Graph-Decomposition-Based Frameworks for Subset-Cover Broadcast Encryption and Efficient Instantiations

Nuttapong Attrapadung and Hideki Imai

Imai Laboratory, Institute of Industrial Science, University of Tokyo
4-6-1 Komaba, Meguro-ku, Tokyo 153-8505, Japan.
nuts@imailab.iis.u-tokyo.ac.jp,imai@iis.u-tokyo.ac.jp

Abstract. We present generic frameworks for constructing efficient broadcast encryption schemes in the subset-cover paradigm, introduced by Naor et.al., based on various key derivation techniques. Our frameworks characterize any instantiation completely to its underlying *graph decompositions*, which are purely combinatorial in nature. This abstracts away the security of each instantiated scheme to be guaranteed by the generic one of the frameworks; thus, gives flexibilities in designing schemes. Behind these are new techniques based on (trapdoor) RSA accumulators utilized to obtain practical performances.
We then give some efficient instantiations from the frameworks. Our first construction improves the currently best schemes, including the one proposed by Goodrich et.al., without any further assumptions (only pseudorandom generators are used) by some factors. The second instantiation, which is the most efficient, is instantiated based on RSA and directly improves the first scheme. Its ciphertext length is of order $O(r)$, the key size is $O(1)$, and its computational cost is $O(n^{1/k} \log^2 n)$ for any (arbitrary large) constant k; where r and n are the number of revoked users and all users respectively. To the best of our knowledge, this is the first explicit collusion-secure scheme in the literature that achieves both ciphertext size and key size independent of n simultaneously while keeping all other costs efficient, in particular, sub-linear in n. The third scheme improves Gentry and Ramzan's scheme, which itself is more efficient than the above schemes in the aspect of asymptotic computational cost.

Keywords: Broadcast Encryption, Revocation Scheme, Subset-cover, Optimal Key Storage

1 Introduction

Broadcast encryption (BE) involves 1 broadcaster and n receivers. Each receiver is given a unique private key. The broadcaster is given a private broadcaster key. The broadcaster wishes to broadcast messages to a designated set $P \subseteq N = \{1, ..., n\}$ of receivers. Any receivers in P should be able to decrypt the broadcast message using only its private key while a coalition $F \subseteq N \setminus P$ (revoked users) should not be able to do so. Such a scheme is motivated largely by pay-TV

systems, the distribution of copyrighted materials such as CD/DVD. Broadcast encryption schemes were first formalized by Fiat and Naor [13]. Since then, many variants of the basic problem were proposed. The arguably most challenging variant is the one which considers the case where P can be an arbitrary subset in N while the collusion is considered the full one, $N \setminus P$, and also that the private key stored by each user is fixed from the initialization time (stateless receiver). The main goal is to construct efficient schemes that satisfy the above variant and require only small size of both the header of broadcast and the private key as a function of n or $r := n - |P|$. The *header* is the encapsulation of session key that is used to encrypt data.

An efficient solution which is considered a ground work to many consequences is the Complete (binary) Subtree scheme (CS) by Naor et al. [18]. Schemes which were considered the current state of the art (before two very recent works, see below) are: (i) Pseudo-random sequences generator (PRSG) based schemes such as the Subset Difference scheme (SD) [18], its refinement–the Layered SD scheme (LSD) [14], and their somewhat generalizations in [4]. (ii) RSA accumulator based schemes such as Asano's scheme [2], and its optimal generalizations in [3, 11]. See Table 1 for the efficiency comparison. No scheme above could achieve simultaneous small header size independent of n, small key size of order $O(\log n)$, while keeping computational cost and all other costs grow only sub-linear in n.

More recently, Goodrich et al. [12] and Wang et al. [20] independently propose more efficient schemes that break the above barrier. In particular, they achieve simultaneously header size of order $O(r)$ and key size of $O(\log n)$, and computational cost of $O(n^{1/k})$ for arbitrary constant k. (In fact, in [20] only the case when $k = 1, 2$ is considered).

In this paper, we propose generic frameworks for constructing broadcast encryption and give some efficient instantiations. One of our instantiations (Instantiation 2 in Table 1) achieves not only small header size as of order $O(r)$ but also small key size as $O(1)$ with no extra non-secret storage, while keeping computational cost $O(n^{1/k} \log^2 n)$ which grows only sub-linear in n. Thus this is the first scheme that achieves header and private key size independent of n while keeping computational cost sub-linear in n, with no extra non-secret storage. The contributions in more detail are described below.

1.1 Our Contributions

In the general subset-cover paradigm of [18], which includes almost all of the above schemes, it has been *implicitly* understood that one can separate the design of such a scheme into two seemingly orthogonal problems namely: designing combinatorial set system which enables subset covering (this step determines the header size), and defining computational key derivation (this step determines the private key size and computational cost). This is first explicitly characterized by Gentry-Ramzan [11] for the case of Akl-Taylor's RSA based key derivation [1].

Framework. In this paper, we characterize the two orthogonal components in general. We then explicitly present three generic sub-frameworks for computational key derivation component (*generic* as arbitrary set systems are ap-

Table 1. Comparison among previous schemes and our instantiations. (k is an arbitrary parameter, a is an arbitrary constant)

	Header size		Priv. key size	Comp. cost (bit complexity)	
	Complexity	$\leq$		Prime-gen	Others
CS [18]	$O(r\log(\frac{n}{r}))$		$\log n+1$	-	$O(\log\log n)$
PRSG or OWF -based ↓					
SD [18]	$O(r)$	$2r-1$	$O(\log^2 n)$	-	$O(\log n)$
LSD [14]	$O(r)$	$2kr-k$	$O(\log^{1+1/k} n)$	-	$O(\log n)$
GST04 [12]	$O(r)$	$4kr$	$2\log n$	-	$O(n^{1/k})$
WNR04 [20]	$O(r)$	$4r$	$2\log n$	-	$O(n^{1/2})$
Instantiation 1	$O(r)$	$2kr$	$\leq \log n+1$	-	$O(n^{1/k})$
RSA Accumulator -based ↓					
Asano [2]	$O(r\log_a(\frac{n}{r})+r)$		1	$O(2^a\log_a^5 n)$	$O(2^a\log_a^2 n)$
GR04 [11]	$O(r\log_a(\frac{n}{r})+r)$		1	$O(a\log_a^5 n)$	$O(a\log_a^2 n)$
Instantiation 3	$O(r\log_a(\frac{n}{r})+r)$		1	$O(1)$	$O(a\log n)$
$(\mathsf{SD})^{\mathsf{acc}}$	$O(r)$	$2r-1$	1	$O(n\log^4 n)$	$O(n)$
Instantiation 2	$O(r)$	$2kr$	1	$O((\log^5 n)/k^5)$	$O((n^{1/k}\log^2 n)/k)$

plicable): PRSG based technique (re-formalizing from [4] so as to be consistent with presentations here), non-trapdoor- and trapdoor- RSA Accumulator based techniques. The non-trapdoor RSA based one is a new optimal generalization of Akl-Taylor's technique and is further improved by the trapdoor RSA based one.

The main issue is that we characterize three sub-frameworks so that such instantiations in these frameworks and their resulting efficiencies will depend solely on properties related to *graph decompositions* of the set systems being instantiated; while in the same time the security will be guaranteed *automatically* from the general frameworks. The PRSG based framework will be based on *tree decomposition*, and the two RSA based frameworks will be based on *chain decomposition*; both are purely combinatorial. Therefore the whole paradigm abstracts away the computational security issues and reduces the problem to only pure combinatorics. Moreover it allows modularity in designing a scheme: it is a matter of finding a set system which yields a good header size in the first step, and then finding a graph decomposition of that set system that yields a good private key size and computational cost.

As for the generic efficiency characterization, both RSA based frameworks achieve key size of $O(1)$ for all instances. One generic property of the trapdoor based framework that makes it superior to the non-trapdoor based one is that when restricting to the same asymptotic resources and instantiating the same set system (or to be more precise, its hierarchical version and itself respectively), if the non-trapdoor based one allows n users in the scheme, then the trapdoor based one will allow n^k users for any (arbitrary large) constant k. Indeed, the costs due to prime generation are exactly the same (not only asymptotically).

Efficient Instantiations. For the combinatorial set system component, all of our schemes are based on new set systems we call Subset Incremental chain (SIC) and Layered-SIC (LSIC) which are designed so to achieve small header size as being $O(r)$ while intrinsically have graph decompositions with good properties.

For the computational key derivation component, we instantiate the LSIC set system by presenting their graph decompositions, resulting in various concrete schemes upon each sub-framework as follows. We use the notation $(\mathsf{X})^{\mathsf{y}}$ to denote an instantiation of the set system X using the y-based framework. Denote $\mathsf{LSIC}[k]$ as LSIC with parameter k. Note that $\mathsf{LSIC}[1] = \mathsf{SIC}$.

Instantiation 1 : $(\mathsf{LSIC}[k])^{\mathsf{prsg}}$. This scheme directly improves the scheme of [12, 20] (and it is fair to compare with since the same assumption, PRSG, or equivalently one-way function, was used). In particular it can reduce some overheads, albeit only within constant terms in the worst case: the worst-case key sizes are half of those in [12, 20]. Indeed the key size in our scheme is non-uniform among users; some users are even required to store only constant-size keys (cf. Theorem 4, 6, and Eq.(4)). Our scheme also reduces the computational cost from [12], but only in the average case (the worst-case costs are asymptotically the same).

Instantiation 2 : $(\mathsf{LSIC}[k])^{\mathsf{acc}}, (\mathsf{LSIC}[k])^{\mathsf{tacc}}$. Note that (t)acc is for (trapdoor) accumulator. The performance of this scheme is as mentioned previously. It is the first scheme that achieves header and private key size independent of n while keeping computational cost sub-linear in n, with no extra non-secret storage. The number of primes used per user is optimal as being $O(\log n)$ for $(\mathsf{LSIC}[\mathsf{k}])^{\mathsf{acc}}$ and further reduced to $O((\log n)/k)$ for $(\mathsf{LSIC}[\mathsf{k}])^{\mathsf{tacc}}$ (so that the on-the-fly prime generation cost is $O((\log^5 n)/k^5)$). Had one used the non-optimal Akl-Taylor's framework as put forth to the context of BE by [2, 3, 11], it would be $O(n^{1/k} \log n)$ which is super-logarithmic (and the prime generation cost would be $O(n^{1/k} \log^5 n)$).

Instantiation 3 : $(\mathsf{LSIC}[\log_a n])^{\mathsf{tacc}}$. This scheme improves Gentry and Ramzan's scheme [11], which itself is more efficient than the above schemes in the aspect of asymptotic computational cost. Our scheme reduces *poly-logarithmic* cost due to prime generation, which was the dominant cost, to only a *constant* one without affecting the other parameters. Among the *constant-key-size* schemes with header size $O(r \log_a(n/r) + r)$ and no extra non-secret storage, this is the first one in the literature that achieves $O(\log n)$ overall computational cost. (And in fact, ours uses only a constant number of primes). The previous improvement for this class of schemes was done by [11] to improve [2] but only in the constant term involving a. (See Table 1).

1.2 Other Related Works

Very recently, Boneh et.al. [7] propose a *public-key* broadcast encryption scheme which achieves size $O(1)$ for both header and private key. However, the size of the public key to be used by an encrypter, which is also the non-secret storage needed for the decrypter, is $O(n)$. Moreover, the computational cost is $O(n - r)$ (albeit with small coefficient). The second scheme in [7] reduces the non-secret storage size to $O(\sqrt{n})$ but with the price of the increased header size as $O(\sqrt{n})$, and not independent of n anymore. Boneh and Silverberg [6] show that n-linear maps can be used to construct an optimal public-key scheme with

constant private key, public key, and header size. However, there are currently no known constructions for such a map for $n > 2$. Most recently, Jho et.al. [15] propose some efficient schemes with small header size when r is not too small. However, their schemes do not enjoy practical *asymptotic* performances as either the header size is $c_1 r + c_2 n = O(n)$ (for some constant c_1, c_2) or the key size is $\binom{n-1}{k} = O(n^k)$ (where $k \geq 2$) for their best two schemes.

2 Framework and Some Preliminaries

2.1 Framework

We refer to [18] for the definitions and the security notions for private-key broadcast encryption. Now we recap the subset-cover framework [18] separately into two components as follows.

Combinatorial Set System Component We first redefine a set system which is useful for such a scheme in this framework called complement-cover set system. Such a set system is a family of subsets of a universe with the property that every subset of the universe can be efficiently partitioned to a union of some collection of subsets in the family.

Definition 1 (Complement-Cover Set System) *For a map* $\mathsf{c} : \mathbb{Z}_{>0}^2 \to \mathbb{Z}_{>0}$, *a set system* $\mathcal{S} = \{S_1, ..., S_m\}$ *over a base set* $N = \{1, ..., n\}$ *is* c-complement-cover *if there is a polynomial-time algorithm such that upon input any subset* $R \subset N$, *outputs* $\{S_{i_1}, ..., S_{i_t}\}$ *for some* $1 \leq i_1, \ldots, i_t \leq m$ *such that* $N \setminus R = \bigcup_{j=1}^{t} S_{i_j}$ *and that* $t \leq \mathsf{c}(n, |R|)$. □

As usual n, r is the number of all users and revoked users respectively. Such a $\mathsf{c}(n, r)$-complement-cover set system yields a broadcast encryption scheme in the subset-cover framework with the header size $\mathsf{c}(n, r)$. The scheme is as follows. The broadcaster defines a subset key for each subset in the family. Each user stores a set of keys in such a way that he can derive all the keys of subsets (in the family) that he is a member. (Thus, the easiest way to do is to store them all. However to reduce the storage of keys, it would be better to store only some and derive the others from those stored keys on the fly. Such derivation patterns are predefined by the broadcaster.) To revoke the set R of users, the broadcaster just let a header to be a session key encrypted with each key of subsets in the partition of $N \setminus R$. Thus the header size is $\mathsf{c}(n, r)$. We often denote $\mathsf{c}_{\mathsf{X}}(n, r)$ for $\mathsf{c}(n, r)$ of the set system $\mathcal{S}_{\mathsf{X}}$, where X is the name of that set system.

Computational Key Derivation Component We formalize the specification on key derivations in the context of access control scheme as the following. Denote by $\mathsf{k}(S)$ the subset key for $S \in \mathcal{S}$ and $\mathsf{p}(u)$ the private key of $u \in N$. Informally, the security of such a scheme requires that with $\mathsf{p}(u)$, one can derive $\mathsf{k}(S)$ if and only if $u \in S$; moreover, the collusion $N \setminus S$ cannot derive it.

Definition 2 (Access Control Scheme, AC) *An Access Control Scheme* AC *for a set system* $\mathcal{S}$ *over a base set* N *is a 2-tuple of polynomial-time algorithms* (Keygen, Derive), *where:*

$\mathsf{Keygen}(1^\lambda)$: *Takes as input a security parameter* 1^λ*. It returns all* $\mathsf{k}(S_i)$*'s, all* $\mathsf{p}(u)$*'s, and public parameter* pub*.*

$\mathsf{Derive}(\langle u, \mathsf{p}(u)\rangle, S_i, \mathsf{pub})$: *Takes as input* $u \in N$*, the key* $\mathsf{p}(u)$*,* $S_i \in \mathcal{S}$*, and* pub*. It returns* $\mathsf{k}(S_i)$ *if* $u \in S_i$*, or special symbol* $\perp$ *otherwise.* □

Naor et al. [18] proved that BE in the subset-cover paradigm whose the access control component is secure in the sense of Key-Indistinguishability (KIND) is secure in the standard notion, namely IND-CCA1. Dodis and Katz [10] use the technique involving multiple encryption to obtain a generic scheme which is IND-CCA2-secure. Key-Intractability (KINT) can be defined analogously. These definitions are captured in the full version of this paper due to limited space here. Also note that there is a simple conversion from KINT-secure scheme to KIND-secure one. Thus KIND or KINT is sufficient for the security of the scheme.

Denote $(\mathsf{X})^\mathsf{y}$ to be the access control scheme for set system $\mathcal{S}_\mathsf{X}$ that is constructed via AC framework y. Denote $\mathsf{KeySize}_{(\mathsf{X})^\mathsf{y}}(u)$ to be the number of keys of u (i.e., $|\mathsf{p}(u)|$, when $\mathsf{p}(u)$ is treated as a set) and $\mathsf{CompCost}_{(\mathsf{X})^\mathsf{y}}$ to be the worst-case computational cost for Derive. We also refer $(\mathsf{X})^\mathsf{y}$ as a BE scheme via the complement-cover set system $\mathcal{S}_\mathsf{X}$. For any y, $\mathsf{HeaderSize}_{(\mathsf{X})^\mathsf{y}}(n, r) = \mathsf{c}_\mathsf{X}(n, r)$.

2.2 Some Terminology

Viewing Set system as Poset. A set system is partially ordered by the inclusion relation ($\subset$). Interpreting a set system as a partially ordered set (poset) is useful when defining key derivations in AC. Intuitively, Derive algorithm implies that whenever $S_i \subset S_j$, anyone who can access $\mathsf{k}(S_i)$ is allowed to access $\mathsf{k}(S_j)$.

Terminology for Posets, Graphs. The terminology for posets and graphs used in this paper is quite standard one (cf.[9]) (with some exceptions, see below). Here we review some. A graph is a pair $G = (V, E)$ of sets satisfying $E \subseteq \binom{V}{2}$. V is the set of vertices (or nodes), usually denoted $V(G)$, E is the set of edges, usually denoted $E(G)$. Often, we abuse notation $v \in G$ to mean $v \in V(G)$. A tree is a connected acyclic graph. We often denote $x = \mathsf{parent}_T(y)$ if x is the parent of y in tree T. A directed graph is a pair $G = (V, E)$ of sets satisfying $E \subseteq V \times V$, i.e., an edge is an ordered pair. A directed acyclic graph (DAG) is a directed graph with no directed cycle in it. A notation of chain $x \to y \to z$ means a directed graph which $E = \{x, y, z\}, V = \{(x, y), (y, z)\}$ and is generalized naturally.

An inclusion poset $\mathcal{S}$ can be represented by a DAG G by setting $V = \mathcal{S}$, $E = \{(S, S') : S \subset S';\ S, S' \in \mathcal{S}\}$. This is called the maximal representation, denoted $\mathsf{DAG}_{\mathsf{max}}(\mathcal{S})$. The minimal representation, denoted $\mathsf{DAG}_{\mathsf{min}}(\mathcal{S})$, is the one with $E = \{(S, S') : S \subset_c S';\ S, S' \in \mathcal{S}\}$ where we say $S \subset_c S'$ iff there is no $S'' \in \mathcal{S}$ such that $S \subset S'' \subset S'$.

In our context[1], a graph decomposition (often denoted $\mathcal{G}$) of a poset $\mathcal{S}$ is a family of connected subgraphs whose sets of nodes partition the set of all nodes in the $\mathsf{DAG}_{\mathsf{max}}(\mathcal{S})$. (Thus we sometimes say $\mathcal{G}$ is a graph decomposition of $\mathsf{DAG}_{\mathsf{max}}(\mathcal{S})$). When each subgraph is a tree whose edges are directed away from

[1] Our notions for tree and chain decompositions are *not* standard ones (cf.[9]). Instead the notions introduced here might be named as *tree cover* and *path cover*, resp.

the root, we call it a tree decomposition (often denoted $\mathcal{T}$). When each graph is a directed chain whose edges are directed in the same direction, we call it a chain decomposition (often denoted $\mathcal{C}$). An induced graph decomposition is one in which each subgraph is an induced subgraph. Fig.1 shows graph decompositions of the set system for toy example 1, $\mathcal{S}_{\mathsf{toy1}} = \{\{1\},\{2\},\{3\},\{4\},\{1,2\},\{2,3\},\{2,4\},\{3,4\},\{1,2,3\}\}$. From now we abuse some notations, often in figures, e.g., writing 12 or $1,2$ instead of $\{1,2\}$ if it causes no confusion. Note that every chain decomposition is a tree decomposition.

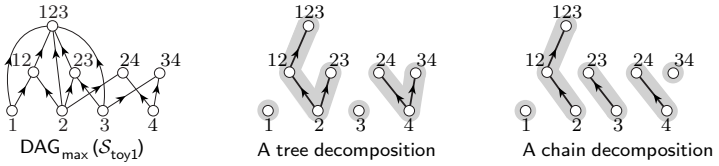

Fig. 1. Toy example 1 and its graph decompositions

We will fix BT to be the complete binary tree of n leaves labeled $1,...,n$ from left to right. The level of node in BT is the distance from root to it. For a fixed node, its left (resp., right) nodes are those nodes with the same level and appear on the left (resp., right). BT will be used only to help defining set systems and should not be confused with the graph representations of posets of set systems.

3 New Set Systems

3.1 Subset Incremental Chain (SIC) Set System

The SIC Set System. For $i,j \in N = \{1,...,n\}$ and $i < j$, denote

$$i \rightharpoonup j := \{\{i\},\{i,i+1\},\ldots,\{i,\ldots,j\}\},$$
$$i \leftharpoonup j := \{\{j\},\{j,j-1\},\ldots,\{j,\ldots,i\}\},$$

and $(i \rightharpoonup i) = (i \leftharpoonup i) := \{\{i\}\}$. Consider the binary tree BT. For a node v in BT, let l_v (resp., r_v) be the leftmost (resp., rightmost) leaf under v. We define the set system SIC (of n users) by letting

$$\mathcal{S}_{\mathsf{SIC}} = \bigcup_{v\in\mathsf{BT_L}} (l_v+1 \leftharpoonup r_v) \cup \bigcup_{v\in\mathsf{BT_R}} (l_v \rightharpoonup r_v-1) \cup (1 \rightharpoonup n) \cup (2 \leftharpoonup n), \qquad (1)$$

where $\mathsf{BT_L}$ (resp., $\mathsf{BT_R}$) are the set of internal nodes which are left (resp., right) children. An informal visual view of $\mathcal{S}_{\mathsf{SIC}}$ is shown in Fig.2, where the union of all the collections written there is the only important information.

Theorem 1 *$\mathcal{S}_{\mathsf{SIC}}$ is $(2r)$-complement-cover set system.*

Proof. We call a set of the form $\{i,i+1,\ldots,j\}$ for some $i \le j$ a consecutive set. We first claim that any consecutive set, say $A = \{i,\ldots,j\}$, can be partitioned to no more than 2 sets in $\mathcal{S}_{\mathsf{SIC}}$; then prove it as follows. Let a be the least common ancestor node of the leaves i and j in BT, denoted $\mathsf{lca}(i,j) = a$. Let s be the least ancestor of a which is in $\mathsf{BT_L}$ if $a \in \mathsf{BT_R}$ and which is in $\mathsf{BT_R}$ if $a \in \mathsf{BT_L}$. Let x,y be the left and right children of a. First if $i = 1$ then $A \in (1 \rightharpoonup n) \subseteq \mathcal{S}_{\mathsf{SIC}}$; else if $j = n$ then $A \in (2 \leftharpoonup n) \subseteq \mathcal{S}_{\mathsf{SIC}}$ (since $2 \le i$). Now assume $i \ne 1$, $j \ne n$. We list all possible cases of (i,j) as follows. Let $*$ be an unspecified value.

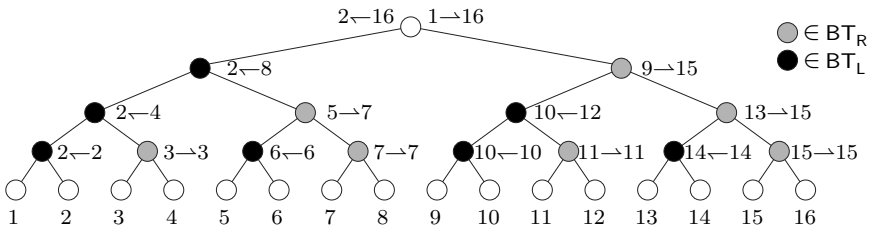

Fig. 2. Set system SIC defined by the union of all the collections written at each node

1. If $(i = l_a; j = *; a \in \mathsf{BT_L})$ then $A \in (l_s \rightharpoonup r_s - 1) \subseteq \mathcal{S}_{\mathsf{SIC}}$ (since $i = l_s$; $j < r_s - 1$; and $s \in \mathsf{BT_R}$),
2. If $(i = *; j = r_a; a \in \mathsf{BT_R})$ then $A \in (l_s + 1 \leftharpoonup r_s) \subseteq \mathcal{S}_{\mathsf{SIC}}$ (since $j = r_s$; $l_s + 1 < i$; and $s \in \mathsf{BT_L}$),
3. If $(i = l_a; j \neq r_a; a \in \mathsf{BT_R})$ then $A \in (l_a \rightharpoonup r_a - 1) \subseteq \mathcal{S}_{\mathsf{SIC}}$ (since $j \leq r_a - 1$),
4. If $(i \neq l_a; j = r_a; a \in \mathsf{BT_L})$ then $A \in (l_a + 1 \leftharpoonup r_a) \subseteq \mathcal{S}_{\mathsf{SIC}}$ (since $l_a + 1 \leq i$),
5. If $(i \neq l_a; j \neq r_a; a \in *)$ then $A = P \cup Q$; $P = \{i, \ldots, r_x\}$, $Q = \{l_y, \ldots, j\}$, and we have $P, Q \in \mathcal{S}_{\mathsf{SIC}}$ (since
   - $\mathsf{lca}(i, r_x) = x$, thus (i, r_x) will fall to the case 2 or 4 and $P \in \mathcal{S}_{\mathsf{SIC}}$;
   - $\mathsf{lca}(l_y, j) = y$, thus (l_y, j) will fall to the case 1 or 3 thus $Q \in \mathcal{S}_{\mathsf{SIC}}$).

These proved the claim. Now we are back to the proof, it is obvious that $N \setminus R$ can be partitioned to no more than r consecutive sets if 1 or $n \in R$; or to no more than $r+1$ such sets otherwise. In the former case, the partition size to sets in $\mathcal{S}_{\mathsf{SIC}}$ is $\leq 2r$; while in the latter case (where $\{1, ..., s\}$ and $\{t, ..., n\}$ for some s, t are included in the partition), it is $\leq 1(1) + 2(r-1) + 1(1) = 2r$. □

Intuitively, SIC has graph decompositions with good properties since each collection in the union of Eq.(1) forms a chain of subset. This will become clearer in the next section. The set system LSIC below generalizes SIC.

3.2 Layered SIC (LSIC) Set Systems

The LSIC[k] Set System. We view BT consisting of subtrees (also binary and complete) of $n^{1/k}$ leaves so that there are exactly k layers of such subtrees, where $k | \log n$. We will call such subtree an "atomic" subtree (to distinguish from other kinds of subtrees in BT). Informally, each atomic subtree contributes sets to $\mathcal{S}_{\mathsf{LSIC}}$ as in the SIC set system for that subtree, albeit each leaf in the subtree represents all the leaves under it in BT. More formally, for node z in BT, let $A_z := \{l_z, l_z + 1, ..., r_z\}$ (i.e., all the leaves under z). Let us consider the leaves u, v in an atomic subtree where v is some node on the right of u. We denote $u^{(+1)}, u^{(+2)}$ (and so on) be the next one, two (and so on) *right* leaves to u in that atomic subtree. Denote $u^{(-1)}, u^{(-2)}$ analogously. Denote

$$u \rightharpoonup v := \{A_u, A_u \cup A_{u^{(+1)}}, \ldots, A_u \cup \cdots \cup A_v\},$$
$$u \leftharpoonup v := \{A_v, A_v \cup A_{v^{(-1)}}, \ldots, A_v \cup \cdots \cup A_u\}.$$

Let l'_w, r'_w be the leftmost and rightmost leaves under w *in the atomic subtree* and not w itself; for example, $l'_{\mathsf{root}} = a, r'_{\mathsf{root}} = d$ and $l'_a = 1, r'_a = 4$ in Fig.3. Let

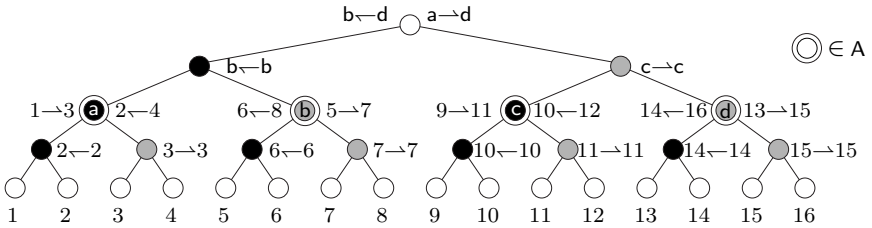

Fig. 3. Set system $\mathsf{LSIC}[k]$, $k = 2$, as the union of all collections written at each node

A be the set of all nodes which are the roots of atomic subtrees but excluding the root of BT. We define $\mathsf{LSIC}[k]$ analogously to Eq.(1) by letting

$$\mathcal{S}_{\mathsf{LSIC}[k]} = \bigcup_{v \in \mathsf{BT_L} \cup \mathsf{A}} (l'^{(+1)}_v \leftharpoonup r'_v) \cup \bigcup_{v \in \mathsf{BT_R} \cup \mathsf{A}} (l'_v \rightharpoonup r'^{(-1)}_v) \cup (l'_{\mathsf{root}} \rightharpoonup r'_{\mathsf{root}}) \cup (l'^{(+1)}_{\mathsf{root}} \leftharpoonup r'_{\mathsf{root}}). \quad (2)$$

Intuitively, each $v \in \mathsf{A}$ has two collections $(l'^{(+1)}_v \leftharpoonup r'_v), (l'_v \rightharpoonup r'^{(-1)}_v)$ attached since it is the root of an atomic subtree, which SIC applies (cf. Eq.(1) and Fig.2).

Theorem 2 *$\mathcal{S}_{\mathsf{LSIC}[k]}$ is $(2kr)$-complement-cover set system for a constant k ; and $\mathcal{S}_{\mathsf{LSIC}[\log_a n]}$ is $O(r \log_a(n/r) + r)$-complement-cover set system for a constant a.*

Note that when $k = \log_a n$, from the former claim we already have that $\mathcal{S}_{\mathsf{LSIC}[\log_a n]}$ is $(2r \log_a n)$-complement-cover, but the claim above gives a sharper bound.

Proof. First we will prove that $\mathcal{S}_{\mathsf{LSIC}[k]}$ is $(2kr)$-complement-cover. Let ST_R denote the Steiner tree of a set of leaves $R \subseteq N$, i.e., the subtree of BT that consists of all paths from the root to each leaf in R. We call a node v special if $v \in \mathsf{A}$. We "color" a node if it is special but is not in ST_R and all of its special ancestors are in ST_R. Denote C the set of all color nodes. Hence $N \setminus R = \bigcup_{v \in \mathsf{C}} A_v = \bigcup_{j=1}^{k} \bigcup_{v \in L_j \cap \mathsf{C}} A_v$ where we denote L_j to be the set of all special nodes in the j-th special layer away from root (i.e., at distance $j(\log n)/k$ from the root). It suffices to prove that for each special layer j, the set $Y_j := \bigcup_{v \in L_j \cap \mathsf{C}} A_v$ can be partitioned to at most $2r$ sets in the family $\mathcal{S}_{\mathsf{LSIC}}$. Denote x_i to be the number of uncolored special nodes in the i-th atomic subtrees from left to right in this j-th layer. From Theorem 1, it is easy to deduce that Y_j can be partitioned to at most $2(x_1 + x_2 + \cdots + x_p)$ sets in $\mathcal{S}_{\mathsf{LSIC}}$, where p is the last atomic subtree in this layer (in fact, $p = n^{(j-i)/k}$). But we have $x_1 + \cdots + x_p \leq r$ since the Steiner tree of r leaves passes through all these uncolored special nodes. This proves the claim.

Next we will prove that $\mathcal{S}_{\mathsf{LSIC}[\log_a n]}$ is $O(r \log_a(n/r) + r)$-complement-cover. We first give the definition of Stratified Subset-Difference set system with each atomic subtree of a leaves (SSD_a): $\mathcal{S}_{\mathsf{SSD}_a} = \{A_u \setminus A_v : u \text{ is an ancestor of } v \text{ in the same atomic subtree}\}$. It is known [11] that $\mathcal{S}_{\mathsf{SSD}_a}$ is $(O(r \log_a(n/r)+r))$-complement-cover. Using a similar approach as when proving Theorem 1, it is not hard to see that each $A_u \setminus A_v$ can be partitioned to at most 2 sets in $\mathcal{S}_{\mathsf{LSIC}[\log_a n]}$. (The proof is omitted here due to space). Combining these we have that $\mathsf{LSIC}[\log_a n]$ has $\mathsf{c}_{\mathsf{LSIC}[\log_a n]}(n, r) = 2\mathsf{c}_{\mathsf{SSD}_a}(n, r) = O(r \log_a(n/r) + r)$. □

4 Key Derivation based on PRSG

4.1 Reformalize the PRSG based Framework of [4]

Framework Idea (review). In this framework, we use pseudo-random sequence generators to derive keys from one subset to another. The correctness of access control schemes allows this to be done only if the first set is included in the latter (e.g.,$\{1\} \subset \{1,2\}$). Thus such derivations can be defined in correspondence with directed edges in a graph decomposition of $\mathsf{DAG}_{\mathsf{max}}(\mathcal{S})$, in which all the inclusion relations in $\mathcal{S}$ are included. One exception is that there should be no node with $\mathsf{indegree} > 1$ in any graph in the decomposition since it would imply a collision of PRSG, which should be computable by neither broadcasters nor adversaries. Therefore, all the valid decompositions are *tree decompositions*, of which the class includes all graph decompositions of the poset that allow $\mathsf{indegree} \leq 1$ for all nodes. Each user then stores keys for subsets which he is in and are closest to the root of that tree. For the toy example 1 in Fig.1, our paradigm with the tree decomposition in the figure namely $\mathcal{T}_{\mathsf{toy1}}$ allows the user 2 to store only the keys at $2, 24$.

Note that in order to be provably secure in the KIND sense, it is mandatory to make an adaptation so that keys are not derived from another key *directly*. Instead, one should use intermediate keys denoted $\mathsf{t}(S)$ for $S \in \mathcal{S}$; how to use this is explained in the construction. This was neglected in many recent schemes that use similar one-way derivation approaches.

The Construction $(\mathsf{X})^{\mathsf{prsg}}$**.** This is based solely on a tree decomposition, say $\mathcal{T}$, of the poset $\mathcal{S}_{\mathsf{X}}$. The scheme applies to an arbitrary complement-cover set system X.

Keygen : (Subset keys) At a root S of a tree in $\mathcal{T}$, let $\mathsf{t}(S) \leftarrow \{0,1\}^{\lambda}$. For each node S (either root or non-root of a tree in $\mathcal{T}$) whose all children are $S_{i_1}, ..., S_{i_d}$ where d is the outdegree of S, we define the following recurrence relation:

$$\mathsf{t}(S_{i_1}) \| \cdots \| \mathsf{t}(S_{i_d}) \| \mathsf{k}(S) \leftarrow \mathsf{PRSG}_{d+1}(\mathsf{t}(S)), \tag{3}$$

where $|\mathsf{t}(S_{i_1})| = \cdots = |\mathsf{t}(S_{i_d})| = |\mathsf{k}(S)| = \lambda$ bits; $\mathsf{PRSG}_j : \{0,1\}^{\lambda} \rightarrow \{0,1\}^{j\lambda}$.

(User keys) For $u \in N$, we define $\mathsf{p}(u) = \{\mathsf{t}(S) | u \in S; u \notin \mathsf{parent}_G(S), G \in \mathcal{T}\}$.

Derive : Find the tree where S is in and then use Eq.(3) to derive $\mathsf{k}(S)$.

Characterizing Efficiency. Let $\mathsf{RN}_{\mathcal{T}}(u) = |\{S \mid u \in S; u \notin \mathsf{parent}_G(S), G \in \mathcal{T}\}|$ and call it the *reachability number* of u in $\mathcal{T}$ (since it is the minimal number of sufficient nodes such that when traversing from these nodes in the edge direction we meet all $S \in \mathcal{S}$ such that $u \in S$). Let $\mathsf{DD}_{\mathcal{T}}$ = the depth of the deepest trees. We have

$$\mathsf{KeySize}_{(\mathsf{X})^{\mathsf{prsg}}}(u) = \mathsf{RN}_{\mathcal{T}}(u), \quad \mathsf{CompCost}_{(\mathsf{X})^{\mathsf{prsg}}} = \mathsf{DD}_{\mathcal{T}}. \tag{4}$$

Theorem 3 ([4]) $(\mathsf{X})^{\mathsf{prsg}}$ *is secure in the sense of KIND assuming secure* PRSG.

4.2 PRSG based Instantiation for SIC, LSIC

Instantiating SIC. It suffices to define a tree decomposition of $\mathcal{S}_{\mathsf{SIC}}$ and the concrete scheme will follow automatically from the general construction of the framework. We choose the following natural one and prove that it is the optimal decomposition for SIC. For $i \leq j \in N$, define a graph $\mathsf{G}(i \rightharpoonup j)$ as $\{i\} \rightarrow \{i, i+1\} \rightarrow \cdots \rightarrow \{i, ..., j\}$; $\mathsf{G}(i \leftharpoonup j)$ as $\{j\} \rightarrow \{j, j-1\} \rightarrow \cdots \rightarrow \{j, ..., i\}$. Let

$$\mathcal{T}_{\mathsf{SIC}} = \{\mathsf{G}(l_v+1 \leftharpoonup r_v) | v \in \mathsf{BT_L}\} \cup \{\mathsf{G}(l_v \rightharpoonup r_v - 1) | v \in \mathsf{BT_R}\} \cup \{\mathsf{G}(1 \rightharpoonup n), \mathsf{G}(2 \leftharpoonup n)\} \tag{5}$$

Let $\langle x \rangle$ denotes the binary representation of x. We have the following theorem.

Theorem 4 *The tree decomposition $\mathcal{T}_{\mathsf{SIC}}$ yields minimal $\max_{u \in N} \mathsf{RN}_{\mathcal{T}}(u)$, indeed we have*

$$\mathsf{RN}_{\mathcal{T}_{\mathsf{SIC}}}(u) = \begin{cases} \log n + 2 - f(\langle u-1 \rangle) & ; 2 \leq u \leq n \\ 1 & ; u = 1, \end{cases}$$

where $f(y) :=$ the number of the same consecutive least significant bits of y. In particular, $\max_{u \in N} \mathsf{RN}_{\mathcal{T}_{\mathsf{SIC}}}(u) = \log n + 1$. We also have $\mathsf{DD}_{\mathcal{T}_{\mathsf{SIC}}} = n$.

Proof. We define $F_v = l_v + 1 \leftharpoonup r_v$ if $v \in \mathsf{BT_L}$ and $l_v \rightharpoonup r_v - 1$ if $v \in \mathsf{BT_R}$. $\mathcal{T}_{\mathsf{SIC}}$ is really a tree decomposition since $\{F_v : v \in \mathsf{BT_L} \cup \mathsf{BT_R}\} \cup \{(1 \rightharpoonup n), (2 \leftharpoonup n)\}$ can be proved to be a pairwise non-intersecting family (somewhat straightforwardly). Next we prove the formula for $\mathsf{RN}_{\mathcal{T}_{\mathsf{SIC}}}(u)$. For $u \in N \setminus \{1\}$, only possible trees in $\mathcal{T}_{\mathsf{SIC}}$ that u appears are those graphs $\mathsf{G}(F_v)$ for internal nodes v on the path from the leaf u to the root in BT, and $\mathsf{G}(1 \rightharpoonup n), \mathsf{G}(2 \leftharpoonup n)$. Each graph $\mathsf{G}(\cdot)$ that u appears contribute one key for u. Thus $\mathsf{RN}_{\mathcal{T}_{\mathsf{SIC}}}(u)$ is at most $(\log n - 1) + 2$. Let $u, w_1, ..., w_{\log n}$, root be the nodes on that path. Due to symmetry, we assume w.l.o.g. that $w_1, ..., w_{z-1} \in \mathsf{BT_L}$ and $w_z \in \mathsf{BT_R}$. Now it is easy to see that

$$\begin{array}{ll} \text{for } 1 \leq j \leq z-1 & : \mathsf{G}(F_{w_j}) = \mathsf{G}(l_{w_j} + 1 \leftharpoonup r_{w_j}) \text{ does not contain } u (= l_{w_j}); \\ \text{for } j = z & : \mathsf{G}(F_{w_j}) = \mathsf{G}(l_{w_z} \rightharpoonup r_{w_z} - 1) \text{ contains } u (= l_{w_z}); \\ \text{for } z < j \leq \log n & : \mathsf{G}(F_{w_j}) \text{ contains } u \text{ (since } l_{w_j} < u < r_{w_j}), \end{array}$$

and that $z = f(\langle u-1 \rangle)$. Thus $\mathsf{RN}_{\mathcal{T}_{\mathsf{SIC}}}(u) = (\log n - 1) + 2 - (f(\langle u-1 \rangle) - 1)$ as desired. Now we prove that $\mathcal{T}_{\mathsf{SIC}}$ is optimal (obtaining minimal $(\max_{u \in N} \mathsf{RN}_{\mathcal{T}}(u))$ among all $\mathcal{T}$ of SIC). Observe that for all $\mathcal{T}$ of SIC, $\sum_{u \in N} \mathsf{RN}_{\mathcal{T}}(u) = \sum_{S \in \mathcal{S}_{\mathsf{SIC}}} |\{u : u \in S, u \notin \mathsf{parent}_G(S), G \in \mathcal{T}\}| \geq |\mathcal{S}_{\mathsf{SIC}}| = n \log n + 1$. Hence $\max_{u \in N} \mathsf{RN}_{\mathcal{T}}(u) \geq \lceil \frac{n \log n + 1}{n} \rceil = \log n + 1$. Our decomposition matches this bound. □

The number of keys at each user is not uniform as recorded in the corollary below. While sharing some similarities with our scheme, the basic schemes in [12, 20] assign one-way chains in both left and right directions at each node in BT while we use only one direction and exploit some symmetries. This can be an intuition as to why we can reduce key size at least 2 times (and up to $\log n$ in the best case, user 1). Those schemes can be considered as instantiations in our framework, but with storage-redundancies in the sense that the set systems extracted from their schemes are sets with repetition. Moreover, the scheme of [12] can also be shown to be derivation-redundant since its derivation graph as exposed in our framework contains loop edges. (See our full paper).

Corollary 5 *In the scheme* $(\mathsf{SIC})^{\mathsf{prsg}}$*, there are exactly* 2^x *users who store exactly* $x+2$ *keys for* $0 \le x \le (\log n) - 1$ *and exactly 1 user who stores 1 key.*

Instantiating LSIC. Before describing our default tree decomposition of $\mathcal{S}_{\mathsf{LSIC}}$, denoted $\mathcal{T}_{\mathsf{LSIC}[k]}$, we first describe a more straightforward one, denoted $\mathcal{T}'_{\mathsf{LSIC}[k]}$, which is constructed, informally, as the union of all $\mathcal{T}_{\mathsf{SIC}}$ applied to each atomic subtree in BT. More formally, we can define $\mathsf{G}(u \rightharpoonup v)$ for u, v which are leaves in the same atomic subtree, analogously as before, by letting $\mathsf{G}(u \rightharpoonup v) = A_u \to A_u \cup A_{u^{(+1)}} \to \cdots \to (A_u \cup \cdots \cup A_v)$, and analogously for $\mathsf{G}(u \leftharpoonup v)$. Without going into details, we can define $\mathcal{T}'_{\mathsf{LSIC}[k]}$ from Eq.(2) in an analogous way when we defined $\mathcal{T}_{\mathsf{SIC}}$ in Eq.(5) from Eq.(1).

Now $\mathcal{T}_{\mathsf{LSIC}[k]}$ is constructed by an observation that $\mathsf{G}(l'_v \rightharpoonup r'^{(-1)}_v)$ and $\mathsf{G}(v \rightharpoonup *)$ can be combined into one chain (and in particular, one tree) since the maximum element in the former, $A_{l'_v} \cup \cdots \cup A_{r'^{(-1)}_v}$, is included in A_v, the minimum element of the latter. For $v \in \mathsf{BT_R} \cup \{\mathsf{root}\}$, let $w_1, ..., w_m$ be the sequence of nodes in $\mathsf{BT_L} \cap \mathsf{A}$ such that $w_1 = l'_v$; for $1 \le i \le m-1$, $w_{i+1} = l'_{w_i}$; and $l_v = l'_{w_m}$, then define $\bar{\mathsf{G}}(l'_v \rightharpoonup x) := \mathsf{G}(l'_{w_m} \rightharpoonup r'^{(-1)}_{w_m}) \to \cdots \to \mathsf{G}(l'_{w_1} \rightharpoonup r'^{(-1)}_{w_1}) \to \mathsf{G}(l'_v \rightharpoonup x)$ where x is some right node of l'_v. (Here, '$\to$' means to connect the chains). The definition for $\bar{\mathsf{G}}(x \leftharpoonup r'_v)$ for $v \in \mathsf{BT_L} \cup \{\mathsf{root}\}$ can be done analogously. Now we define

$$\mathcal{T}_{\mathsf{LSIC}[k]} = \{\bar{\mathsf{G}}(l'^{(+1)}_v \leftharpoonup r'_v) | v \in \mathsf{BT_L}\} \cup \{\bar{\mathsf{G}}(l'_v \rightharpoonup r'^{(-1)}_v) | v \in \mathsf{BT_R}\} \cup \{\bar{\mathsf{G}}(l'_{\mathsf{root}} \rightharpoonup r'_{\mathsf{root}}), \bar{\mathsf{G}}(l'^{(+1)}_{\mathsf{root}} \rightharpoonup r'_{\mathsf{root}})\}. \quad (6)$$

The abstraction of this decomposition may disguise the simplicity of the scheme; in Fig.4 we thus give an explicit example when $n = 16$ and $k = 2$ (cf. Fig.3).

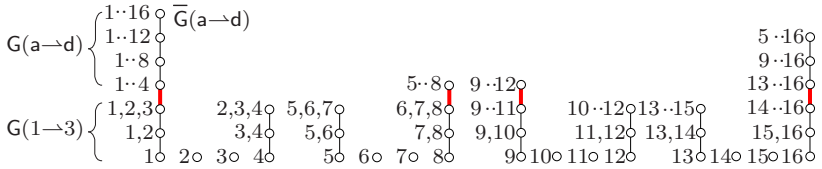

Fig. 4. The tree decomposition $\mathcal{T}_{\mathsf{LSIC}[k]}$ of the set system $\mathsf{LSIC}[k]$ (see Fig.3). A more simple decomposition $\mathcal{T}'_{\mathsf{LSIC}[k]}$ is the one without the thick red edges.

The following theorem and corollary can be proved by an elementary counting argument based on Theorem 4. We omit the proof to the full version of this paper.

Theorem 6 *The tree decomposition* $\mathcal{T}_{\mathsf{LSIC}[k]}$ *yields*

$$\mathsf{RN}_{\mathcal{T}_{\mathsf{LSIC}[k]}}(u) = \log n + 1 + k - g_k(\langle u-1 \rangle)$$

where $g_k(\langle x \rangle) := f(0||\langle x_1 \rangle) + f(b_1||\langle x_2 \rangle) \cdots + f(b_{k-1}||\langle x_k \rangle)$ *where we parse* $\langle x \rangle$*, with padding of* 0*s on the left so to have length* $\log n$ *bits, as* $\langle x_1 \rangle || \cdots || \langle x_k \rangle$ *so that each* $\langle x_i \rangle$ *has length* $(\log n)/k$ *bits;* b_j *is the least significant bit of* $\langle x_j \rangle$*. In particular,* $\max_{u \in N} \mathsf{RN}_{\mathcal{T}_{\mathsf{LSIC}[k]}}(u) = \log n + 1$*. We also have* $\mathsf{DD}_{\mathcal{T}_{\mathsf{LSIC}[k]}} = kn^{1/k}$*.*

As an example, user 4 will store 2 keys: $\mathsf{k}(1234), \mathsf{k}(4)$ (see Fig.4). This can be calculated as $|\mathsf{p}(4)| = 4+1+2-(f(0||00)+f(0||11)) = 2$ (Note $\langle 4-1\rangle = 0011$).

Corollary 7 *In* $(\mathsf{LSIC}[\mathsf{k}])^{\mathsf{prsg}}$*, exactly* $\sum_{j=0}^{x-1}\binom{k}{j}C(x-1,j,(\log n)/k)2^{x-1-j}$ *users store exactly* x *keys for* $2 \le x \le (\log n)+1$ *and exactly 1 user stores 1 key where* $C(a,b,c)$ *is the number of integer compositions (ordered partitions) of* a *into* b *positive integers, each* $\le c$*.*[2]

5 Key Derivation based on Non-Trapdoor RSA

5.1 The New Non-Trapdoor RSA based Framework

Framework Idea. We first briefly review the access control scheme of Akl-Taylor [1]. There, each $S \in \mathcal{S}$ is assigned a publicly known prime. The key of S is defined as $\mathsf{k}(S) = s^{\prod_{T:S\not\rightarrow T} p_T}$ modulo an RSA modulus, where s is a secret; and $S \not\rightarrow T$ means (S,T) is not an edge in $\mathsf{DAG}_{\mathsf{max}}(\mathcal{S})$. Each user u just stores $\mathsf{k}(\{u\})$. The terms in the exponents are arranged so that even any collusion cannot compute keys that are not supposed to be computable by them. However, the number of primes used in the above schemes are too large as $|\mathcal{S}|$. Such primes will be stored as non-secret storage or derived on-the-fly.[3] We propose a new paradigm which makes uses of *prime powers* so that the number of primes used becomes optimal. We will see shortly that assigning prime powers depends essentially on a *chain decomposition* of $\mathsf{DAG}_{\mathsf{max}}(\mathcal{S})$. Indeed, the number of primes used will be exactly the number of chains; and each node in the same chain will correspond to the same prime but with a distinct power. For the toy example 1 in Fig.1, our new paradigm with the chain decomposition $\mathcal{C}_{\mathsf{toy1}}$ will result in only 5 primes used while the Akl-taylor's needs 9 primes. We will describe how to assign those powers over primes by an incidence matrix. We formalize the notion of incidence matrices that admit a secure scheme as *maximin matrix*:

Maximin Matrix. An $n \times m$ matrix $\{a_{ij}\}$ where $a_{ij} \in \mathbb{Z}_{\ge 0}$ is called a maximin matrix for set system X if for all $S \in \mathcal{S}_{\mathsf{X}}$, there exists j: $1 \le j \le m$ such that $\boxed{\max_{i\in S} a_{ij} < \min_{i\in N\setminus S} a_{ij}.}$ We give a formal treatment of RSA functions as accumulators and our construction first, then explain later.

RSA Accumulators. We fix a function $\mathsf{f} : \mathsf{U_f} \times \mathsf{E_f} \to \mathsf{U_f}$ to be an RSA function: $\mathsf{f}(x,e) := x^e \bmod \eta$ where $\eta = pq, p = 2p'+1, q = 2q'+1$ and p, q, p', q' are distinct odd primes. We restrict that $\mathsf{U_f}$ is the set of quadratic residues and $\mathsf{E_f}$ is the set of primes not equal to p', q'. We say f is generated from an RSA function generator $\mathsf{G_{RSA}}(1^\lambda)$. The function f is an instance of RSA accumulators,

[2] For example $C(5,3,2) = 3$ since $5 = 1+2+2 = 2+1+2 = 2+2+1$. The exact formula of $C(a,b,c)$ is quite complicated and is shown in [19].

[3] In the latter, a sequence of integers $\{x_j\}$ is pre-specified by the broadcaster and p_i is defined to be the first prime in $[x_i, x_{i+1})$; the program to recognize $\{x_j\}$ has negligible size (cf. [2]). More primes imply more computational cost on-the-fly.

first proposed in [5], which has a quasi-commutative property: for all $x \in \mathsf{U_f}$, and $e_1, e_2 \in \mathsf{E_f}$, $\mathsf{f}(\mathsf{f}(x, e_1), e_2) = \mathsf{f}(\mathsf{f}(x, e_2), e_1)$. If $E = \{e_1, ..., e_h\}$ where each $e_i \in \mathsf{E_f}$, then we denote $\mathsf{f}(x, E) := \mathsf{f}(\mathsf{f}(...\mathsf{f}(x, e_1), ...), e_h)$. Note that a set E is threaten as a multi-set, where the repetition of members is important. We thus denote a repetition of a member e which occurs t_e times as $t_e \lhd e$. For example, $\mathsf{f}(x, \{s \lhd e_1, t \lhd e_2\}) = x^{(e_1^s \cdot e_2^t)}$.

The Construction $(\mathsf{X})^{\mathsf{acc}}$.

Keygen : Run a $\mathsf{G_{RSA}}$ to obtain a description of $\mathsf{f} : \mathsf{U_f} \times \mathsf{E_f} \to \mathsf{U_f}$. Pick a random secret $s \in \mathsf{U_f}$. For $1 \leq j \leq m$, pick an element $p_j \in \mathsf{E_f}$. Let pub consist of all p_j's and $\{a_{ij}\}$; indeed we let user derive prime p_j only when necessary by predetermining the intervals of those primes (see below). Let

$$\begin{aligned} \mathsf{p}(u) &= \mathsf{f}(s, \{a_{uj} \lhd p_j : 1 \leq j \leq m\}), \\ \mathsf{k}(S) &= \mathsf{f}(s, \{(\max_{i \in S} a_{ij}) \lhd p_j : 1 \leq j \leq m\}). \end{aligned} \tag{7}$$

for user $u \in N$ and set $S \in \mathcal{S}_{\mathsf{X}}$.

Derive : Compute $\mathsf{k}(S) = \mathsf{f}(\mathsf{p}(u), \{(\max_{i \in S} a_{ij} - a_{uj}) \lhd p_j : 1 \leq j \leq m\})$.

Theorem 8 $(\mathsf{X})^{\mathsf{acc}}$ *is KINT-secure assuming the strong RSA assumption.*

First it is easy to see that the correctness holds: Derive is computable. Next we will give an intuition as to why for each $S \in \mathcal{S}$, the collusion of all users from $N \setminus S$ cannot compute the key of S. Informally, the best they can do is to obtain the value with the same base s and the exponent term being GCD of all the exponent terms of the keys for users in $N \setminus S$, which is $\prod_{j=1}^{m} p_j^{\min_{i \in N \setminus S} a_{ij}}$ (by the well-known trick involving using the extended Euclid's algorithm). To be able to compute the key of S, it must divide $\prod_{j=1}^{m} p_j^{\max_{i \in S} a_{ij}}$. But this will not happen due to the property of the maximin matrix.

Constructing a Maximin Matrix. Consider a chain decomposition $\mathcal{C} = \{G_1, ..., G_m\}$ of $\mathcal{S}_{\mathsf{X}}$. For each chain $G_j : S_1 \to \cdots \to S_l$, construct j-th column by letting

$$a_{ij} := \begin{cases} 0 \text{ if } i \in S_1 \\ w \text{ if } i \in S_{w+1} \setminus S_w \\ l \text{ otherwise} \end{cases} \tag{8}$$

Proposition 9 *The above construction is a maximin matrix. Moreover, $\mathcal{C}$ with the minimum number of chains will imply the maximin matrix with the minimum m, the number of all primes used.*

Proof. We will prove that the construction by Eq.(8) is a maximin matrix for X. Consider arbitrary $S \in \mathcal{S}$, observe that there is a chain $G_j : S_1 \to \cdots \to S_l$ and some w, $0 \leq w \leq l-1$, such that $S = S_{w+1}$ (since $\mathcal{C}$ is a chain decomposition). For all $i \in S$ we have $0 \leq a_{ij} \leq w$ by the construction. For all $i' \in N \setminus S$ we have $w > a_{i'j}$ also by the construction. This implies $\max_{i \in S} a_{ij} \leq w < \min_{i' \in N \setminus S} a_{i'j}$ which is what we wanted to prove. To prove the second claim, it is sufficient to prove the converse of the first claim: from any maximin matrix for X one can construct a a chain decomposition in which the number of chains is less than or equal to the number of columns of the matrix. The proof idea is essentially the same as the first, thus we omit the detail to the full version of this paper. □

Characterizing Efficiency. We will generate primes on the fly using the technique in [2] (cf. footnote 3). Without going into detail, this technique requires computational cost $O(\log^4 P)$ to generate one prime, and produces each prime of size $O(P \log P)$, where P is the number of all primes needed in such a scheme. In our scheme, $P = m$. Note that only when $P = O(1)$, it is worthless to use this technique; we just store the least P primes (which requires only negligible storage) so the cost for prime generation in this case is $O(1)$.

Using the notation defined earlier, we have that $\mathsf{RN}_{\mathcal{C}}(u)$ represents the number of chains in $\mathcal{C}$ that u appears; and $\mathsf{DD}_{\mathcal{C}}$ represents the length of the longest chain in $\mathcal{C}$. The number of all chains in $\mathcal{C}$ is $|\mathcal{C}|$ (and$= m$). We obtain:

$$\mathsf{KeySize}_{(\mathsf{X})^{\mathsf{acc}}}(u) = 1, \quad \mathsf{CompCost}_{(\mathsf{X})^{\mathsf{acc}}} = O(\mathsf{MC}^{\mathsf{acc}}_{\mathcal{C}} + \mathsf{PC}^{\mathsf{acc}}_{\mathcal{C}}),$$

where $\mathsf{MC}^{\mathsf{acc}}_{\mathcal{C}}(u), \mathsf{PC}^{\mathsf{acc}}_{\mathcal{C}}(u)$ are the cost due to Modular exponentiation and on-the-fly Prime generation for user u respectively and $\mathsf{MC}^{\mathsf{acc}}_{\mathcal{C}} := \max_{u \in N} \mathsf{MC}^{\mathsf{acc}}_{\mathcal{C}}(u)$, $\mathsf{PC}^{\mathsf{acc}}_{\mathcal{C}} := \max_{u \in N} \mathsf{PC}^{\mathsf{acc}}_{\mathcal{C}}(u)$. Such costs depend solely on $\mathcal{C}$ and can be characterized as:

$$\mathsf{MC}^{\mathsf{acc}}_{\mathcal{C}}(u) = O(\mathsf{DD}_{\mathcal{C}} \cdot (\log |\mathcal{C}|) \cdot \mathsf{RN}_{\mathcal{C}}(u)), \quad \mathsf{PC}^{\mathsf{acc}}_{\mathcal{C}}(u) = O((\log^4 |\mathcal{C}|) \cdot \mathsf{RN}_{\mathcal{C}}(u)).$$

The analysis are as follows. The cost of modular exponentiation for computing Derive is logarithm in the exponent term which is $\prod_{j=1}^{m} p_j^{(\max_{i \in S} a_{ij} - a_{uj})}$. To determine its complexity, observe that $\max_{i \in S} a_{ij} = a_{uj}$ for all but only $\mathsf{RN}_{\mathcal{C}}(u)$ terms of j due to Eq.(8) and the fact that u appears only $\mathsf{RN}_{\mathcal{C}}(u)$ chains. Also, observe that $\max_{i \in S} a_{ij} - a_{uj} \leq \mathsf{DD}_{\mathcal{C}}$ due to Eq.(8). Each p_j is $O(m \log m)$, hence has bit length $O(\log m)$. Combining these, we get $\mathsf{MC}^{\mathsf{acc}}_{\mathcal{C}}(u)$ as above. The cost for prime-generation above follows from the fact that the number of primes to be generated when deriving keys are $\mathsf{RN}_{\mathcal{C}}(u)$.

Remark 1. The MC of our scheme is asymptotically optimal among all non-trapdoor RSA-accumulator based paradigms (if there are any others) since it matches the lower bound in [11], which states that the optimal MC is of the same order as the number of subsets (in the set system) that one user is in, albeit here we calculate in bit complexity which includes the size of primes.

Remark 2. The Akl-Taylor's scheme [1] is a special case of our framework where the trivial chain decomposition (the collection of all one-node chains) is used.

5.2 Non-Trapdoor RSA based Instantiation for $\mathsf{SIC}, \mathsf{LSIC}$

Instantiating $\mathsf{SIC}, \mathsf{LSIC}$. We will state the result for LSIC so that the result for SIC can be obtained by setting $k = 1$. It suffices to define a chain decomposition of $\mathcal{S}_{\mathsf{LSIC}[k]}$ and the concrete scheme will follow automatically. We choose a chain decomposition $\mathcal{C}_{\mathsf{LSIC}[k]} = \mathcal{T}_{\mathsf{LSIC}[k]}$ defined in Eq.(6). (Note that it is obvious that $\mathcal{T}_{\mathsf{LSIC}[k]}$ was also a chain decomposition). A concrete example for $(\mathsf{SIC})^{\mathsf{acc}}$ is shown in Fig.5 for $n = 8$. As an example, the subset key $\mathsf{k}(567) = s^{(p_1^6 p_2^1 p_3^1 p_4^3 p_5^2 p_6^1 p_7^1 p_8^3)}$.

The following result follows directly from Theorem 4, 6 and the generic efficiency characterization of the framework with the fact that $|\mathcal{C}_{\mathsf{LSIC}[k]}| = n$.

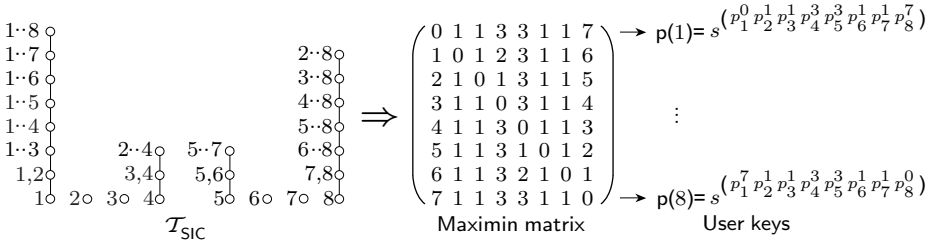

Fig. 5. Instantiating SIC ($n = 8$) by the non-trapdoor RSA accumulator based framework

Corollary 10 $\mathsf{MC}^{\mathsf{acc}}_{\mathcal{C}_{\mathsf{LSIC}[k]}} = O(kn^{1/k}\log^2 n)$ *and* $\mathsf{PC}^{\mathsf{acc}}_{\mathcal{C}_{\mathsf{LSIC}[k]}} = O(\log^5 n)$.

Scheme $(\mathsf{LSIC}[k])^{\mathsf{acc}}$ has computational cost $O(\max\{kn^{1/k}\log^2 n, \log^5 n\})$. For trillion users ($n = 10^{12}$), choose k as low as 4 we have $4n^{1/4}\log^2 n < \log^5 n$ so that the computational cost is dominant by the latter, which is roughly as in Asano's scheme (but ours enjoy exceptionally lower header size).

Remark 3. If we instantiate with with Akl-Taylor's, its chain decomposition has $\max_{u\in N} h_u = O(n^{1/k}\log n)$, and $m = O(2^k \cdot n^{1/k}(\log n)/k)$. Thus $\mathsf{PC} = O(n^{1/k}\log^5(n))$, which is much worse than ours, $O(\log^5 n)$. Moreover, this cost always dominates over the optimal MC for LSIC, $O(n^{1/k}\log^2 n)$.

6 Key Derivation based on Trapdoor RSA Accumulator

6.1 The New Trapdoor RSA based Framework

Framework Idea. The framework in this section is applicable to a class of posets that we call *tree-stratifiable posets*. Informally, such a poset of this type is defined as one which can be considered as formed by a tree hierarchy of atomic posets (not necessarily homogeneous), as shown in Fig.6. There, the graph decomposition $\mathcal{G} = \{G_x, G_y, G_z, ...\}$ is said to form a hierarchy represented by tree $\mathcal{H}$ where $V(\mathcal{H}) = \{x, y, z, ...\}$. Intuitively, such a graph decomposition is said to form a hierarchy if all the inclusion relations from every node in a lower subgraph (one with a lower index in the hierarchy), say G_y in the figure, to the next upper one in the hierarchy, G_x, are via a unique minimal node in that upper subgraph. Denote this minimal node as M_{G_y}. We will put a "dummy node" in each subgraph so that it will be the "representative" of that poset to reach that unique minimal node in the upper poset. (In the figure, the dummy node is D_{G_y} for subgraph G_y to reach M_{G_y}).

The idea for key derivations are as follows. First we define the key for each node in the highest sub-poset in the hierarchy by using the RSA-based framework in the last section. Recursively in a top-down fashion, we will define the set of keys corresponding to each lower sub-poset in the hierarchy. At some point, the set of keys for the nodes in G_x are defined. Then we define the "dummy key" for the dummy node in a next lower level sub-poset by applying a random permutation perm (w.l.o.g we will use the reverse direction) to the key of the minimal element in that upper sub-poset that it connects, that is, $\mathsf{k}(D_{G_y}) = \mathsf{perm}^{-1}(\mathsf{k}(M_{G_y}))$. To define keys for the other nodes in this lower sub-poset (at

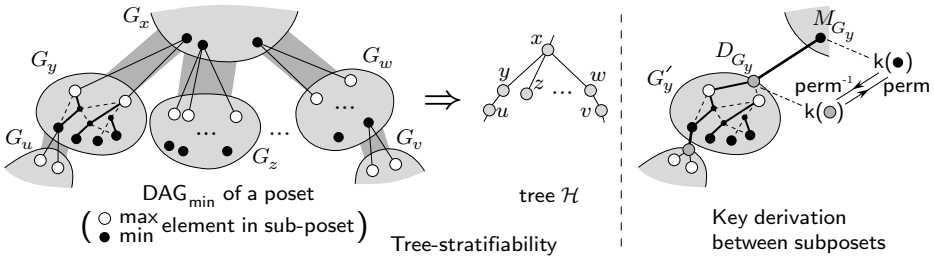

Fig. 6. The underlying idea for the trapdoor RSA based framework

G_y), we will again use the RSA-based framework for that sub-poset. However, this time the key for the dummy node has been already determined, while all the keys must agree with the relations of $(G'_y)^{\mathsf{acc}}$, where G'_y is the modified subgraph that includes the dummy node, i.e., the relation of keys as defined in Eq.(7) instantiated to a poset that has G'_y as its representation. To solve this, it suffices to use the *trapdoor* of RSA. In this way, we can define keys recursively until reaching the lowest sub-posets. Users, on the other hand, do not have to use trapdoor since they only compute keys in the bottom-up fashion. Note that $(\mathsf{perm}, \mathsf{perm}^{-1})$ is a public permutation, such as any block cipher with a fixed known key. We will model perm as an ideal random permutation in the security proof (the random permuation model).

The idea of reducing the whole poset by instantiating RSA-based framework in each sub-poset results in the use of only small number of primes for the overall scheme since the same set of primes can be used across different instantiations for different sub-posets.

To formalize this, we first define some more notations. For a directed graph G, denote $V_{\mathsf{min}}(G)$ the set of all minimal elements of poset $\mathcal{S}$ such that $\mathsf{DAG}_{\mathsf{min}}(\mathcal{S}) = G$. $V_{\mathsf{max}}(G)$ is defined analogously. The definition below captures what we have explained in the framework idea. Essentially, the bijection π below maps $G_x \mapsto x$.

Definition 3 (Tree-Stratifiable Poset) *An inclusion poset $\mathcal{S}$ is called* tree-stratifiable poset *iff there exist an induced graph decomposition $\mathcal{G}$ of $\mathcal{S}$ and a tree $\mathcal{H}$ with a bijection $\pi : \mathcal{G} \to V(\mathcal{H})$ such that for each $G \in \mathcal{G}$ if we define G' by letting $V(G') = V(G) \cup \{D_G\}$ and $E(G') = E(G) \cup \{(S, D_G) : \ S \in V_{\mathsf{max}}(G)\}$ where D_G is a dummy node; define $M_G := \bigcup_{S \in V_{\mathsf{max}}(G)} S$; and define a graph $\mathcal{W}$ by letting $V(\mathcal{W}) = \bigcup_{G \in \mathcal{G}} V(G')$ and $E(\mathcal{W}) = \bigcup_{G \in \mathcal{G}} \big(E(G) \cup \{(D_G, M_G)\}\big)$, then we have that (1) for all $G \in \mathcal{G}$, $M_G \in V_{\mathsf{min}}(\pi^{-1}(\mathsf{parent}_{\mathcal{H}}(\pi(G))))$ and (2) $E(\mathsf{DAG}_{\mathsf{min}}(\mathcal{S})) \subseteq E(\mathsf{DAG}_{\mathsf{max}}(\mathcal{W}))$.* □

Trapdoor RSA Accumulators. A trapdoor RSA function generator $\mathsf{G}_{\mathsf{tRSA}}$ is the one that works exactly the same as $\mathsf{G}_{\mathsf{RSA}}$ but in addition also outputs the *trapdoor* td which is $\phi(\eta)$ where ϕ is the Euler's phi function. With td, given the description of f, any $y \in \mathsf{U}_{\mathsf{f}}$, and a (multi-)set of accumulated values E, one can efficiently compute $x \in \mathsf{U}_{\mathsf{f}}$ such that $\mathsf{f}(x, E) = y$. Denote such x by $\mathsf{f}_{\mathsf{td}}(y, E^{-1})$.

Towards formalizing the construction, we "normalize" each sub-poset $G \in \mathcal{G}$ so that its base set will be $B_G = \{1, ..., |V_{\mathsf{min}}(G')|\}$ as follows. Construct γ :

$V(G') \to 2^{B_G}$ by first picking an injective map $\tilde{\gamma} : V_{\mathsf{min}}(G) \to B_G$ then define for $S \in V(G')$, $\gamma : S \mapsto \{\tilde{\gamma}(U) : \ U \in V_{\mathsf{min}}(G),\ U \subseteq S\}$. Let $\mathcal{S}_G = \gamma(V(G'))$ (the set of all images by γ from $V(G')$) be the set system with the base set B_G.

The Construction $(\mathsf{X})^{\mathsf{tacc}}$**.** For simplicity we will consider homogeneously stratifiable poset, i.e., each $\mathcal{S}_G$ is isomorphic to each other (in the sense that its corresponding DAG is isomorphic), say the set system Y. Let $\{a_{ij}\}_{1 \le i \le d, 1 \le j \le m}$ be a maximin matrix for set system Y, where d is the cardinality of its base set.

Keygen : Run a $\mathsf{G}_{\mathsf{tRSA}}$ to obtain a description of $\mathsf{f} : \mathsf{U}_{\mathsf{f}} \times \mathsf{E}_{\mathsf{f}} \to \mathsf{U}_{\mathsf{f}}$ and trapdoor td. For $1 \le j \le m$, pick an element $p_j \in \mathsf{E}_{\mathsf{f}}$. Let perm and perm^{-1} be a publicly available permutation mapping $\mathsf{U}_{\mathsf{f}} \to \mathsf{U}_{\mathsf{f}}$. Let pub consist of all p_j's and $\{a_{ij}\}$. Pick a random $t \in \mathsf{U}_{\mathsf{f}}$. Define keys recursively in a top-down fashion in the tree $\mathcal{H}$:

[**Top**]. At the subgraph $G_{\mathsf{root}} \in \mathcal{G}$, where root is the root of $\mathcal{H}$, by definition we have $N = M_{G_{\mathsf{root}}}$. We let $\mathsf{k}(N) = \mathsf{k}(M_{G_{\mathsf{root}}}) = t$.

[**Intermediate**]. At each atomic subgraph $G \in \mathcal{G}$, the key $\mathsf{k}(M_G)$ is previously determined. Define the key for the dummy node: $\mathsf{k}(D_G) = \mathsf{perm}^{-1}(\mathsf{k}(M_G))$. By using the trapdoor td and $\mathsf{k}(D_G)$, we solve Eq.(11) by setting $S = D_G$ (thus $\gamma(S) = B_G$) to determine the secret s_G, i.e.,

$$s_G = \mathsf{f}_{\mathsf{td}}(\mathsf{k}(D_G), \{(\max_{i \in B_G} a_{ij}) \lhd p_j : 1 \le j \le m\}^{-1}). \tag{9}$$

Then we define the key at each element in this subgraph, $S \in V(G)$, by:

$$\mathsf{k}(S) = \mathsf{f}(s_G, \{a_{\tilde{\gamma}(S),j} \lhd p_j : 1 \le j \le m\}) \qquad (\text{for } S \in V_{\mathsf{min}}(G)), \tag{10}$$

$$\mathsf{k}(S) = \mathsf{f}(s_G, \{(\max_{i \in \gamma(S)} a_{ij}) \lhd p_j : 1 \le j \le m\}) \qquad (\text{for } S \in V(G)). \tag{11}$$

[**Bottom**]. For each $u \in N$, we let $\mathsf{p}(u) = \mathsf{k}(\{u\})$.

Derive : Compute from the relations given in Eq.(9),(10),(11) but in the *bottom-up* fashion by using applications of $\mathsf{f}(\cdot,\cdot)$, $\mathsf{perm}(\cdot)$ starting from $\mathsf{f}(\mathsf{p}(u),\cdot)$. Note that td is not required to do this.

Theorem 11 $(\mathsf{X})^{\mathsf{tacc}}$ *is KINT-secure in the random permutation model (*perm *as an ideal random permutation), assuming the strong RSA assumption.*

Characterizing Efficiency. If the set system X of n users is tree-stratifiable homogeneously into a set system Y of d users with the tree $\mathcal{H}$ then

$$\mathsf{KeySize}_{(\mathsf{X})^{\mathsf{tacc}}}(u) = 1, \quad \mathsf{CompCost}_{(\mathsf{X})^{\mathsf{tacc}}} = O(\mathsf{MC}^{\mathsf{tacc}}_{\mathsf{X}} + \mathsf{PC}^{\mathsf{tacc}}_{\mathsf{X}}),$$

where the cost from modular exponentiation and prime generation are depended solely on both $\mathcal{H}, \mathsf{Y}$ and only Y respectively, and can be characterized as:

$$\mathsf{MC}^{\mathsf{tacc}}_{\mathsf{X}} = \mathsf{h}_{\mathcal{H}} \cdot \mathsf{MC}^{\mathsf{acc}}_{\mathcal{C}_{\mathsf{Y}}}, \quad \mathsf{PC}^{\mathsf{tacc}}_{\mathsf{X}} = \mathsf{PC}^{\mathsf{acc}}_{\mathcal{C}_{\mathsf{Y}}}, \tag{12}$$

where $\mathsf{h}_{\mathcal{H}}$ is the deepest depth of $\mathcal{H}$. The first claim follows from the fact that a user has to compute Eq.(11) for at most $\mathsf{h}_{\mathcal{H}}$ times. The second claim is from the fact that we reuse the same set of primes across sub-posets. There is also the cost due to applications of perm, which is $O(\mathsf{h}_{\mathcal{H}})$, but this is suppressed by MC.

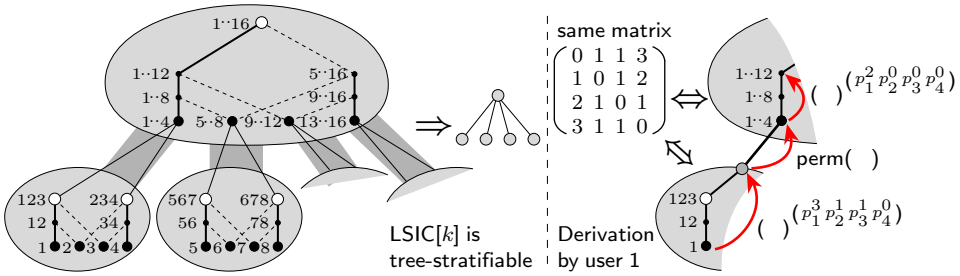

Fig. 7. Instantiating $\mathsf{LSIC}[k]$ ($n = 16, k = 2$, see Fig.3) by trapdoor RSA based framework

Generic Application. We now confine our interest to the case where $\mathcal{H}$ is the balanced completed $n^{1/k}$-ary tree of depth $\mathsf{h}_{\mathcal{H}} = k$. This forces the base sets of Y and X to have cardinality $n^{1/k}$ and n respectively. In this case we say $\mathsf{X} = \mathsf{hier}_k(\mathsf{Y})$. The operation hier_k is well-defined and can be thought as the converse direction of tree-stratification; thus, from any poset Z one can construct a tree-stratifiable poset, namely $\mathsf{hier}_k(\mathsf{Z})$, by first scaling down the cardinality of the base set of Z to $n^{1/k}$. (Since usually any set system is originally defined in term of n). We write $\mathsf{Z}(n^{1/k})$ to emphasize the cardinality of base set. The point is that when k is a constant, Eq.(12) allows one to construct a full scheme of n users but with exactly the same asymptotic performances as those of $(\mathsf{Z}(n^{1/k}))^{\mathsf{acc}}$, which is a "scaled-down" scheme, in *both* parameters MC, PC! Moreover, if $\mathsf{c}_{\mathsf{Z}(n)}(n, r) = O(r)$ then we can show that $\mathsf{c}_{\mathsf{hier}_k(\mathsf{Z}(n^{1/k}))}(n, r) = O(kr) = O(r)$ (by exactly the same proof as that of Theorem 2); therefore, $\mathsf{HeaderSize}$ is also unaffected.

6.2 Trapdoor RSA based Instantiation for LSIC

It is easy to see that $\mathsf{LSIC}[k]$ is tree-stratifiable since $\mathsf{LSIC}[k] = \mathsf{hier}_k(\mathsf{SIC}(n^{1/k}))$. (We could have define LSIC via hier operation rather than directly in Sec.3.2). An example is shown in Fig.7. From the efficiency characterization we have:

Corollary 12 (i) $\mathsf{MC}^{\mathsf{tacc}}_{\mathsf{LSIC}[k]} = O(n^{1/k}(\log^2 n)/k)$, $\mathsf{PC}^{\mathsf{tacc}}_{\mathsf{LSIC}[k]} = O((\log^5 n)/k^5)$. (ii) $\mathsf{MC}^{\mathsf{tacc}}_{\mathsf{LSIC}[\log_a n]} = O(a \log a \log n)$, $\mathsf{PC}^{\mathsf{tacc}}_{\mathsf{LSIC}[\log_a n]} = O(1)$.

Proof. See that $\mathsf{MC}^{\mathsf{acc}}_{\mathcal{C}_{\mathsf{SIC}(n^{1/k})}} = O((n^{1/k} \log^2 n)/k^2)$, $\mathsf{PC}^{\mathsf{acc}}_{\mathcal{C}_{\mathsf{SIC}(n^{1/k})}} = O((\log^5 n)/k^5)$; and $\mathsf{MC}^{\mathsf{acc}}_{\mathcal{C}_{\mathsf{SIC}(a)}} = O(a \log^2 a)$, $\mathsf{PC}^{\mathsf{acc}}_{\mathcal{C}_{\mathsf{SIC}(a)}} = O(1)$. (In fact, for the case $\mathsf{SIC}(a)$, the maximum number of primes used per user is $\log a + 1$, a small constant). □

7 Concluding Remarks

We presented three generic frameworks for constructing broadcast encryption and give some efficient instantiations. Almost all subset-cover broadcast encryption schemes based on PRSG (or one-way function) or RSA accumulator in the literature can be rewritten as instantiations in our paradigms. In fact, [18, 14, 17, 4, 12, 20, 15] can be viewed as PRSG-instantiated schemes and [2, 3, 11] are non-trapdoor-RSA-instantiated schemes from our frameworks.

The whole paradigm abstracts away the computational security issues and reduces the problem to only pure combinatorics. We leave as an open problem

the question of showing any combinatorial bound from the efficiency characterization in each sub-framework. Note that the previous bounds for broadcast encryption [16] are done in the setting where no key derivation is involved.

Acknowledgements. We would like to thank Kazukuni Kobara, Ryo Nojima, and also anonymous reviewers for their valuable comments on earlier versions.

References

1. S. G. Akl, P. D. Taylor, "Cryptographic Solution to a Problem of Access Control in a Hierarchy," ACM Transactions on Computer Systems, Vol. 1, No. 3 (1983), pp. 239-248.
2. T. Asano, "A Revocation Scheme with Minimal Storage at Receivers," ASIACRYPT 2002, LNCS 2501, pp. 433-450.
3. N. Attrapadung, K. Kobara, H. Imai, "Broadcast Encryption with Short Keys and Transmissions," ACM Workshop on Digital Rights Management, 2003.
4. N. Attrapadung, K. Kobara, H. Imai, "Sequential Key Derivation Patterns for Broadcast Encryption and Key Predistribution Schemes," ASIACRYPT 2003, LNCS 2894, pp. 374-391.
5. J. Benaloh, M. de Mare,"One-way accumulators: A decentralized alternative to digital signatures," EUROCRYPT 1993, LNCS 765, pp. 274-285.
6. D. Boneh, A. Silverberg, "Applications of Multilinear Forms to Cryptography," Contemporary Mathematics, 324, pp. 7190, (2003).
7. D. Boneh, C. Gentry, B. Waters, "Collusion Resistant Broadcast Encryption With Short Ciphertexts and Private Keys," CRYPTO 2005, LNCS 3621 (to appear).
8. G. C. Chick, S. E. Tavares, "Flexible Access Control with Master Keys," CRYPTO89, LNCS 435, pp. 316-322.
9. R. Diestel, "Graph theory," 2nd ed., Graduate texts in mathematics 173, (2000).
10. Y. Dodis, J. Katz, "Chosen-Ciphertext Security of Multiple Encryption," TCC 2005, LNCS 3378, pp. 188-209.
11. C. Gentry, Z. Ramzan, "RSA Accumulator Based Broadcast Encryption," ISC 2004, LNCS 3225, pp. 73-86.
12. M. T. Goodrich, J.Z. Sun, R. Tamassia, "Efficient Tree-Based Revocation in Groups of Low-State Devices," CRYPTO 2004, LNCS 3152, pp. 511-527.
13. A. Fiat, M. Naor, "Broadcast Encryption," CRYPTO 1993, LNCS 0773, pp. 480-491.
14. D. Halevy, A. Shamir, "The LSD Broadcast Encryption Scheme," CRYPTO 2002, LNCS 2442, pp. 47-60.
15. N. Jho, J.Y. Hwang, J.H. Cheon, M.H. Kim, D.H. Lee, E.S. Yoo, "One-Way Chain Based Broadcast Encryption Schemes," EUROCRYPT 2005. LNCS 3494, pp. 559-574.
16. M. Luby, J. Staddon, "Combinatorial Bounds for Broadcast Encryption," EUROCRYPT 1998, LNCS 1403, pp. 512-526.
17. M.J. Mihaljevic, "Key Management Schemes for Stateless Receivers Based on Time Varying Heterogeneous Logical Key Hierarchy," ASIACRYPT 2003, LNCS 2894, pp. 137-154.
18. D. Naor, M. Naor and J. Lotspiech, "Revocation and Tracing Schemes for Stateless Receivers," CRYPTO 2001, LNCS 2139, pp. 41-62.
19. Z. Star, "An Asymptotic Formula in the Theory of Compositions," Aequationes Math (1976).
20. P. Wang, P. Ning, D.S. Reeves, "Storage-Efficient Stateless Group Key Revocation," ISC 2004, LNCS 3225, pp. 25-38.