will be enough to cover all remaining atoms in the intersection $(\mathsf{S}) \cap \mathsf{P}_u$. Otherwise we also define the subset $\mathsf{S}_{\mathsf{j}_3} = \{u+1, \mathsf{str2int}(s_3)\}$ where $s_3 = L01^{\log n - |L| - 1}$. We observe that none of these three subsets are intersecting and they all belong to the set system.

The case $b = 0$ is symmetric. If u is not the rightmost leaf in the subtree rooted at v_L, the subset $\mathsf{S}_{\mathsf{j}_1} = \{u+1, \ldots, \mathsf{str2int}(s_1)\}$ where $s_1 = L1^{\log n - |L|}$, contains all leaves located on the right of the leaf encoded by u. If $u-1$ has a prefix of $L0$ then the subset $\mathsf{S}_{\mathsf{j}_2} = \{\mathsf{str2int}(LbR), \ldots \mathsf{str2int}(s_2)\}$ where $s_2 = L01^{\log n - |L| - 1}$, will be enough to cover all remaining atoms in the intersection $(\mathsf{S}) \cap \mathsf{P}_u$. Otherwise we also define the subset $\mathsf{S}_{\mathsf{j}_3} = \{\mathsf{str2int}(s_3), \ldots, u-1\}$ where $s_3 = L10^{\log n - |L| - 1}$. As before it is easy to see that none of these three subsets are intersecting and that they all belong to the set system. ■

Applying Theorem 2.36 gives an upper bound on the transmission overhead for the key chain tree method, which yields an overhead of $2r+1$.

A direct revocation algorithm.

An direct revocation algorithm is also possible as follows. First we determine the contiguous sets of enabled users. Given $\mathsf{R} = \{\ell_1, \ldots, \ell_r\}$, we will get at most $r+1$ contiguous subsets of enabled users. One of them is possibly starting with the leftmost user and another one is possibly ending with the rightmost user. It is possible to cover any other consecutive set of users with at most 2 subsets of the key-chain tree set system. To see this consider a set of consecutive users $\mathsf{S} = \{u_1, \ldots, u_t\}$, and denote the least common ancestor of u_1 and u_t by node v of the binary tree of the set system. If u_1 is leftmost leaf or u_t is rightmost leaf of the subtree rooted at v, then it is already the case that S belongs to the set system. Otherwise, say u_i is the rightmost leaf of the left child of the node v for some i; this is guarranteed to exist due to the choice of v. It holds that u_{i+1} will be the leftmost leaf of the right-child of the node v. Hence, both $\{u_1, \ldots, u_i\}$ and $\{u_{i+1}, \ldots, u_t\}$ belong to the set system and they cover the initial subset S. This revocation algorithm yields a broadcast pattern with size at most $2r$.

2.6 Generic Transformations for Key Posets

A generic transformation for key posets is a technique of deriving a new key poset from an existing one. Such transformations have frequently the capability to improve the performance characteristics of the underlying set-systems. In this section we will see two such transformation techniques.

2.6.1 Layering Set Systems

In this section, we present a generic transformation of a set system to a new set system that supports a greater number of receivers. A k-layering of a set system over d receivers is a transformation that produces a set system over d^k receivers where k is an positive integer parameter. While the new set system supports an exponential in k increase in the number of receivers, the transmission overhead and the key storage will increase by a factor of only k while the computation overheadfor key decompression will remain the same. Such a change in efficiency parameters has different net effects depending on the underlying basic set system and may yield advantageous trade-offs between the transmission and computation overheads and the key storage needed at the receiver side.

Provided that the computation overhead for copmression of the basic set system is linear in number of receivers, the transformation results in a set-system such that the computation overhead is reduced by a $1/k$ root. Provided that transmission overhead and/or the key storage is linear in number of revoked users the parameters in the transformed system would increase by a multiplicative factor of k; in case the dependency is logarithmic in the number of receivers there would be no change. The usefulness of the transformation can be immediately evidenced by applying the transformation over the key chain tree set system of the previous section. This will yield a non-trivial trade-off between the transmission overhead and the computation overhead. Recall that the key chain tree set system enjoys a logarithmic key-storage and a transmission overhead that is linear in number of revoked receivers while it suffers from a computation overhead that is linear in number of receivers. The resulting set system would decrease the computation overhead by an $1/k$-th root while sacrificing somewhat the transmission overhead which will bear an increase by a multiplicative factor of k.

We describe the transformation in detail next. Let Φ^{BS_d} be the basic set system defined over a set $[d] = \{1, \ldots, d\}$ on which we apply the transformation. The k-layering of the basic set system is constructed by generating $\frac{d^k-1}{d-1}$ copies of the basic set system and connecting them in a top-down manner by merging a leaf of an upper-level set system with the root of a lower-level set system. We formally, define the transformation as follows:

Definition 2.48. *The transformation k-**LTrans**, given a key-poset of set system Φ^{BS_d} over $[d] = \{1, \ldots, d\}$, outputs a new key-poset of a set system over a set $[d^k] = \{1, \ldots, d^k\}$. The construction for the new key-poset will be as follows:*

1. *Generate $\frac{d^k-1}{d-1}$ copies of Φ^{BS_d}, and label each one with a string $s \in \{1, \ldots, d\}^x$ where $0 \leq x \leq k-1$. We denote the copy with a label s by $\Phi^{BS_d}_s$.*

2. *Replace the i-th leaf of the set system $\Phi_s^{BS_d}$ that corresponds to the user-index $i \in [d]$, for $s \in \{1, \ldots, d\}^x$, $0 \leq x \leq k-1$, with the top subset of the set system $\Phi_{si}^{BS_d}$.*

We denote the output of k-layering by $\mathbf{LTrans}^k(BS_d)$.

It is very intuitive in the key-poset framework to reflect the above transformation as depicted in the Figure 2.18.

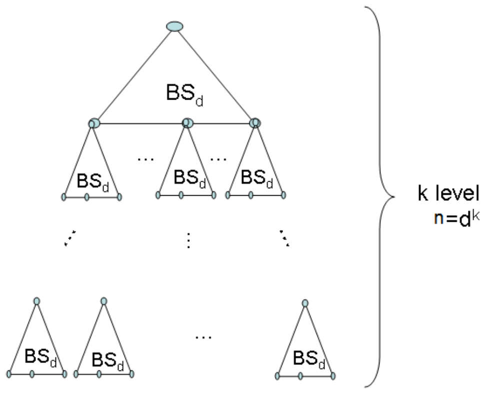

Fig. 2.18. Graphical depiction of the key-poset of the k-layering of a basic set system BS for d users.

The computational specification of the new set system $\mathbf{LTrans}^k(\Phi^{BS_d})$ in the KP framework can be defined based on the specification of the basic set system BS_d. Recall that $\mathbf{LTrans}^k(\Phi^{BS_d})$ consists of $\frac{d^k-1}{d-1}$ copies of the set system Φ^{BS_d}. The labeling of each copy can be comprehensible in the construction figure 2.18 by traversing the set systems top-down: the root set system of the figure is labeled by ϵ, the label of an i-th child set system is constructed by appending i to its parent's label. An encoding for the set system $\mathbf{LTrans}^k(\Phi^{BS_d})$ is a pair (s, j) where $\mathsf{j} \in \mathcal{J}^{BS_d}$ and $s \in \{1, \ldots, d\}^x$, $0 \leq x \leq k-1$, is the label of the set system the subset corresponding to the encoding (s, j) is located in. The algorithmic specification of the set system $\mathbf{LTrans}^k(BS_d)$ can be defined based on the specification of Φ^{BS_d} by taking the labels of the encodings into account: If the labels of two encodings are comparable, i.e. one is strict prefix of another, then the subset with longer label contains the other; if the labels are same, then whatever rules within the basic set system hold would be effective; if the labels are not comparable then the subsets corresponding to the encodings are disjoint. Since all six algorithms that we require in Definition 2.18 can be constructed without much effort, we will not explicitly state the descriptions in here and leave them as an exercise for the reader.

Similar to the algorithmic specification, the key-forest of the new set system can trivially be constructed by retrieving the key-forests of the set systems $\Phi_s^{BS_d}$ for $s \in \{1, \ldots, d\}^x$ with $0 \leq x \leq k-1$. The union of all these key forests constitutes the key forest of the new set system $\mathbf{LTrans}^k(BS_d)$.

Factorizability of the layering transformation.

Provided that the set system Φ^{BS_d} is factorizable, the layering transformation preserves the factorizability property for $n = d^k$ receivers. Hence, the new set system is accompanied with an efficient and optimal revocation algorithm due to the Theorem 2.35. We prove this in the next theorem.

Theorem 2.49. *Given that the fully exclusive set system Φ^{BS_d} defined over $\{1, \ldots, d\}$ is factorizable, then the transformation* **LTrans** *described in Definition 2.48 preserves the factorizability, i.e.* $\mathbf{LTrans}^k(\Phi^{BS_d})$ *is a factorizable set system defined over* $\{1, \ldots, d^k\}$.

Proof of Theorem 2.49: We will prove that $\mathbf{LTrans}^k(\Phi^{BS_d})$ is a factorizable set system by showing that it satisfies the diamond property, i.e. given any two intersecting subsets we will show that their union is in the new set system. Our proof will be based on the location of subsets that are intersecting:

Suppose, now, we have two subsets $\mathsf{S}_1 \in \mathbf{LTrans}^k(\Phi^{BS_d})$ and $\mathsf{S}_2 \in \mathbf{LTrans}^k(\Phi^{BS_d})$ so that $\mathsf{S}_1 \cap \mathsf{S}_2 \neq \emptyset$. If S_1 and S_2 are located in the same set system $\Phi_s^{BS_d}$ for some string $s \in \{1, \ldots, d\}^x$ with $0 \leq x \leq k-1$, then it will hold that $(\mathsf{S}_1 \cup \mathsf{S}_2) \in \mathbf{LTrans}^k(\Phi^{BS_d})$ due to the factorizability of the set system $\Phi_s^{BS_d}$. Otherwise, given that they are overlapping, it must be the case that one is covering the other and hence again their union is in the set system. ■

Efficiency parameters of the transformation.

We now come to the point to discuss the transmission overhead of a set system constructed by the layering transformation. Even though the transformation defines a new set system that increases by an exponential factor of k the size of the receiver population, it turns out that it increases the transmission overhead and the key storage by only a multiplicative factor of k. This will be observed in the following theorem that bounds the order of a lower-maximal partition for the intersection of any ideal with the complement of an atomic-filter. In the same theorem we also argue that the storage overhead will only suffer a multilplicative factor of k independently of the method that is used to assign the keys.

Theorem 2.50. *Suppose that a fully-exclusive set system Φ defined over a set $[d] = \{1, \ldots, d\}$ is factorizable.*

(i) If for any subset $\mathsf{S} \in \Phi$ and an atom $u \in \Phi$ it satisfies that $\mathsf{ord}((\mathsf{S}) \cap \mathsf{P}_u) \leq \mathsf{t}(d)$ *within the poset Φ for some function* $\mathsf{t}(\cdot)$, *then it holds that*

$\mathsf{ord}((\mathsf{S}') \cap \mathsf{P}_{u'}) \leq k \cdot \mathsf{t}(d)$ *for any* $\mathsf{S}' \in \mathbf{LTrans}^k(\Phi)$ *and any atom* $u' \in \mathbf{LTrans}^k(\Phi)$.

(ii) if there exists a key-assignment technique for Φ *such that the key-storage for each user is bounded by* $\mathsf{s}(d)$ *for some function* $\mathsf{s}(\cdot)$ *and the computation overhead is bounded by* $\mathsf{c}(d)$ *for some function* $\mathsf{c}(\cdot)$*, then there is a key-assignment technique that the key-storage in* $\mathbf{LTrans}(\Phi)$ *is bounded by* $k \cdot \mathsf{s}(d)$ *and computation overhead is bounded by* $\mathsf{c}(d)$.

Proof of Theorem 2.50: (i) If $u' \notin \mathsf{S}'$, then the proof is straightforward since $(\mathsf{S}') \cap \mathsf{P}_{u'} = (\mathsf{S}')$. Otherwise, consider the leaf-to-root path from u' to the root of the key-poset of the set system $\mathbf{LTrans}(\Phi)$. This path passes over a sequence of nodes that are at the leaf level of the component set systems, i.e., given that $u' \in [d^k]$ can be represented by a string $b_1 b_2 \ldots b_k$ with $b_i \in \{1, \ldots, d\}$ for $i \in [k]$, the path passes through the set systems $\Phi_{s_0}, \Phi_{s_1}, \ldots, \Phi_{s_{k-1}}$ where $s_0 = \epsilon$ and $s_i = b_1 \ldots b_i$ is a substring of the d-digit representation of u' with length i. It holds that S' belongs to Φ_{s_j} for some $j \in \{0, \ldots, k-1\}$. We observe next that the lower maximal partition of the intersection $(\mathsf{S}') \cap \mathsf{P}_{u'}$ will include $\mathsf{t}(d)$ sets for each one of the $k - j$ levels starting from the j-th level and going to the lowest one for a total of $(k - j) \cdot \mathsf{t}(d)$. This completes the proof of item (i).

(ii) The key-assignment for the new set system can be done by employing the key-assignment independently for each underlying set system. This will yield a key-storage for a user to be $k \cdot \mathsf{s}(d)$ since a user $u \in [d^k]$ with a d-digit representation of length k is related to k basic set systems in the new key-poset. On the other hand, the computation time will be same as if the set system is Φ, since the key of a subset is derivable within the basic set system that contains this subset. ■

Applying the layering transformation to the KCT set system.

We next give an example for the application of the layering transformation. The set system based on Key Chain Tree enjoys a transmission overhead that is same as the transmission overhead for the Subset Difference along with a logarithmic key storage while the Subset Difference method has a log-square key storage. On the other hand the Key Chain Tree method suffers from the computation overhead in the worst case which is linear in number of receivers.

Consider the set system Φ^{KCT} of key chain tree for d receivers. The transformation described in definition 2.48 will yield factorizable set systems over exponentially growing receiver populations that satisfy the statement of the theorem 2.50.

2.6.2 X-Transformation

In this section we describe another transformation of set systems that also preserves the factorizability property and has superior performance characteristics compared to the layering transformation. The transformation has no parameters and requires a certain property from the underlying set system, called the $\mathrm{X-property}$, that we will define herein. The number of users in the resulting set system equals the square of the number of users of the underlying set system while the transmission overhead increases by a constant amount while the storage approximately doubles. The transformation preserves a logarithmic computation overhead, i.e. given that the computation overhead of the basic set system is logarithmic in the number of receivers, the new computation overhead remains logaritmic in the new size of the receiver population.

In order to understand the transformation it will be helpful to think of the set system as the graph corresponding to the transitive reduction diagram of the corresponding poset. The X-property that is required by the basic set system is crucial in curbing the increase in the parameters of the set system while squaring the number of receivers. More formally, the property is defined as follows.

Definition 2.51. *We say a fully-exclusive set system* $(\Phi, \subseteq)$ *defined over a set* $\mathsf{M} = \{1, \ldots, 2^m\}$ *satisfies the* $\mathrm{X-property}$ *if*

1. There exist two elements $\mathsf{S}_1, \mathsf{S}_2 \in \Phi$ *so that* $\mathsf{F}(\mathsf{S}_1)$ *and* $\mathsf{F}(\mathsf{S}_2)$ *are disjoint full binary trees of height* $m-1$*. We denote them by* F^1_Φ *and* F^2_Φ *respectively.*

2. For each user $u \in \mathsf{M}$*, the complement of the atomic filter* $\mathsf{F}(u)$ *intersects with the above binary trees on a single path of length* m*; i.e.* $\mathsf{P}_u \cap (\mathsf{F}(\mathsf{S}_1) \cup \mathsf{F}(\mathsf{S}_2)) = \{\mathtt{u}^1_p, \ldots, \mathtt{u}^m_p\}$ *where* $\mathtt{u}^i_p$ *is a node in the key poset of the set system* Φ *that is a parent of* $\mathtt{u}^{i-1}$ *for* $i = 2, \ldots, m$*.*

Observe that due to the structure imposed by the $\mathrm{X-property}$ the top leaves of the binary tree filters $\mathsf{F}(\mathsf{S}_1)$ and $\mathsf{F}(\mathsf{S}_2)$ will be as many as the number of receivers in the set system. The second requirement of the $\mathrm{X-property}$ ensures that each leaf is uniquely related to an atom, and further it will hold that the leaves represent the subsets of the form $\mathsf{C}_u =_{df} \mathsf{M} \setminus \{u\}$.

The goal of the transformation is to expand the receiver population covered by the set system in a non-trivial way. This will be again through generating many copies of the basic set system and combining them in a non-trivial way. The first requirement in the $\mathrm{X-property}$ gives a way to combine a set system with an upper level copy and connecting non-trivial edges between those two set systems. Recall that the k-layering transformation doesn't support such interaction between the copies. The second requirement not only helps in supporting edge-transitions between upper and lower level copies but also makes it easier to employ the computational key assignment discussed in Definition 2.21. Such employment will make the transformation keep the computation overhead logarithmic while giving a reasonable increase in key storage. We next define the transformation formally.

Definition 2.52. *The transformation* **XTrans** *is a mapping over set systems that satisfy the* $\mathrm{X-property}$*. The transformation takes the key-poset of a set system* Φ *over* $\mathsf{M} = \{1, \ldots, 2^m\}$ *and outputs a new key-poset of a set system* *over* $\mathsf{M}' = \{1, \ldots, 2^{2m}\}$*. The construction for the new key-poset will be as follows:*

1. Generate 2^m *copies of* Φ*. We enumerate and denote the copies by* $\Phi_1, \Phi_2, \ldots, \Phi_{2^m}$*. We say* Φ_i *is defined over the set* $\mathsf{M}_i = \{(i-1)2^m+1, \ldots, (i-1)2^m + 2^m\}$ *while* Φ *will be called as the 'body' of the transformation.*

2. For $i = 1, \ldots, 2^m$ *generate a copy of* $\mathsf{F}^1_{\Phi_i}$ *and* $\mathsf{F}^2_{\Phi_i}$*, and denote them by* F^1_i *and* F^2_i *respectively. These collections should be thought as graphs, while their subset content will be determined in the remaining steps of the transformation.*
Constructing the 'feet' of the transformation:

3. Replace the leaf $\{v\} \in \Phi$ *that corresponds to the user* $v \in \mathsf{M}$ *with the leading subset of the set system* Φ_v*. We will name these parts as the feet of the new set system.*
Constructing the 'head' of the transformation:

4. Remove the edges outgoing from the leading subset of Φ*.*

5. For any $v \in \mathsf{M}$*, add edges* $\mathsf{C}_v \subset \mathsf{F}^1_v$ *and* $\mathsf{C}_v \subset \mathsf{F}^2_v$*. (here* F^1_v *and* F^2_v *represent the roots of the corresponding binary trees)*

6. Connect the leading subset of Φ *with the leaves of binary tree* F^b_v *for all* $v \in \mathsf{M}$ *and* $b \in \{1, 2\}$*.*
Finally :

7. For any $v \in \mathsf{M}$ *and* $b \in \{1, 2\}$*, add an edge between* S *and* S' *where* $\mathsf{S}' \in \mathsf{F}^b_v$ *is the copy of the subset* $\mathsf{S} \in \mathsf{F}^b_{\Phi_v}$*. Recall that* F^b_v *and* $\mathsf{F}^b_{\Phi_v}$ *are isomorphic to each other.*

We denote the output of the X-Transformation by $\mathbf{XTrans}(\Phi)$*.*

The above transformation is illustrated in Figure 2.19. Note that the depiction lacks the representation of step 7 (as the inclusion would make it rather hard to read - for a complete example the reader is referred to the example in the end of this section).

We will now prove that the transformation preserves the $\mathrm{X}-$property.

Theorem 2.53. *The transformation* **XTrans** *described in definition 2.52 preserves the* $\mathrm{X}-$property*.*

Proof of Theorem 2.53: To begin with, it is easy to observe that the new set system Φ' is defined over the set $\mathsf{M}' = \{1, 2, \ldots, 2^{2m}\}$ and it is fully-exclusive; indeed, the new key poset is connected and for each $u \in \mathsf{M}'$ there exists a corresponding node in the key poset. It also holds that the subsets in the head of the new key poset are valid subsets in the new set system since each has an incoming edge from a subset in the feet of the key poset.

We prove next that the new key-poset satisfies the two properties required in Definition 2.51. We will use the following argument: if a subset S in the has incoming edges from s different subsets $\mathsf{S}_1, \ldots, \mathsf{S}_s$, then it holds that $\mathsf{S} = \bigcup_{i=1}^{s} \mathsf{S}_i$.

1. Recall that there exist two elements $\mathsf{S}_1 \in \Phi$ and $\mathsf{S}_2 \in \Phi$ such that $\mathsf{F}(\mathsf{S}_1)$ and $\mathsf{F}(\mathsf{S}_2)$ are disjoint full binary trees of height $m-1$ so that C_v is a leaf of one of these binary trees for any $v \in \mathsf{M}$ (this is ensured by the second

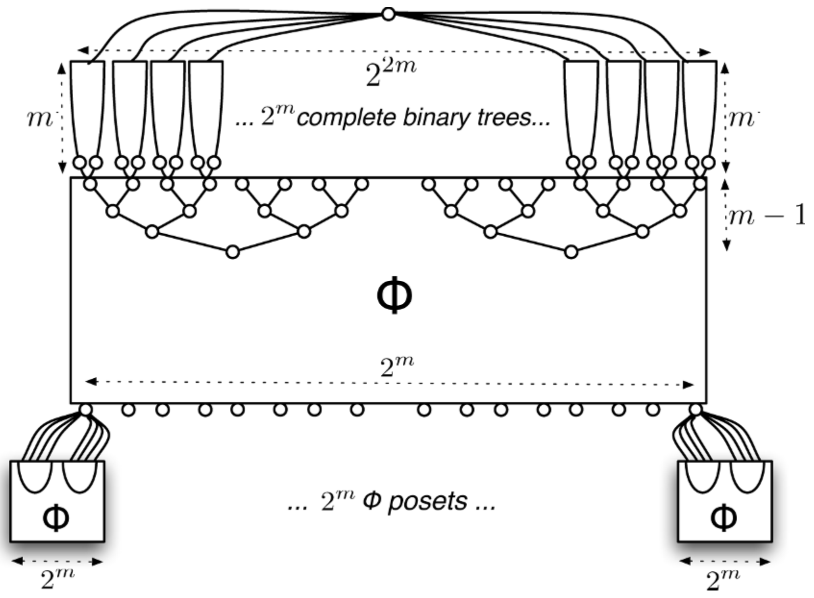
2^{2m}
m
... 2^m complete binary trees...
m
$m-1$
Φ
2^m
... 2^m Φ posets ...
Φ
Φ
2^m
2^m

Fig. 2.19. The transformation of definition 2.52 (note that the illustration does not include the connections described in step number 7).

requirement of the X − property of the set system Φ). We will show that the filters of the same subsets are the required filters we need to demonstrate their existence of for the set system Φ'. It is easy to see that $\mathsf{F}(\mathsf{S}_1)$ and $\mathsf{F}(\mathsf{S}_2)$ within the new set system Φ' are full binary trees of height $2m-1$. This is true as for each $v \in \mathsf{M}$, the node $\mathsf{C}_v \in \Phi_b$ was connected (see the 5-th step of the transformation) to two full binary trees of height $m-1$; it follows that $\mathsf{F}(\mathsf{S}_1)$ and $\mathsf{F}(\mathsf{S}_2)$ within Φ' are binary trees of height $m-1+1+m-1 = 2m-1$.

2. For any index $u \in \mathsf{M}'$, there exists an integer $v \in \{1,\ldots,2^m\}$ so that $u \in \mathsf{M}_v$. Since Φ_v satisfies the X−property, there is a unique path $\{u_v^1,\ldots,u_v^{m-1}\}$ not including the atom u that is intersecting with the either the filter $\mathsf{F}^1_{\Phi_v}$ or $\mathsf{F}^2_{\Phi_v}$. Since those filters that are structurally upward full binary trees are copied (refer to the second step of transformation) to the head of the resulting set system we can discover an isomorphic copy of this path that we denote by $\{u_{v*}^1,\ldots,u_{v*}^m\}$. Note that by definition the subsets corresponding to u_{v*}^i for $i = 1,\ldots,m$ exclude the atom u.

Recall now that $v \in \mathsf{M}$, hence there exists a unique path within one of the two filters of the body set system Φ. The subsets corresponding to the nodes over this path do not contain the node v which is atomic in the body while it is the leading subset of the lower level set system Φ_v. Hence, this path does not contain the receiver indexed by $u \in \mathsf{M}'$. Moreover, the path ends with the node corresponding to the subset $\mathsf{C}_v = \mathsf{M} \setminus \{v\}$. Recall that in the 5-th step of the transformation, we connected the set C_v with the filters F_v^1 and F_v^2. Observe now that the path within the body that excludes v and the path in the extension (the one denoted by $\{u_{v*}^1, \ldots, u_{v*}^m\}$) on the head are connected and all corresponding subsets exclude u. This results to a unique path of length $2m$ in the transformed set system. ■

The computational specification of the resulting set system $\mathbf{XTrans}(\Phi)$ in the KP framework can be described based on the specifications of the basic set system Φ defined over $\{1, \ldots, 2^m\}$. Recall that $\mathbf{XTrans}(\Phi)$ generates 2^m copies of the set system Φ. An encoding for the set system $\mathbf{XTrans}(\Phi)$ is a triple (b, s, j) where $\mathsf{j} \in \mathcal{J}^{\Phi}$ and $s \in \{\epsilon\} \cup \{0,1\}^m$ is the label of the set system the node corresponding to the encoding j is located in while $b \in \{0,1\}$. In case, $b = 0$ then the subset j is located in one of the set systems $\Phi, \Phi_1, \ldots, \Phi_{2^m}$, while in case $b = 1$ then the subset encoded by $\mathsf{j} \in \mathcal{J}^{\Phi}$ is a copy of a subset in the filter and it is located on the head of the transformed set system. Note that not all triples (b, s, j) correspond to a legal subset. The algorithmic specification of the set system $\mathbf{XTrans}(\Phi)$ can be defined based on the specification of Φ by taking the labels of the encodings into account. Since all six algorithms we require in Definition 2.18 can be constructed without so much effort, we will not explicitly state the algorithms in here and leave this task as an exercise for the reader.

An important characteristic of X transformation is that, it is respecting the key-forest key compression approach as it was defined in Section 2.3.3. Given that the underlying set system Φ has a key-forest F_{Φ}, the key forest of the set system $\Phi' = \mathbf{XTrans}(\Phi)$ is the union of the key-forests of the underlying building set systems while the binary trees on the head can be appended to corresponding trees of the key-forest defined for the body of the extended family Φ'. This means that it is possible to maintain a small overall number of trees in the key-forest as the transformation is applied repetitively. For an example of this the reader is referred to the explicit instantiation of an X-transformation shown in the end of this section.

Factorizability of the X-Transformation.

We will prove next that the transformation described in definition 2.52 preserves the factorizability of the underlying set system.

Theorem 2.54. *Given that the fully exclusive set system Φ defined over $\{1, \ldots, 2^m\}$ is factorizable and further satisfies the* X − property*, then it holds that the set system* $\mathbf{XTrans}(\Phi)$ *defined over $\{1, \ldots, 2^{2m}\}$ is also factorizable.*

Proof of Theorem 2.54: We will prove that $\mathbf{XTrans}(\Phi)$ is a factorizable set system by showing that it satisfies the diamond property, i.e. given any two intersecting subsets we will show that their union is in the new set system. Our proof will be based on the location of subsets that are intersecting:

Suppose, now, we have two subsets $\mathsf{S}_1 \in \mathbf{XTrans}(\Phi)$ and $\mathsf{S}_2 \in \mathbf{XTrans}(\Phi)$ so that $\mathsf{S}_1 \cap \mathsf{S}_2 \neq \emptyset$. If S_1 and S_2 are in the same Φ component of the new key-poset, it will hold that $(\mathsf{S}_1 \cup \mathsf{S}_2) \in \mathbf{XTrans}(\Phi)$ due to the factorizability of the set system Φ. Otherwise, we have three cases:

1. S_1 is located on the head, and S_2 is located in the body. It holds that S_1 extends $\mathsf{C}_i \in \Phi$ for some $i \in \{1, \ldots, 2^m\}$. Let S_2^* be the subset of Φ that corresponds to S_2. If $i \in \mathsf{S}_2^*$, this means that S_2 contains all atoms of Φ_i while on the other hand S_1 contains all other atoms (as it extends C_i). It follows that $\mathsf{S}_1 \cup \mathsf{S}_2 = \mathsf{M}'$, hence their union is the top subset of the set system which belongs to $\mathbf{XTrans}(\Phi)$. On the other hand, if $i \notin \mathsf{S}_2^*$ then it holds that $\mathsf{S}_2 \subset \mathsf{S}_1$.

2. S_1 is located on the head, and S_2 is located on the feet. It holds that S_1 is in a binary tree that extends $\mathsf{C}_i \in \Phi$ for some $i \in \{1, \ldots, 2^m\}$. If $\mathsf{S}_2 \in \Phi_i$, then there is a corresponding copy of S_1 within Φ_i say called $\mathsf{S}_3 \in \Phi_i$. It is easy to see that $\mathsf{S}_2, \mathsf{S}_3$ are also intersecting and hence the union of $\mathsf{S}_3 \cup \mathsf{S}_2$ exists within Φ_i. The union is also in the filter of S_3, hence it holds that there would be a corresponding element of the union-subset on the head of new key-poset $\mathbf{XTrans}(\Phi)$ that will contain S_1. On the other hand, if S_2 is located on the set system $\Phi_{i'}$ for some leaf $i' \neq i$, then it holds that $\mathsf{S}_2 \subset \mathsf{S}_1$.

3. S_1 is located in the body, and S_2 is located on the feet: it holds that S_2 is located on the set system Φ_i for some leaf $\{i\} \in \Phi$. Let S_1^* be the subset of Φ that corresponds to S_1. Since $\mathsf{S}_1 \cap \mathsf{S}_2 \neq \emptyset$ then it holds that $i \in \mathsf{S}_1^*$. This suggests that $\mathsf{S}_2 \subset \mathsf{S}_1$. ■

Efficiency parameters of the transformation.

We now come to the point to discuss the transmission overhead of the set system constructed by the X-transformation. Provided that the set system satisfies the $\mathrm{X-property}$, even though the transformation defines a new set system that squares the size of the receiver population, it increases the transmission overhead by a constant factor only. This can be observed by the following theorem on bounding the order of a lower-maximal partition for the intersection of any ideal with the complement of an atomic-filter. At the same time we show how the key storage and computation overhead are affected by the transformation.

Theorem 2.55. *Suppose that a fully-exclusive set system Φ defined over a set $\mathsf{M} = \{1, \ldots, 2^m\}$ is factorizable and further satisfies the* $\mathrm{X-property}$.

(i) If for any subset $\mathsf{S} \in \Phi$ and an atom $u \in \Phi$ it holds that $\mathsf{ord}((\mathsf{S}) \cap \mathsf{P}_u) \leq \mathsf{t}(m)$ within the poset Φ for some function $\mathsf{t}(\cdot)$, then it holds that $\mathsf{ord}((\mathsf{S}') \cap \mathsf{P}_{u'}) \leq \mathsf{t}(m) + 2$ for any $\mathsf{S}' \in \mathbf{XTrans}(\Phi)$ and any atom $u' \in \mathbf{XTrans}(\Phi)$.

(ii) If there exists a key-assignment technique for Φ such that the key-storage for each user is bounded by $\mathsf{s}(m)$ for some function $\mathsf{s}(\cdot)$ and the computation overhead is bounded by $\mathsf{c}(m)$ for some function $\mathsf{c}(\cdot)$, then there is a key-assignment technique so that the key-storage in $\mathbf{XTrans}(\Phi)$ is bounded by $2\mathsf{s}(m) + m$ and computation overhead is bounded by $\max(\mathsf{c}(m), m)$.

Proof of Theorem 2.55: (i) On transmission overhead of the transformation:

For any leaf $u' \in \mathbf{XTrans}(\varPhi)$ there exists a leaf v' in the body which is replaced with the set system $\varPhi_{v'}$ that satisfies $(v'-1)2^m + 1 \leq u' \leq v' \cdot 2^m$. Denote the leading subset of $\varPhi_{v'}$ by (v'). First recall that the subset $\mathsf{C}^{v'}_{u'} =_{df} (v') \setminus \{u'\}$ exists in the set system $\varPhi_{v'}$ located in the feet of the transformation. We have three cases depending on the location of $\mathsf{S}' \in \mathbf{XTrans}(\varPhi)$:

1. S' is located on the head: it holds that S' is in a binary tree extending $\mathsf{C}_v \in \varPhi$ for some $v \in \{1, \ldots, 2^m\}$, recall that we define $\mathsf{C}_v = \mathsf{M} \setminus \{v\}$ and S' is corresponded to a subset in $\varPhi_v$; we will denote its corresponding subset by S'^{*}. We have two subcases:
(i) if $v = v'$: Consider now the lower-maximal partition $\langle \mathsf{I}_1, \ldots, \mathsf{I}_o \rangle$ of $(\mathsf{S}'^{*}) \cap \mathsf{P}_{u'}$ within the set system $\varPhi_{v'}$ (here $\mathsf{P}_{u'}$ is the complement of the atomic filter u' in key-poset $\varPhi_{v'}$); due to the factorizability of the set system and the theorem's statement it holds that the lower maximal partition satisfies $o \leq \mathsf{t}(m)$. Consider now the intersection $(\mathsf{S}') \cap \mathsf{P}_{u'}$ within the set system $\mathbf{XTrans}(\varPhi)$; it will hold that the users of $(\mathsf{S}') \cap \mathsf{P}_{u'}$ can be covered in the worst case by $\mathsf{C}_v \cup (\cup_{j=1}^{o} \mathsf{I}_j)$ (it might be possible that $\mathsf{C}_v \cup \mathsf{I}_j$ exists in the new set system for some j something that would give an even smaller cover). In any case, $\mathsf{ord}((\mathsf{S}') \cap \mathsf{P}_{v'}) \leq \mathsf{t}(m) + 1$.
(ii) if $v \neq v'$: The users of the set $(\mathsf{S}') \cap \mathsf{P}_{u'}$ can be covered by the sets $\mathsf{S}'^{*} \cup \mathsf{C}^{v'}_{u'}$ as well as the users in the intersection $(\mathsf{C}_v \cap \mathsf{P}_{v'})$; note that the latter intersection is taken within the body of the transformation. Hence, in any case the cover of $(\mathsf{S}') \cap \mathsf{P}_{u'}$ will number at most $\mathsf{ord}((\mathsf{S}') \cap \mathsf{P}_{u'}) \leq \mathsf{t}(m) + 2$ subsets.

2. S' is located in the body: Provided that $u' \in \mathsf{S}'$ (otherwise the intersection is S' itself), it holds that $v' \in \mathsf{S}'$ within the set system $\varPhi$ and hence it holds that the users of $(\mathsf{S}') \cap \mathsf{P}_{u'}$ can be covered by $\mathsf{C}^{v'}_{u'} \cup (\mathsf{S}' \cap \mathsf{P}_{v'})$ where the latter intersection is taken within the body of the transformation. It follows that $\mathsf{ord}((\mathsf{S}') \cap \mathsf{P}_{u'}) \leq \mathsf{t}(m) + 1$.

3. S' is located on the feet of the transformation: Suppose S' is located in the set system $\varPhi_v$ for some $v \in \{1, \ldots, 2^m\}$. If $v \neq v'$ then the intersection is of the ideal (S') and the complement of the atomic filter $\mathsf{F}(u')$ would be (S') itself. Otherwise S' is located within the set system with u' for which the intersection yields a lower maximal partition of order less than or equal to $\mathsf{t}(m)$ due to the factorizability of the set system $\varPhi$.

(ii) Dealing with the body and feet part of the transformation is relatively easy. We will employ the key assignment technique of the basic set system for the set system of the body $\varPhi$ as well as $\varPhi_1, \varPhi_2, \ldots, \varPhi_{2^m}$ used in the transformation. Observe that the user would need to store exactly $2 \cdot \mathsf{s}(m)$ keys (one set of keys for the component $\varPhi_i$ it belongs to and one set of keys for $\varPhi$) and in order to derive any key in the head and body part would require computation overhead bounded by $\mathsf{c}(m)$.

With respect to the subsets located on the head of the transformation no extra keys would be needed. We will use the structure of the binary trees and the computational key compression method of described in definition 2.21.

Observe that a user u is able to derive the keys for each subset $\mathsf{C}_{u'}$ in the body. Each subset C_u is the root of an upward binary tree for which the user requires all or some of the keys. Specifically, based on the X-property there is a single binary tree for which the user u requires only a subset of the keys and in fact he would need exactly m keys as it needs a key for each sibling of a node in the path that excludes u. Hence, employing a 3-fold key extender to derive the keys for all the nodes in the binary trees using the keys of the C_u sets as the locals a user needs to store m extra keys (the labels of the nodes that are siblings of the nodes in the unique path that excludes the user) and thus the computation overhead is bounded by m. It follows that the key storage is bounded by $2\mathsf{s}(m) + m$. The computation overhead is bounded by the maximum of $\mathsf{c}(m)$ and m. ■

Instantiation for X-Transformation

Theorem 2.55 proves that the X-transformation of a set system squares the size of the population, while the efficiency parameters increase only linearly. This motivates us to apply successive applications of the transformation described in Definition 2.52 over a fully-exclusive set system.

Definition 2.56. *Let $\Phi^{(d)}$ be a factorizable set system defined over a set $[d] = \{1,\ldots,d\}$ and further satisfies $\mathrm{X-property}$. We define $\mathrm{AS}^k_{\Phi(d)}$ as the fully-exclusive factorizable set system defined over a set of $n = d^{2^k}$ users where $\mathrm{AS}^j_{\Phi(d)} = \mathbf{XTrans}(\mathrm{AS}^{j-1}_{\Phi(d)})$ for $j = 1,\ldots,k$ with $\mathrm{AS}^0_{\Phi(d)} = \Phi^{(d)}$.*

Theorem 2.36 and Theorem 2.55 implies the following corollary which provides an upper bound on the transmission overhead of the set system $\mathrm{AS}^k_{\Phi(d)}$ and the efficiency of the key-assignment technique.

Corollary 2.57. *Let $n = d^{2^k}$ for some $d, k \in \mathbb{N}$ and $\Phi^{(d)}$ is a factorizable set system defined over a set of size d that satisfies the $\mathrm{X-property}$. Suppose that for any subset $\mathsf{S} \in \Phi^{(d)}$ and an atom u of $\Phi^{(d)}$ it holds that $\mathsf{ord}((\mathsf{S}) \cap \mathsf{P}_u) \leq \mathsf{t}(d)$ within the poset Φ, the key storage is bounded by $\mathsf{s}(d)$ and the computation overhead is bounded by $\mathsf{c}(d)$ for some functions $\mathsf{t}(\cdot), \mathsf{s}(\cdot)$ and $\mathsf{c}(\cdot)$. Consider now the set system $\mathrm{AS}^k_{\Phi(d)}$ over n users:*

1. If $d = O(1)$, then the transmission overhead to disable a set of r users in the set system $\mathrm{AS}^k_{\Phi(d)}$ would be $O(r(2\log\log n))$, the key storage is $O(\log\log n \cdot \log n)$ and the computation overhead would be $O(\log n)$.

2. If $k = O(1)$, then the transmission overhead to disable a set of r in the set system $\mathrm{AS}^k_{\Phi(d)}$ would be $r(2k + \mathsf{t}(d))$, the key storage is $2^k \cdot \mathsf{s}(d) + \frac{k \log n}{2}$ and the computation overhead would be $\max(\mathsf{c}(d), \log n)$.

Proof of Corollary 2.57: Due to the theorem 2.55 the order of a lower maximal partition of $(\mathsf{S} \cap \mathsf{P}_u)$ for any subset $\mathsf{S} \in \mathrm{AS}^k_{\Phi(d)}$ and any atom $u \in [d^{2^k}]$ can be bounded by incrementing $\mathsf{t}(d)$ with $2k$.

Recall that the Theorem 2.36 relates the transmission overhead of a set system with the upper bound on the lower maximal partition order for the intersection of an ideal with a complement of an atomic filter. Hence, we obtain the transmission overhead for the ciphertext revoking r users is bounded by $r(2k+\mathsf{t}(d)-1)+1$. That would conclude the case when k is a constant. While on the other hand, if d is a constant then we obtain $r(\log\log n)$.

Regarding the key storage and the computation overhead, we successively apply the bounds given in theorem 2.55 to obtain for $i = 1, \ldots, k$:

$$\mathsf{s}_i = 2\mathsf{s}_{i-1} + 2^{i-1}\log d$$

where $\mathsf{s}_0 = \mathsf{s}(d)$. By applying a telescopic sum over i values we will obtain $\mathsf{s}_k = 2^k\mathsf{s}(d) + k2^{k-1}\log d = 2^k\mathsf{s}(d) + \frac{k\log n}{2}$. That would conclude the case when k is a constant. In case d is constant we have $k = \log\log n$, hence the key storage is $O(\log\log n \cdot \log n)$. ■

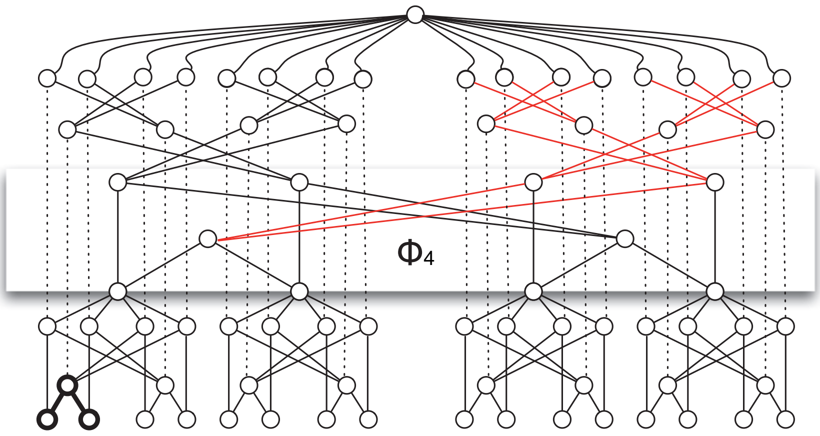

Fig. 2.20. The X-transformation over the set system $\Phi_4 = \mathrm{AS}^1_{\Phi_{\{1,2\}}}$.

We will next give an example of an AS set system and an application of the transformation technique above. We first start with a set system $\Phi_{\{1,2\}} = \{\{1,2\},\{1\},\{2\}\}$ that is the power set (without the empty set) of the set $\{1,2\}$. Observe that this basic set system satisfies the X − property and it is factorizable set system. Hence, the transformation described in definition 2.52 will yield factorizable set systems over exponentially growing receiver populations that satisfy the statement of the corollary 2.57. Observe that the resulting set system $\mathrm{AS}^2_{\Phi_{\{1,2\}}}$ is defined over a receiver population of size 16. The figure 2.20 illustrates the resulting set system after applying the transformation twice on the above simple set system.

The key-forest of the above construction is depicted in Figure 2.21. Regarding the parameters of the set system $\mathrm{AS}_{\Phi^k_{\{1,2\}}}$ that is the factorizable set sytem defined over a set of size $n = 2^{2^k}$, we have the following immediate corollary.

Corollary 2.58. *Consider the broadcast encryption scheme based on the set system* $\mathrm{AS}^k_{\Phi_{\{1,2\}}}$ *where* $k = \log\log n$*; the transmission overhead for the ciphertext revoking* r *users is* $O(r \log\log n)$*. The key-storage for each receiver is* $O(\log\log n \cdot \log n)$*, and the computation overhead for a receiver is bounded by* $\log n$*.*

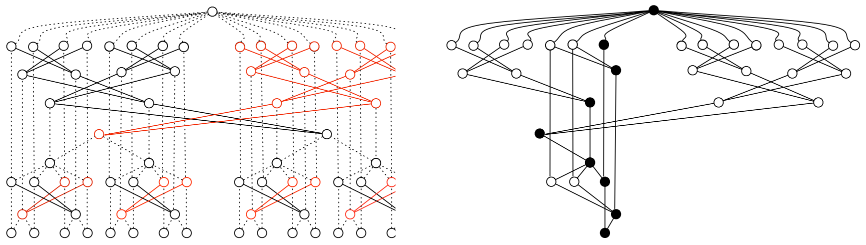

Fig. 2.21. (left) the key-forest of the set system $\mathrm{AS}^2_{\Phi_{\{1,2\}}}$. The edges define the trees in the key-forest. (right) the filter for a specific user, the black nodes represent the roots of the trees in the intersection of the key-forest and the filter.

2.7 Bibliographic notes

The concept of broadcast encryption was introduced in the work by Berkovits [10]. Combinatorial broadcast encryption schemes can be constructed by employing probabilistic techniques or with explicit combinatorial constructions. Examples of this category include the first paper [42] that introduced the first formal construction of a broadcast encryption and employed a probabilistic design as well as others such as [87, 53] that employed explicit combinatorial constructions such as the ones we focused on in this chapter. Recall that *structured* broadcast encryption schemes utilize the key-space so that it has some structure that enables the preparation of ciphertexts that are decipherable only by the enabled users. Examples of this category include schemes that are based on polynomial interpolation [90, 37] where keys are points of a polynomial over a finite field and schemes based on bilinear maps such as [18] where the discrete-logarithms of the keys over an elliptic curve group are different powers of the same base.

Given that the intended application of broadcast encryption is content distribution, it is expected that the scheme would handle large messages. This is solved in our exposition through hybrid encryption and recall that this approach requires the broadcast encryption scheme to implement a “Key Encapsulation Mechanism” (KEM). KEM is introduced by Shoup [108] in the context of public key encryption. We further require the underlying cryptographic primitive to support the CCA1 type of encryption security, or what is known as a security against lunch-time attacks [91].

Regarding combinatorial constructions the underlying key-space can be structured as an exclusive set system. Some constructions of exclusive set systems for different parameters are known [4, 117, 83] along with some existence results (see [77]) using the probabilistic method. Various tools have been used for explicit constructions including polynomials over finite fields, [49], and algebraic-geometric codes [76].

The subset cover framework introduced by Naor, Naor and Lotspiech in [87] as a wide class of exclusive set systems with some specific properties. [87] also proposed the two schemes of Complete Subtree (CS) and Subset Difference (SD). The idea in both of these schemes was to locate the receivers on the leaves of a binary tree. The CS is related to the logical key hierarchy (LKH) that was proposed independently by Wallner et al. [121] and Wong et al. [125], for the purpose of designing a key distribution algorithm for Internet multicasting.

The applicability of this framework is also evidenced by the fact that a subset cover scheme (a simple variant of the subset-difference method) is at the heart of the AACS standard [1] that is employed by high definition DVDs (Blu-Ray and HD-DVD). The Advanced Access Content System (AACS) is a standard for content distribution and digital rights management, intended to restrict access to and copying of the next generation of optical discs and DVDs. The specification was publicly released in April 2005 and the standard has been adopted as the access restriction scheme for Blu-ray (BD) discs. It is developed by AACS Licensing Administrator, LLC (AACS LA), a consortium that includes Disney, Intel, Microsoft, Matsushita (Panasonic), Warner Bros., IBM, Toshiba and Sony. In this particular application, the distribution channel is an optical disc or DVD. The decoders required to playback the content are either hardware DVD-players or software video-players. The decoders have embedded the necessary secret-key information at the manufacturing stage.

In a broadcast encryption scheme, all unlicensed receivers must be excluded in the broadcast pattern. The transmission overhead of a broadcast encryption can be further reduced in some settings when some of the unlicensed receivers are allowed to continue receiving the transmission. Abdalla et al. in [3] introduced the concept of *free riders* to capture this observation and investigate the performance gains. Ramzan and Woodruff [97] proposed an algorithm to optimally choose the set of free riders in the CS system, and Ak, Kaya and Selcuk in [5] presented a polynomial time algorithm which computes the optimal placement for a given number of free riders in an SD scheme.

After the introduction of Subset Cover Framework by Naor, Naor and Lotspiech in [87], this combinatorial framework gave rise to a diverse number of fully-exclusive set systems: Complete Subtree and Subset Difference [87], the Layered Subset Difference [53], the Key Chain Tree [122], the Stratified Subset Difference [50], the Subset Incremental Chain [6] and the class of set systems of [55] as well as that of [56, 57].

The Key Chain Tree [122], the Stratified Subset Difference [50] and the Subset Incremental Chain [6] are all employing the same set system with slight differences in their key-handling procedures. Those differences result in different efficiency parameters but still they utilize the exact same key-poset. It is quite easy to observe that [6] and [122] have the same key-poset whereas it is more involved to see the isomorphism of the set system Stratified Subset Difference ([50]) to the former two without explicitly thinking of the key-poset of [50]. We refer the reader to [105] for more information on partial ordered sets.

As a short diversion within this short set of bibliographic notes, we will show that, indeed, the key-poset of the Stratified Subset Difference is isomorphic to the key-poset of Key Chain Tree. In a nutshell, the set system $\varPhi^{SSD}$ of [50] consists of subsets that are formed as follows. Consider a subset of a SD method, and denoted by S_j where $\mathsf{j} = (v_i, v_k)$ is a pair of nodes in the binary tree that has leaves the set of users. The set $\mathsf{S}_\mathsf{j}^L \in \varPhi^{SSD}$ is defined as the set of nodes rooted at v_i and located on the left of the nodes rooted at v_k. The set $\mathsf{S}_\mathsf{j}^R \in \varPhi^{SSD}$ is defined as the set of nodes rooted at v_i and located on the right of the nodes rooted at v_k. Observe that both subsets S_j^L and S_j^R consist of continuous nodes at the leaf level in the subtree rooted at node v_i.

Theorem 2.59. *The key poset of Stratified Subset Difference is equal to the key poset stemming from the schemes of [6, 122] that is depicted in figure 2.16.*

Proof. of Theorem 2.59 It is quite obvious that any subset described in the stratified subset difference exists in the set system of Key Chain Tree. We will, now, complete the proof by showing that for any subset $\mathsf{S}_{\mathsf{j}^k} \in \varPhi^{\mathtt{KCT}}$ there exists some $\mathsf{S}_{\mathsf{j}^s} \in \varPhi^{\mathtt{SSD}}$ that satisfies $\mathsf{S}_{\mathsf{j}^k} = \mathsf{S}_{\mathsf{j}^s}$:

We have two cases for the location of leaves in $\mathsf{S}_{\mathsf{j}^k} = \{u_1, \dots, u_t\}$:

(i) $\mathsf{S}_{\mathsf{j}^k}$ is set of consecutive leaves rooted at a minimal node v_1 so that u_1 is the leftmost leaf of v_1. Denote the least common ancestor of u_t and u^* by v_2 where u^* comes after the leaf u_t in the left-to-right ordering of the leaves. Consider the right child of v_2 and name it by v_3. Observe now that the subset $\mathsf{S}_{\mathsf{j}^s}^L$ for $\mathsf{j}^s = (v_1, v_3)$ contains all leaves rooted at v_1 and located on the left of the nodes rooted at v_3.

(ii) $\mathsf{S}_{\mathsf{j}^k}$ is set of consecutive leaves rooted at a minimal node v_1 so that u_t is the rightmost leaf of v_1. Denote the least common ancestor of u_1 and u^* by v_2 where u^* comes before the leaf u_1 in the right-to-left ordering of the leaves. Consider the left child of v_2 and name it by v_3. Observe now that the subset $\mathsf{S}_{\mathsf{j}^s}^R$ for $\mathsf{j}^s = (v_1, v_3)$ contains all leaves rooted at v_1 and located on the right of the nodes rooted at v_3.

Hence, in any case, there will be a corresponding subset in the Stratified Subset Difference for each subset in the Key Chain Tree. ■

In the formalization of the Key-Poset framework we introduced in this chapter, the subset cover families of [6, 50, 87, 122] are all factorizable (see Section 2.5) and thus the results we laid out provide a unified way for solving revocation efficiently and optimally in all these schemes.

The transformation we discussed in section 2.6.1 was introduced in [55] to reduce the computation overhead sacrificing some transmission and key-storage capacity. A similar technique was already discussed in [6, 50], and employed to improve the computation overhead. The generic transformation described in [55] increases the transmission overhead by a factor of k. To take the transmission back from $2kr$ to $2r$, the work [56] presented a technique that works only for the schemes based on key-chains [6, 50, 122]. This technique will result in, not suprisingly, the basic set system depicted on the left of the Figure 2.16.

In Section 2.6.2 we presented a new generic transformation that gives rise to new schemes with different efficiency parameters. We presented an instantiation of the transformation, which ignoring $\log\log n$ factors, is a scheme that simultaneously achieves communication overhead proportional to r (the number of revoked users), receiver storage $\log n$, and receiver computation proportional to $\log n$. The same efficiency parameters are exhibited by the general Layered Subset Difference (LSD) method (see ([53])), and it turns out that this instantiation is an isomorphic representation of the general LSD method (in particular when the LSD is used for the optimal depth of layering). It worth pointing out that the description of the LSD is different from the recursive application of **XTrans** presented in this chapter and it requires some effort to observe the equivalence. We leave it as an exercise to establish formally this equivalence.

Subset cover schemes that have been discussed in this chapter are designed for *stateless receivers*. A stateless receiver need not maintain state from one transmission to the next. It can go arbitrarily offline without losing its reception capability in the long run. It is also independent of changes within the receiver population, in particular, the decryption capability of receiver keys do not change if other receivers join or leave the system.

While stateless receivers are more easy to administer in a system deployment they affect the revocation capability as well as the efficiency of the system. Suppose that the revocation list of a system employing broadcast encryption scheme increases as time passes. The state-of-the-art suggests that this would yield an increase in the transmission overhead of at least linear in the number of revocations. Such increase can be unbearable for many application scenarios. For instance, systems that employ smart-card like devices can only handle limited computation and storage. Refreshing keys in a fully stateful manner has been discussed in the context of key-management protocols such as the Logical Key Hierarchy, cf. [25, 26, 107, 121, 125]. A hybrid approach is also possible, where some degree of statefulness is required and the system introduces phases or periods over which the receiver has to maintain a state, cf. [7, 37, 45, 74, 82, 90]. The notion of *long-lived broadcast encryption* introduced by Garay et al. [47] where the revoked or compromised keys of the receivers are discarded. A long-lived scheme will minimize the cost of rekeying as well as maintaining the security for enabled users even as the compromised keys are discarded.

A natural extension to the broadcast encryption setting is to allow multiple content providers to broadcast to the same receiver population. An incentive for such an extension is that it accommodates dynamically changing content providers. As a result of using the same broadcast channel for all different providers, the underlying broadcasting infrastructure will be unified and simplified, with decreased storage and equipment costs for the receiver side. However, a shared broadcast channel raises the problem of handling the private information shared between the content providers and the broadcast center as it might be the case that the broadcast channel is entirely separated from the subscription service where the receivers acquire their secret keys.

A trivial solution to that problem is to distribute the same broadcasting key to all content providers. This risks a violation of the content protection if any of the providers is corrupted. On the other hand, distributing different broadcasting keys and their corresponding user keys will isolate providers but not scale properly as the number of content providers grows. This discussion recalls the similar deficiencies of encryption in the symmetric key setting, and leads to investigating broadcast encryption in the public key setting where the content providers all share a publicly known encryption key while the receivers are given the secret decryption keys uniquely assigned to them subject to their subscription levels. A number of early works in designing broadcast encryption schemes in the public key setting include [35, 36, 37, 90].

While these works have transmission overhead dependent on the size of the revocation list, this barrier was overcome by Boneh and et al. in [18]. They presented a construction that achieves constant size transmission overhead. As the public key size is linear in number of receivers, this scheme is not particularly practical. A number of constructions (cf. [19, 33, 34, 103]) gave trade-offs between the efficiency parameters indicated above. We would like to note that all these constructions are based on pairings over elliptic curves, a technique which has received a lot interest in the design of cryptographic schemes. See [46] for an overview of pairing-based cryptography. The latter constructions also support identity-based encryption (see [106, 17]) which suggests the fact that the public key associated to the scrambling of a transmission can be any string.