Sequential Key Derivation Patterns for Broadcast Encryption and Key Predistribution Schemes

Nuttapong Attrapadung, Kazukuni Kobara, 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,{kobara,imai}@iis.u-tokyo.ac.jp

Abstract. We study two closely related primitives: Broadcast Encryption and Key Predistribution Schemes (KPS). Broadcast Encryption allows a broadcaster to broadcast an encrypted message so that only a designated group of users can decrypt it. KPS allows a designated group of users to establish a common key non-interactively. We discover a generic method to construct efficient broadcast encryption schemes and KPSs naturally from Pseudo-Random Sequence Generators (PRSG) by observing that there are general "patterns" to do so. The two currently best PRSG-based broadcast encryption schemes such as the "Subset Difference" (SD) scheme by Naor Naor and Lotspiech and its refinement, the "Layered SD" (LSD) scheme by Halevy and Shamir, are indeed two special cases of our method. We demonstrate the power of this generic method by giving: (1) A solution to the most challenging variant of KPS: the one which supports arbitrary number of users to form a group yet secure against any collusion. We obtain a lower bound of the private key size at each user for any PRSG-based KPSs in this setting and construct a KPS that meets this bound. (2) An evidence that previous PRSG-based BE schemes, such as SD and LSD, can be further improved without any further assumption using this general method. We construct "Flexible SD" and "Flexible LSD" broadcast encryption schemes, which require less private key size while still maintain exactly the same broadcast size compared to their original SD/LSD schemes.

1 Introduction

Our main contribution is a generic method to construct efficients schemes of the two following closely related primitives naturally from Pseudo-Random Sequence Generators (PRSG). The primitives are:

Key Predistribution Scheme. Key Predistribution Scheme (KPS) involves n users. Each user is given a unique private key. For a group of users $P \subseteq N = \{1, ..., n\}$, any users in P should be able to non-interactively compute a common key k_P using only its private key while other receivers outside P should not be able to do so even if they collude. Such a scheme is motivated by the scenario of secure conferences over network. KPS can be viewed as a special case of broadcast encryption as we will see below.

Broadcast Encryption. Broadcast encryption (BE) involves 1 broadcaster and n receivers. Each receiver is given a unique private key. The broadcaster is given a 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 other receivers outside P should not be able to do so even if they collude. A broadcast encryption scheme is sometimes called a *Revocation Scheme*, where one is interested in a subset of non-privileged users or so-called revoked users rather than privileged ones hence its name. Such a scheme is motivated largely by pay-TV systems, the distribution of copyrighted material.

Relating 2 Primitives. In BE, a body of message is typical long and should be encrypted by a key commonly known to P. We call such a key a message encryption key Mek. To share Mek among P the broadcaster produces a header Hdr such that given Hdr and a private key of user in P one can obtain Mek. If the private keys are generated by KPS, each user in P already has a common key before hand thus there is no need of Hdr in this case. In this sense, KPS is known as *zero-header* BE.

Overview on Previous Works. KPSs were introduced by Blom [5] and formalized by Matsumoto-Imai [20]. Broadcast encryption schemes were first formally studied by Fiat-Naor [13]. Since then, many variants of the basic problem of KPSs and BEs are proposed. The relations of two primitives are also captured in many works (see, e.g., [18]). Since a KPS can be viewed as a special case of a BE, each variant of KPSs will be a variant of BEs (but not the converse). Keep in mind that a KPS is a zero-header BE. Therefore it is enough to describe variants of BEs as follows. To name just a few, the scheme might support bounded or unbounded number of privileged users and/or the maximum number of adversarial coalition; the privileged subset of users can be fixed, slowly changing, rapidly changing; the keys stored by each user can be stateful or stateless (to be updated or not); it might be possible to trace a traitor who illegally leak its secret key in the scenario so-called *tracing-traitor*; the scheme might be symmetric-key or public-key; and so on. We found that it is convenient to categorize the relevant schemes by their approaches as follows. For BEs there are (1)combinatorial approaches: schemes using combinatorial design such as [6, 23, 19, 18, 17, 15, 4]; and schemes using tree structure such as [25, 24, 21, 16, 10, 3] and (2)algebraic approaches: schemes using secret sharing scheme on the exponent to perform ElGamal-like encryption such as [2, 22, 11, 12]. For KPSs almost all of them are using combinatorial approaches. Most of the past works for KPSs can be found in Kurosawa, et al. [18].

The Most Challenging Variant. We study the most challenging variant of BE (and KPS) where it supports unbounded number of users in privileged subsets; unbounded number of revoked users allowed to form adversarial coalition (adaptively by central adversary); the privileged subset does not depend on the history; the private key stored by each user is stateless, i.e., it is fixed from the initialization time. This combined variant is arguably the hardest but the most desired one especially stateless scenario as argued explicitly first by Naor-Naor-Lotspiech [21].

The Main Goal and Some Solutions. The main goal towards BE and KPS problems is to construct efficient schemes satisfying the above mentioned variant where for KPS: the private key size is small, and for BE: both the header size and the private key size are small in the function of n, $|P|$, or $r := n - |P|$. A BE scheme which solves above mentioned variant problem and satisfies good efficiency in only one side is trivial. On one side, the private key size is independent of n but the header size is linear in $|P|$. On the other side, the header size is zero but the private key size is exponential in n. Note that the latter is a trivial KPS, which is definitely inefficient, however, is considered the best known solution for the above mention variant of KPS. As opposed to KPS, there are many BE schemes which have efficiency far better than the trivial schemes. One solution which is considered a ground work to many consequent works is due to Naor-Naor-Lotspiech [21]. It associates each user with the leaf of a balanced binary tree yielding a scheme called complete subtree (CS) in which the header size is $O(r \log(n/r))$ and the private key size is $O(\log n)$.

Major improvement to this idea were the subset difference (SD) method in their same paper [21] and its refinement, layered SD method, by Halevi-Shamir [16]. Both obtain the header size $O(r)$. While the SD scheme obtains the private key size $O(\log^2 n)$, the LSD scheme obtains the private key size $O(\log^{1+\epsilon} n)$ for small $\epsilon > 0$. More recent improvement due to Asano [3] utilizes the master key technique of Chick-Tavares [8] on balanced a-ary tree version of CS where $a > 2$ (instead of binary tree). This scheme obtains the header size $O(r(\log_a(n/r) + 1))$ and the private key size $O(1)$. These 3 schemes are considered the current state of the art for BE in the sense that while SD/LSD scheme obtain less header size, utilize a weak computational assumption, obtain much less computational cost; Asano's scheme obtains minimum private key size.

The basic idea of the schemes above is a mechanism called *subset-cover framework* [21]. Such a scheme in this framework varies to one another by (1) an underlying collection of subsets of a particular form, and (2) techniques which make use of computational assumption to enable the generation of many *computational unrelated* private keys. The improvements of recent works are primarily due to sophisticated design of the underlying collection in (1) to shorten the header size, and utilization of technique in (2) to shorten the private key size.

Shortening Private Key Size. Various methods to shorten the private key size are depicted in Figure 1. We capture these in 4 types. For simplicity, let us consider KPS where $N = \{1, 2, 3\}$. Each user u is supposed to be able to compute the common key k_S of set $S \subseteq N$ where $u \in S$. In the trivial KPS, user $u \in N$ just stores $\{k_S : u \in S \subseteq N\}$ as the private key set. The goal now is to reduce the number of elements in the private key set (recall that KPS is zero-header BE thus we do not worry about reducing header for now).

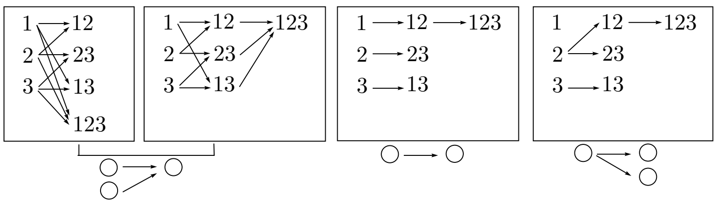

Fig. 1. Various methods to shorten key size: type 1-4 from left to right.

A natural way to do so is to let each user keep one master key which can be used to derive all common keys that he is supposed to be able to compute. This method is shown as type 1 in Figure 1. We denote by $A \rightarrow B$ where $A \subset B \subseteq N$ a one-way computation which takes as input k_A outputs k_B, i.e., one can easily compute k_B given k_A but given k_B it is hard to compute k_A. Note that this arrow notation applies to all types in the figure. Observe that every method in type 2 in fact also falls into type 1. The good functionality of one-way computation of these first 2 types is that: for any different inputs which are intractable to compute given one another, one can design a one-way computation such that it results in the same output. This functionality give these first 2 types the very short private key as one master key: $k_{\{u\}}$ for each user u.

The master key trick is originally introduced by Akl-Taylor [1] and Chick-Tavares [8] and brought to the context of BE first by Asano [3]. This trick uses the RSA assumption and falls into type 2. Another elegant trick for mastering key is proposed recently by Boneh-Silverberg [7]. Their scheme utilizes multilinear forms and assumes the Diffie-Hellman inversion assumption. This trick falls into type 1. Now that we have BEs with zero-header and the private key size as minimum as possible, so does this mean that we are done? The answer is no. The reasons are as follows. For the trick using multilinear form, unfortunately there is *no* known concrete construction of such forms up to date and it is believed that it is hard to find ones as argued by the authors themselves. For the trick using RSA, it turns out that such a scheme requires a large number of primes as $O(2^n)$ which is extremely inefficient both for storing and for generation. Note that, however, the trick using RSA works fine for non-zero-header BEs [3]. Nonetheless, for the trick using RSA in all cases, a critical disadvantage besides the issue of primes is a heavy computation due to the modular exponentiation with the exponent being products of many primes without knowing the order.

Type 3 is implicitly mentioned first in Akl-Taylor [1]. In this type the functionality of one-way computation as opposed to the first 2 types allows only one input per one output. User u just stores k_B whenever $u \in B \subseteq N$ and $u \notin A$ where $A \rightarrow B$ and A appears just before B in the diagram. The method of type 3 just makes use of any length-preserving one-way functions. A natural way to generalize this idea is to use any one-way functions that expand the length of inputs such as pseudo-random sequence generator (PRSG) $G_d(), d \in \mathbb{Z}^+$ which

d-ples the input, i.e., whose the bit length of the output is d times that of the input so that from one key, d many keys can be derived. This idea falls into type 4. The schemes which *implicitly* use this type of method in the context of BE are the SD [21] and LSD [16] method. Note that there is no KPS constructed by using this technique before. Shortening the private key size using only PRSG in type 4 has an obvious advantage over the one using RSA in type 2 since it needs no prime and its computation is much more efficient. Moreover, and perhaps more critical, is that the existence of PRSG is considered a weak assumption. Basing a cryptographic system only on a weak assumption is always preferred.

Motivation. The question is how one can use the method of type 4, shortening the private key size by utilizing PRSG, *at its most beneficial*. We take a look back into the schemes which implicitly utilize this method: the SD and LSD schemes. Although the balance tree representation which is used in SD/LSD has good properties since it somehow captures the nature of PRSGs in the sense that it utilizes PRSG by letting some values related to some parent nodes input to a PRSG so that a PRSG outputs some values related to their child nodes. However, the strict structure of subset difference is too rigid to capture the good properties from PRSGs thus the optimality of LSD scheme which argued by Halevi-Shamir [16] in their work is worked *only for their structure*. To obtain the most beneficial from PRSG, more flexible generalization of the idea must be rigorously captured.

Our Main Results. We observe that there are general "patterns" to construct broadcast encryption schemes and KPSs naturally from PRSGs. We call such a pattern a `sequential key derivation pattern`. We demonstrate the power of this general patterns by giving:

1. A solution to the most challenging variant of KPS: the one which supports arbitrary number of users to form a group yet secure against any collusion. We obtain a lower bound of the private key size at each user for any PRSG-based KPSs in this setting. This lower bound of the private key size appears to be $O(1/n)$ times that of information-theoretically secure KPS in the same setting [9]. We then propose an optimal construction of KPS which meets the bound. This construction makes use of a new combinatorial structure which is of an independent interest.
2. An evidence showing that previous best PRSG-based BE schemes, SD and LSD, can be further improved without any further assumption by using this general method. To do this, we construct "Flexible SD"(FSD) and "Flexible LSD"(FLSD) broadcast encryption schemes. Such schemes require, although asymptotically the same, less *exact* private key size while still maintain exactly the same broadcast size compared to the original SD/LSD schemes. More concretely, the number of private keys are reduced at most $\log n$ from the original schemes. This reduction depends on the user index. In particular, in the FSD and FLSD scheme, there are exactly $n/2$ users and $n/2^{\sqrt{\log n}}$ users who store exactly $\log n$ fewer keys than that of the SD and (Basic) LSD scheme respectively.

Table 1. Summary of results compared to previous works. The parameter in each KPS is the storage size at user: the first term in the addition is private key size, the second term is non-secret storage. The parameters in each BE consists of the private key size (in terms of the exact number of elements) in the upper row, and the header size (in terms of bound on the exact number of elements) in the lower row. The parameter w_u, z_u are functions of user index u, for $u \in N$ (see Theorem 2 and 3 for detail).

Based on $\rightarrow$	Multilinear form (type 1)	RSA (type 2)	PRSG (type 4)
Generic utilization of technique	[7]	[8]	(This work)
KPS	[7] $O(1) + O(n)$	[3](implicitly),[4] $O(1) + O(2^n \log n)$	(This work) $O(\frac{2^n}{n}) + 0$
BE	-	[3] 1 $r(\log_a(\frac{n}{r}) + 1)$	SD[21] $\text{key}_{SD} = (\log^2 n + \log n)/2 + 1$ $2r - 1$ (Basic)LSD[16] $\text{key}_{LSD} = \log^{3/2} n + 1$ $4r - 2$ FSD(This work) $\text{key}_{SD} - w_u$, $0 \leq w_u \leq \log n$, $u \in N$ $2r - 1$ FLSD(This work) $\text{key}_{LSD} - z_u$, $0 \leq z_u \leq \log n$, $u \in N$ $4r - 2$

2 Definitions

Definition 1 (Broadcast Encryption, BE). *A Broadcast Encryption Scheme (*BE*) is a 3-tuple of polynomial-time algorithms* (`Keygen`, `Encrypt`, `Decrypt`), *where:*

BE.`Keygen`$(1^\lambda, n)$: *Takes as input a security parameter 1^λ, the number of users n. It outputs n sets of receiver keys $I_1, ..., I_n$ and a sender key T.*
BE.`Encrypt`(P, T, Mek): *Takes as input a subset $P \subseteq N := \{1, ..., n\}$ of privileged users, the sender key T, and a message encryption key* Mek. *It outputs a header* Hdr *of the ciphertext.*
BE.`Decrypt`(P, Hdr, I_u): *Takes as input a subset $P \subseteq N$, a header* Hdr*, and a receiver key I_u. It outputs the message encryption key* Mek *that was sent if u was in the set P, or the special symbol $\perp$ otherwise.*

In practice, it is used in conjunction with a symmetric encryption algorithm F to encrypt the message body M under the message encryption key Mek *resulting in a broadcast body C_M. The broadcast to receivers consists of (P, Hdr, C_M).*

The security notion for broadcast encryption that we concern here is the one considered by Naor, et al. [21], where the security against chosen-ciphertext attack with adaptive user-corruption is defined.

In the same paper [21], the authors presented the subset-cover algorithm as a sufficient condition to construct such a broadcast encryption scheme. Here we recap its definition as the following. Notice that it is slightly different in context from the original one.

Definition 2 (Subset-Cover Algorithm, SC) *A subset cover algorithm* `SC` *is a 2-tuple of polynomial time algorithms* `(DefineSet, Cover)`*, where:*

`SC.DefineSet`(n) *:Takes as input the number of users n. It outputs a family $\mathcal{S}$ of subsets of N and a user structure Γ (for example, a binary tree of users).*

`SC.Cover`$(P, \mathcal{S})$ *:Takes as input a privileged subset P of users, the family $\mathcal{S}$ defined from* `DefineSet`*. It outputs a partition $\mathcal{S}_P := \{S_{i_1}, S_{i_2}, ..., S_{i_m} : S_{i_j} \in \mathcal{S}\}$ of P, i.e., $P = \bigsqcup_{j=1}^{m} S_{i_j}$, such that the number of subsets in its is the minimum among all possible partitions of P by $\mathcal{S}$.*

A broadcast encryption in the subset-cover framework is a broadcast encryption scheme that makes use of subset-cover algorithm as its subalgorithm as follows.

`BE.Keygen`$(1^\lambda, n)$ Run `SC.DefineSet`(n) to get $(\mathcal{S}, \Gamma)$. ¿From Γ, it determines *subset key* $\mathtt{k}(S_i)$ for each $S_i \in \mathcal{S}$. Then it defines I_u to be the set of λ-bits strings containing the minimal elements yet still being sufficient to easily deduce each subset key $\mathtt{k}(S_i)$ where $u \in S_i$ from I_u. The broadcaster key T is the set consisting of all the subset keys.

`BE.Encrypt`(P, T, Mek) Run `SC.Cover`$(P, \mathcal{S})$ to obtain $\{S_{i_1}, S_{i_2}, ..., S_{i_m}\}$. The Mek is encrypted by an encryption scheme E by each subset key $\mathtt{k}(S_{i_j}), j = 1, ..., m$ yielding a Hdr:

$$\langle (i_1, E_{\mathtt{k}(S_{i_1})}(\text{Mek})), ..., (i_m, E_{\mathtt{k}(S_{im})}(\text{Mek})) \rangle$$

`BE.Decrypt`(P, Hdr, I_u) Parse Hdr as $\langle (i_1, c_1), ..., (i_m, c_m) \rangle$, it finds i_j such that $u \in S_{i_j}$ (in case $u \notin P$ the result is `null`). Denote D the decryption algorithm corresponding to E. It uses I_u to derive $\mathtt{k}(S_{i_j})$ then computes $D_{\mathtt{k}(S_{i_j})}(c_j)$ to obtain Mek.

Also in the same paper [21], they define a security notion for broadcast encryption in the subset-cover framework namely, *Key Indistinguishability* (kIND) and prove that BE in the subset-cover framework is a secure broadcast encryption if it holds kIND and the corresponding encryption scheme E and F are IND-CCA1 secure. Therefore when proving the security of such a BE which is constructed in this framework, we just prove that it holds kIND. Informally, kIND says that any polynomial-time adversary can but with a negligible probability distinguish the subset key of privileged subset P of its choice from a random string of the same length even getting to know all private keys of users outside P.

Definition 3 (Key Predistribution Scheme, KPS). *A Key Predistribution Scheme* `KPS` *consists of a polynomial-time algorithms* `KeyGen`*, where:*

`KPS.Keygen`$(1^\lambda, n)$ *Takes as input a security parameter 1^λ, the number of users n. It outputs n sets of user keys $I_1, ..., I_n$ such that for $S \subseteq N$ and $S \neq \emptyset$, a* conference key $\mathtt{k}(S)$ *can be derived from I_u if and only if $u \in S$.*

Observe that we can use `KPS.Keygen` for `BE.Keygen` resulting in a broadcast encryption in the subset-cover framework in which $\mathcal{S}$ is a collection of all non-empty subsets of N. Consequently, we just let `BE.Encrypt`(P, T, Mek) output nothing (zero-header) and let Mek to be $\mathtt{k}(P)$. In this sense, KPS can be viewed as zero-header BE. Therefore, the security notion for KPS is indeed the key indistinguishability notion mentioned before.

3 Broadcast Encryption from PRSG

3.1 Generic Broadcast Encryption

We formally capture the nature of broadcast encryption scheme which is constructed from pseudo-random sequence generator into a general "pattern". We call such a pattern `Sequential Key Derivation Pattern` or SKDP. This pattern was actually explained briefly in the introduction as the type 4 method to shorten the private key size. We formalize it here. We begin by giving the definition of this pattern first and explain later.

Definition 4 (Sequential Key Derivation Pattern, SKDP). *Let $N := \{1, 2, ..., n\}$. Let Γ be a forest of rooted trees in which each node is labelled a different subset of N. We say that (N, Γ) is a* sequential key derivation pattern *if:*

1. *The label at each node which is not a root in each tree is a superset of the label at its parent node.*
2. *For every subset S of N, S can be partitioned into a disjointed union of subsets labelled at some nodes in Γ.*

Notation. A forest Γ is specified by a set of nodes and a set of edges. Since each node is labelled a different subset of N, we represent a node as its label. One edge is defined by an ordered pair of nodes directed toward from their root. A path from root to leaf is defined by an ordered set of nodes on that path directed toward from their root. We will call a path from root to leaf a *rl-path.* The set of all rl-paths in Γ is denoted by $\mathtt{Path}(\Gamma)$. An i-th node from root in rl-path $\mathbf{a}$ is denoted by $\mathbf{a}[i]$. Observe that for $\mathbf{a} \in \mathtt{Path}(\Gamma)$ it is true from the property 1 that $\mathbf{a}[i-1] \subset \mathbf{a}[i]$, thus we denote $\mathbf{a}[i] \setminus \mathbf{a}[i-1]$ by $\triangle\mathbf{a}[i]$ and call it a *differential label* at node $\mathbf{a}[i]$ for $i \geq 1$. Let $\triangle\mathbf{a}[0] = \mathbf{a}[0]$. For $\mathbf{a} \in \mathtt{Path}(\Gamma)$ define a *differential path* $\triangle\mathbf{a}$ as an ordered set of all differential labels in the rl-path $\mathbf{a}$. Denote the set of all differential paths in Γ by $\mathtt{DPath}(\Gamma)$. Denote $\bigtriangledown$ the opposite operation of $\triangle$, i.e., if $\mathbf{p} = \triangle\mathbf{a}$ then $\bigtriangledown\mathbf{p} = \mathbf{a}$ for $\mathbf{a} \in \mathtt{Path}(\Gamma), \mathbf{p} \in \mathtt{DPath}(\Gamma)$. For clarity, note that $\bigtriangledown\mathbf{p}[i] = (\bigtriangledown\mathbf{p})[i]$. Denote $\mathbf{a}_{\vdash i}$ the ordered set in which elements are taken from the first i elements of $\mathbf{a}$ in the same order. Also we often call a label an absolute label to distinguish it from a differential one.

An example of SKDP is the one shown as type 4 in Figure 1. A path $2 \rightarrow 12 \rightarrow 123$ is an example of rl-path. We represent it as $\mathbf{a} = (\{2\}, \{1, 2\}, \{1, 2, 3\})$. The corresponding differential path is thus $\triangle\mathbf{a} = (\{2\}, \{1\}, \{3\})$.

Intuitively, each node is assigned a subset key of its label (recall that its label is a subset of N). Informally, the property 1 in the above definition allows the one-way computation from a subset key of a node say v to subset keys of its child nodes say $v_1, ..., v_d$. The property 2 makes sure that the subset-cover algorithm can be used. Note that in the following generic scheme, we will not use subset key to compute another subset key directly but will use an *intermediate key* as we will see later.

The cryptographic primitive that is used for one-way computation is *pseudo-random sequence generator* $G_d(), d \in \mathbb{Z}^+$ that *d-ples* the input, i.e., whose output length is d times the length of the input. We say that $G_d : \{0,1\}^\lambda \mapsto \{0,1\}^{d\lambda}$ is a pseudo-random sequence generator if no polynomial-time adversary can distinguish the output of G_t on a random chosen seed from a truly random chosen string of similar length.

Generic Construction. Now we will formally describe the generic broadcast encryption in the subset-cover framework that makes use of SKDP (N, Γ). It is enough to specify only `SC.DefineSet`, `SC.Cover`, and `BE.Keygen` since `BE.Encrypt` and `BE.Decrypt` can be applied transparently from the last section.

`SC.Defineset`(n) It defines $\mathcal{S}$ as the sets of labels at all nodes in Γ. It also output Γ as given from SKDP.

`SC.Cover`$(P, \mathcal{S})$ Due to the property 2 of SKDP, each subset P of N can be partitioned into a disjointed union of subsets labelled at some nodes in Γ, thus a disjointed union of subsets in $\mathcal{S}$. It partitions the set P into $\mathcal{S}_P := \{S_{i_1}, S_{i_2}, ..., S_{i_m} : S_{i_j} \in \mathcal{S}\}$ with the minimum numbers of subsets.

`BE.Keygen`$(1^\lambda, n)$ Before specify the algorithm, the definitions of intermediate key, subset key, and their relation are specified first as follows:

Intermediate Key. Each subset $S_i \in \mathcal{S}$ is assigned an *intermediate key* $\mathtt{t}(S_i)$. Each user in S_i, say u, should be able to derive $\mathtt{t}(S_i)$ from I_u.

Subset key. Each subset $S_i \in \mathcal{S}$ is assigned a *subset key* $\mathtt{k}(S_i)$. A subset key $\mathtt{k}(S_i)$ can be derived from the intermediate key $\mathtt{t}(S_i)$. We say that a node is assigned a subset key $\mathtt{k}(S_i)$ if that node is labelled S_i.

Derivation. Let S_i be a subset labelled at a node which is not a leaf in Γ. Suppose that the outdegree of this node is d. Let $S_{i_1}, S_{i_2}, ..., S_{i_d}$ be subsets labelled at its children and $i_1 < ... < i_d$. The derivation is defined as:

$$\mathtt{t}(S_{i_1})||\mathtt{t}(S_{i_2})||...||\mathtt{t}(S_{i_d})||\mathtt{k}(S_i) := G_{d+1}(\mathtt{t}(S_i))$$

where $|\mathtt{t}(S_{i_1})| = ... = |\mathtt{t}(S_{i_d})| = |\mathtt{k}(S_i)| = \lambda$ bits and $||$ is concatenation. This recurrence relation is well defined if all the initial values, which in fact are all the intermediate keys assigned at root of unconnected trees of the forest, are defined.

Now we will specify `BE.Keygen`. It randomly chooses λ-bits strings in exactly the same number as the number of unconnected trees in Γ. It then assigns each string to be the intermediate key assigned at the root of each unconnected tree respectively. User u should be given the intermediate key assigned at the node whose label contains u, say node v, which appears first when looking from root to leaf in such a rl-path so that u can derive all the intermediate keys assigned at v's descendants, whose labels are some supersets of label at v; but not the ones assigned at v's ancestors, whose labels do not contain u. That is, I_u is the set of all intermediate keys at nodes whose differential labels contain u. Formally $I_u = \{\mathtt{t}(\mathbf{a}[i]) : u \in \triangle\mathbf{a}[i], \mathbf{a} \in \mathtt{Path}(\Gamma)\}$.

Theorem 1 *The above generic broadcast encryption scheme from* SKDP *satisfies the* kIND *property assuming secure pseudo-random sequence generator.*

Remark 1: A Note on Public Key Extension. Public-key extension of our generic broadcast encryption from SKDP can be constructed *directly* by utilizing Hierarchical Identity-Based Encryption [14] with the hierarchical tree obtained by connecting all roots of unconnected tree in Γ of SKDP to a new central root. This is indeed the same method as proposed by Dodis-Fazio [10], but we believe that our interpretation provides a better understanding.

3.2 Flexible SD/LSD Broadcast Encryption

We construct schemes called Flexible SD/LSD to demonstrate the power of our generic method. In general, the following mechanism is just one example of conversion from *any* PRSG-based broadcast encryption schemes into schemes which yielding less private key size while maintaining exactly the same header size. Moreover, with a further adaptation, we can reduce also the header size by trading off the computational cost. In particular, we apply this conversion to the SD scheme [21] and the LSD scheme [16] to get the Flexible SD and LSD schemes (FSD/FLSD) respectively. We call them flexible since their structures came from a flexible generalization of the idea by our general method.

We assume that the reader is familiar with the SD/LSD scheme. Observe that the SD/LSD schemes are indeed two such patterns of SKDP. The SKDP which implicitly used in SD is shown explicitly in Figure 2(left). The SKDP for LSD scheme is just an adaptation of SD scheme with the absence of some labels from SD scheme.

The conversion is very simple. Intuitively we just split the differential label of each node which is not a singleton subset of N in the original scheme into a union of differential labels which are singleton subsets and connect them in an appropriate order (step 1). After step 1, some of rl-paths will become sub-paths (from root) of some other rl-paths. Observe that such repetition parts represent the same sets of absolute labels. Therefore we can delete those sub-paths from the collection of all rl-paths since doing this does not affect the collection of absolute labels, and thus also the header size. These deletions reduce the private key size. In step 3, absolute labels which are not absolute labels in the original schemes are combined until there is no such case. We do this since these labels are not needed for subset-cover algorithm. They would only make rl-paths too long, consequently increase the computational cost. However, as we will see later, step 3 can be skipped and we will obtain a scheme which beside the private key size, the header size is also reduced. Note that the procedures in the following description are somewhat made redundant for a better understanding.

The FSD and FLSD Scheme. Let X be SD or LSD scheme. Let $(N, \varGamma_{\mathrm{X}})$ be a SKDP for X scheme. The conversion from X scheme to FX scheme is done as follows:

Step 1 For each $\mathbf{p} \in \mathtt{DPath}(\varGamma_{\mathrm{X}})$, let $|\mathbf{p}| = p$ and $|\mathbf{p}[i]| = k_i$ and do the following:

1. For $i : 0 \le i \le p-1$, parse $\mathbf{p}[i]$ as $\{a_{i,1}, ..., a_{i,k_i}\}$ where $a_{i,1} < \cdots < a_{i,k_i}$.
2. Define $f(\mathbf{p}) = (\{a_{0,1}\}, ..., \{a_{0,k_0}\}, \{a_{1,1}\}, ..., \{a_{1,k_1}\}, ..., \{a_{p,1}\}, ..., \{a_{p,k_p}\})$.

After all, let $A = \{f(\mathbf{p}) : \mathbf{p} \in \mathtt{DPath}(\varGamma_{\mathrm{X}})\}$.

Step 2 For any $\mathbf{v}, \mathbf{w} \in A$ such that $\mathbf{v} = \mathbf{w}_{\dashv|\mathbf{v}|}$, we decrement A to be $A \setminus \{\mathbf{v}\}$. Repeat this until there is no such case.

Step 3 For each $\mathbf{q} \in A$, if there is j such that $\triangledown\mathbf{q}[j] \notin \mathcal{S}_{\mathrm{X}}$ then renew $\mathbf{q}$ to be $(\mathbf{q}[0], ..., \mathbf{q}[j-1], \mathbf{q}[j] \cup \mathbf{q}[j+1], \mathbf{q}[j+2], ..., \mathbf{q}[|\mathbf{q}|-1])$. Repeat this until there is no such case.

Step 4 Finally we let $\mathtt{DPath}(\varGamma_{\mathrm{FX}}) = A$.

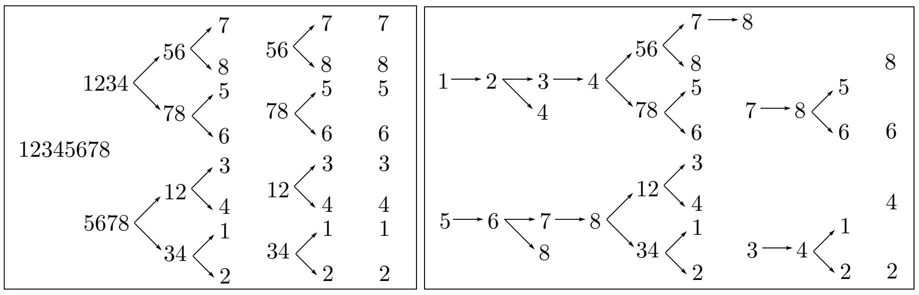

Fig. 2. Differential path representation of SKDP for SD (left) and FSD (right).

Theorem 2 *For $1 \le u \le n-1$, let $x_u := \max\{k : 2^k \mid u\}$. Then*

$$|I_{u,\mathrm{FSD}}| = |I_{u,\mathrm{SD}}| - \log n + x_u,$$

and $|I_{n,\mathrm{FSD}}| = |I_{n,\mathrm{SD}}|$. Recall that $|I_{u,\mathrm{SD}}| = (\log^2 n + \log n)/2 + 1$ for all $u \in N$.

Proof. It is enough to show how many differential paths containing u are deleted in step 2. We put the label N away for a while. For $1 \leq k \leq \log n$, let B_k be the collection of all k-length differential paths of SD. Wlog, we consider the deletions in $B_1, ..., B_{\log n - 1}$ respectively. By inspection, for each k, differential path $\mathbf{p} \in B_k$ will satisfy $f(\mathbf{p}) = f(\mathbf{p}')_{\dashv |f(\mathbf{p})|}$ for some $\mathbf{p}' \in B_{k'}, k' > k$ iff

$$f(\mathbf{p}) = (\{q2^k + 1\}, \{q2^k + 2\}, ..., \{q2^k + 2^k - 1\}),$$

for some $q \in \mathbb{Z}^+$. Thus there will be no deletion for user u iff $2^k \mid u$. Hence in the last step, when considering $k = \log n - 1$, the number of deletions for u will be $\log n - 1 - \max\{k : 2^k \mid u\} = \log n - 1 - x_u$. Finally, if $u \neq n$, then there is one more deletion from the label N thus the reduction will be $\log n - x_u$. $\square$

To illustrate this theorem, one can verify from Figure 2 that: in FSD, for $u = 1, 3, 5, 7 : |I_u| = 4$, for $u = 2, 6 : |I_u| = 5$, and $|I_4| = 6$,$|I_8| = 7$; while in SD, $|I_u| = 7$ for all $u \in N$. The following corollary follows directly from Theorem 2.

Corollary 1 *In the FSD scheme, for $1 \leq j \leq \log n$, there are exactly $n/2^j$ users whose the number of keys is $|I_{u,\mathrm{SD}}| - \log n - 1 + j$. And only one remaining user has $|I_{u,\mathrm{SD}}|$ keys.*

In the following theorem, we concern only the basic LSD (bLSD) scheme for simplicity. The results for general LSD schemes can be obtained similarly.

Theorem 3 *For $1 \leq u \leq n - 1$, let $x_u := \max\{k : 2^k \mid u\}$. Then*

$$|I_{u,\mathrm{FbLSD}}| = |I_{u,\mathrm{bLSD}}| - \log n + x_u + y_u,$$

where

$$y_u = |\{j : 1 \leq j \leq \log n, \sqrt{\log n} \nmid j, 2^j - 2^{\lfloor \frac{j}{\sqrt{\log n}} \rfloor \sqrt{\log n}} + 1 \leq u \bmod 2^j \leq 2^j - 1\}|,$$

and $|I_{n,\mathrm{FbLSD}}| = |I_{n,\mathrm{bLSD}}|$. Recall that $|I_{u,\mathrm{bLSD}}| = \log^{3/2} n + 1$ for all $u \in N$.

Intuitively, the term y_u comes from the number of differential labels containing u that were in $\mathcal{S}_{\mathrm{SD}}$ and would have been deleted in step 2, but are not in $\mathcal{S}_{\mathrm{bLSD}}$. Note that $y_u \leq \log n - x_u$ since we just add back what we would have deleted if it were SD scheme.

Analogous to Corollary 1, in the FbLSD scheme we could indeed show the exact number of users in the function of the number of keys. However, it turns out that the expressions are quite complex. We thus state only a particular case when the number of keys is the fewest to give some intuition as the following.

Corollary 2 *In the FbLSD scheme, there are exactly $n/2^{\sqrt{\log n}}$ users whose the number of keys is $|I_{u,\mathrm{bLSD}}| - \log n$.*

Theorem 4 *The FSD/FLSD scheme require the same header size as the original scheme for every instance and the computational cost bounded by $O(\log^2 n)$.*

We briefly prove this theorem. First, the header size is remain unchanged since $\mathcal{S}_{\mathrm{X}} = \mathcal{S}_{\mathrm{FX}}$. Second, it can be shown that the longest rl-path contains $(\log^2 n + \log n)/2$ edges which is $O(\log^2 n)$.

Remark 2: On Reducing the Header Size. If we skip step 3, then there are some labels which are in $\mathcal{S}_{\mathrm{FX}}$ but not in $\mathcal{S}_{\mathrm{X}}$. This means that in the FX scheme we have more choices to cover any privileged subsets hence the header size will be reduced for some instances of broadcast. On the down side, the longest rl-path will have length n resulting in increasing computational cost. Nevertheless, to skip or not to skip step 3 are the two extreme cases on the spectrum. In this sense, we can trade off the header size and the computational cost.

4 Key Predistribution Scheme from PRSG

4.1 Lower Bound

Theorem 5 *Every PRSG-based KPS satisfies* $\max_{u \in N} |I_u| \geq \lceil (2^n - 1)/n \rceil$.

Note that this is $O(1/n)$ less than the lower bound in the information-theoretically secure KPS in the same setting in which such a bound is 2^{n-1} [9].

A further fact from the proof of this theorem is that two necessary conditions to make an equality holds are: (i) The differential label at each node must be a singleton set. (ii) For each $u \in N$, $|I_u|$ is equal to $\lceil (2^n - 1)/n \rceil$ or $\lfloor (2^n - 1)/n \rfloor$.

4.2 An Optimal Construction

Intuition. To design a set of differential path $\mathtt{DPath}(\Gamma)$ to represent a SKDP for an optimal KPS, we have to make sure the following requirements[1]:

- It really represents a SKDP: the (absolute) labels at any two different nodes are different. For example, there is no such path from root (not necessary to leaf) $1 \rightarrow 12$ and $2 \rightarrow 12$ appear simultaneously; that is to say there is no differential path $1 \rightarrow 2 \rightarrow \cdots$ and $2 \rightarrow 1 \rightarrow \cdots$ at the same time.
- It can be used for KPS: the set of (absolute) labels at all nodes completes the set of non-empty subsets of N.
- It is optimal: for each $u \in N$ the number of differential labels which u appears is $\lceil (2^n - 1)/n \rceil$ or $\lfloor (2^n - 1)/n \rfloor$.

Consider the case where n is a prime larger than 2. When n is not a prime, a construction can be achieved similarly but is more complex. Theorem 5 can be intuitively interpreted as the following: First due to the fact (i), for each nonempty $S \subseteq N$ it must be that $\mathtt{t}(S) \in I_u$ for only one *unique* u. If we put $\mathtt{t}(N)$ away for a while. All other $2^n - 2$ intermediate keys for non empty-subset of N must be distributed equally to each u so that $|I_u| = \lfloor (2^n - 1)/n \rfloor = (2^n - 2)/n$. Note that this is an integer due to the Fermat's little theorem. Finally we pick one unlucky user say v and increment I_v to be $I_v \cup \{\mathtt{t}(N)\}$ so that only v, $|I_v| = \lceil (2^n - 1)/n \rceil = (2^n - 2)/n + 1$ and we will be done since this is optimal.

The question now is how to distribute $2^n - 2$ values of $\mathtt{t}(S), S \subset N, S \notin \emptyset$ equally to each $I_u, u \in N$. We accomplish this by first constructing a structure

[1] Figure 4(lower part) should be helpful to get some insight.

called *block*. One block contains n fix-length differential paths in which labels are all different. Each block has a property that the number of differential labels which u appears is equal for every $u \in N$. Thus we just let $\texttt{DPath}(\Gamma)$ to be composed of many blocks so that we would accomplish the task. However, the difficulty arises since we have to make sure also that any absolute labels from different blocks of the same length are different (as the requirement 1) and the set of all differential paths of length i completes the set of i-subsets of N (as requirement 2). This turns out to be a non-trivial task. We achieve this by defining an equivalence relation between blocks of the same length: informally two blocks are said to be equivalent if the set of labels from two blocks are the same. Therefore we just pick one block from each equivalence class into which this relation partitioned to completes $\texttt{DPath}(\Gamma)$ and we will be done. However, to pick a i-length block, it has to be consistent with some a $(i-1)$-length block picked previously in the sense that all absolute labels of the first i nodes away from the root in each path in two blocks are the same. To accomplish this, we define a relation called *splitting relation* between every two complete collections of classes in which length of blocks are consecutive, say $i-1$ and i. This relation will imply a set of chains that relate from block of length 1 to that of length 2 and so on. Consequently, instead of picking up a block, we will pick up a chain and we will be done. Note that due to (i), wlog, from now on we consider each differential label as an element in N instead of a singleton subset of N.

Building Blocks. Now we will formally define the structure "block" and its equivalence relation. Each block is generated by a vector from the space $D_k := \{(d_1, ..., d_k) \in (N_*)^k : \forall a < b, \sum_{j=a}^{b} d_j \not\equiv 0 \pmod{n}\}$ where we let $N_* = \{1, ..., n-1\}$. Figure 4(upper part) shows examples of block.

Definition 5 (Block) *For a set $F \subseteq N$ and a vector $\mathbf{d} = (d_1, ..., d_{k-1}) \in D_{k-1}$, we define a* block generated by $\mathbf{d}$ over F *as*

$$\langle \mathbf{d} \rangle_F = \{(a, a + d_1, a + d_1 + d_2, ..., a + d_1 + \cdots + d_{k-1}) \bmod n : \ a \in F\},$$

and denote it as $\langle \mathbf{d} \rangle_F$. When $F = N$, we simply denote $\langle \mathbf{d} \rangle$.

Recall that each element in a block is a differential path whose all differential labels are singleton subset of N so we can treat such a differential path as a vector for simplicity. Furthermore one can verify that if n is prime, no absolute labels from any different differential paths in the same block are the same.

Definition 6 (Equivalence Relation $\equiv$) *For vectors $\mathbf{d}, \mathbf{e} \in D_{k-1}$, we say that $\mathbf{d}$ is equivalent to $\mathbf{e}$ over set F, and write $\mathbf{d} \equiv_F \mathbf{e}$, if*

$$\{\triangledown \mathbf{p} : \mathbf{p} \in \langle \mathbf{d} \rangle_F\} = \{\triangledown \mathbf{p} : \mathbf{p} \in \langle \mathbf{e} \rangle_F\},$$

and when $F = N$, we simply denote $\mathbf{d} \equiv \mathbf{e}$.

It is easy to verified that $\equiv$ is an equivalence relation on the set D_{k-1}, i.e., it has reflexivity, symmetry, and transitivity. So what will the equivalence classes into which the relation $\equiv$ partitions D_{k-1} be like? First, we consider the following lemma. Let Π_k denote a set of all permutations of $(0, 1, ..., k-1)$.

Lemma 1 *For* $\mathbf{d} = (d_1, ..., d_{k-1}), \mathbf{e} = (e_1, ..., e_{k-1}) \in D_{k-1}$ *let* $d_0 := n - \sum_{j=1}^{k-1} d_j \bmod n, e_0 := n - \sum_{j=1}^{k-1} e_j \bmod n$*, we have* $\mathbf{d} \equiv \mathbf{e}$ *if and only if there exists* $(r_0, r_1, ..., r_{k-1}) \in \Pi_k$ *such that for all* $0 \le i \le k-1$, $e_i = \sum_{j=r_i}^{r_{i+1}-1 \bmod k} d_j \bmod n$*, where we let* $r_k := r_0$ *and* $\sum_{j=b}^{a} d_j := d_b + d_{b+1} + \cdots + d_{k-1} + d_0 + \cdots + d_a$ *when* $a < b$.

Definition 7 *For* $\mathbf{r} \in \Pi_k$, $\mathbf{d}' = (d_0, ..., d_{k-1}) \in (N_*)^k$ *such that* $\sum_{j=0}^{k-1} d_j \equiv 0 \pmod{n}$*, let*

$$\mathbf{d}' \triangleright \mathbf{r} := (\sum_{j=r_0}^{r_1-1 \bmod k} d_j, \sum_{j=r_1}^{r_2-1 \bmod k} d_j, ..., \sum_{j=r_{(k-1)}}^{r_0-1 \bmod k} d_j) \bmod n.$$

Lemma 1 implies that such an equivalence class is of the form $[\mathbf{d}'] := \{\mathbf{d}' \triangleright \mathbf{r} : \mathbf{r} \in \Pi_k\}$. To see the concrete classes, we first consider $\mathbf{d}', \mathbf{e}'$ such that $\sum_{j=1}^{k-1} d_j = \sum_{j=1}^{k-1} e_j = n$. One can verify that $[\mathbf{d}'] = [\mathbf{e}']$ if and only if $\mathbf{d}'$ is a cyclic permutation of $\mathbf{e}'$. Next observe that for an arbitrary $\mathbf{d} \in D_{k-1}$ there will be $\mathbf{r} \in \Pi_k$ and $\mathbf{v} = (v_0, ..., v_{k-1}) \in (N_*)^k$ where $\sum_{j=1}^{k-1} v_j = n$ such that $\mathbf{d} \in [\mathbf{v} \triangleright \mathbf{r}]$. Therefore the complete collection of these equivalence classes is the collection of classes $[\mathbf{v}]$ where each $\mathbf{v}$ is a *cyclic positive k-partition* of n. We denote this collection of equivalence classes as $\mathcal{E}_{n,k}$. For example, $\mathcal{E}_{6,3} = \{[411], [321], [312], [222]\}$.

¿From now, if we write $[\mathbf{v}] \in \mathcal{E}_{n,k}$ it is to be understood that $\mathbf{v} = (v_0, ..., v_{k-1}) \in (N_*)^k$ and $\sum_{j=0}^{k-1} v_j \equiv n \pmod{n}$ unless something else specified; in addition we will say that $\mathbf{v}$ is a representative vector of class $[\mathbf{v}]$.

Definition 8 (Splitting Relation) *A splitting relation* $\texttt{Splt}_k \subset \mathcal{E}_{n,k} \times \mathcal{E}_{n,k+1}$ *is defined as* $\texttt{Splt}_k := \{([\mathbf{v}], [\mathbf{y}]) : [\mathbf{v}_{\dashv k-1} \| (v_{k-1} - a \bmod n, a)] = [\mathbf{y}], a \in \mathbb{Z}_{n-1}\}$.

This definition is well defined: we do not aware which representative vector of such a class is to be splitted, i.e., we claim the following lemma:

Lemma 2 *For any* $\mathbf{w}$ *such that* $\mathbf{w} = \mathbf{v} \triangleright \mathbf{r}$ *for some* $\mathbf{r} \in \Pi_k$ *there will be* $\mathbf{s} \in \Pi_{k+1}$ *and* $b \in \{1, ..., n-1\}$ *such that* $\mathbf{w}_{\dashv k-1} \| (w_{k-1} - b \bmod n, b) = (\mathbf{v}_{\dashv k-1} \| (v_{k-1} - a \bmod n, a)) \triangleright \mathbf{s}$.

To prove this Lemma, choose $\mathbf{s} = \mathbf{r} \| (k)$ and $b = w_{k-1} - v_{k-1} + a \bmod n$.

Lemma 3 *For* $2 \le k \le \lceil n/2 \rceil$ *there exists* onto *function* $f_{n,k} : \mathcal{E}_{n,k} \overset{onto}{\mapsto} \mathcal{E}_{n,k-1}$ *such that for all* $[\mathbf{v}] \in \mathcal{E}_{n,k}$, $(f_{n,k}([\mathbf{v}]), [\mathbf{v}]) \in \texttt{Splt}_{k-1}$*. For* $\lceil n/2 \rceil + 1 \le k \le n-1$ *there exists* one-to-one *function* $g_{n,k} : \mathcal{E}_{n,k} \overset{1-1}{\mapsto} \mathcal{E}_{n,k-1}$ *such that* $g_{n,k}^{-1} \subset \texttt{Splt}_{k-1}$.

For the functions denoted above, when $f_{n,k}([\mathbf{v}^k]) = [\mathbf{v}^{k-1}]$ $(2 \le k \le \lfloor n/2 \rfloor)$ or $g_{n,k}([\mathbf{v}^k]) = [\mathbf{v}^{k-1}]$ $(\lceil n/2 \rceil + 1 \le k \le n-1)$, we will represent it as $[\mathbf{v}^{k-1}] \to [\mathbf{v}^k]$. Let $C = \{f_{n,2}, ..., f_{n,\lceil n/2 \rceil}, g_{n,\lceil n/2 \rceil+1}, ..., g_{n,n-1}\}$. A chain $[\mathbf{v}^{k_1}] \to [\mathbf{v}^{k_1+1}] \to \cdots \to [\mathbf{v}^{k_2}]$ is said to be induced by C if every $\to$ in the chain is taken from a mapping in a function in C. Such a chain is said to be $(k_1, k_2) -$ *terminated* if there is no $\to$ directed into $[\mathbf{v}^{k_1}]$ and no $\to$ directed from $[\mathbf{v}^{k_2}]$. Since $f_{n,k}$ is onto function, each terminated chain is $(1, k)$-terminated for some $\lceil n/2 \rceil \le k \le n-1$.

The Optimal Construction.

Step 1 Find a set of functions $A = \{f_{n,k} : \mathcal{E}_{n,k} \overset{onto}{\mapsto} \mathcal{E}_{n,k-1} | 2 \leq k \leq \lceil n/2 \rceil\}$ and $B = \{g_{n,k} : \mathcal{E}_{n,k} \overset{1-1}{\mapsto} \mathcal{E}_{n,k-1} | \lceil n/2 \rceil + 1 \leq k \leq n-1\}$ which satisfy Lemma 3.

Step 2 For each terminated chain $[\mathbf{v}^1] \to [\mathbf{v}^2] \to \cdots \to [\mathbf{v}^k]$, we converts for $j : 1 \leq j \leq k$ each representation vector $\mathbf{v}^j$ to $\mathbf{w}^j$ so that $[\mathbf{v}^j] = [\mathbf{w}^j]$ and for $j : 1 \leq j \leq k-1$,

$$\mathbf{w}^j_{\dashv j-1} \| (w^j_{j-1} - a_j \bmod n, a_j) = \mathbf{w}^{j+1},$$

for some $a_j \in \{1, ..., n-1\}$. Note that we can do this due to Lemma 2 and the fact that there is only one $\to$ directed into $[\mathbf{v}^{j+1}]$ because a function mapped from it determines the $\to$ directed into it.

Step 3 Recall that $A \cup B$ induces only $(1,k)$-terminated chains for some $\lceil n/2 \rceil \leq k \leq n-1$. Let `ChnLst` be a set of all the last terms of terminated chains. Now we construct Γ by letting

$$\mathtt{DPath}(\Gamma) = \bigsqcup_{[\mathbf{x}] \in \mathtt{ChnLst}} \langle \mathbf{x}_{\dashv(|\mathbf{x}|-1)} \rangle.$$

Step 4 Pick one $(n-1)$-length differential path in $\mathtt{DPath}(\Gamma)$, say $\mathbf{p}$. Increment it to $\mathbf{p} \| (a)$ where $a \notin \triangledown \mathbf{p}$.

Theorem 6 *For Γ above, we have that (N, Γ) is an SKDP with* $\max_{u \in N} |I_u| = \lceil (2^n - 1)/n \rceil$ *and the set of all labels of nodes completes the collection of all non-empty subsets of N.*

An Example. $n = 7$. The diagram of chains induced by $A \cup B$ in step 1 is shown in Figure 3(left). After conversion in step 2 we have a diagram in Figure 3(right). $\mathtt{DPath}(\Gamma)$ defined in step 3 is $\langle (1,1,1,1,1) \rangle \sqcup \langle (3,5,5,6) \rangle \sqcup \langle (2,1,1,2) \rangle \sqcup \langle (3,1,2) \rangle \sqcup \langle (2,2,1) \rangle$. Lastly we pick $(1,2,3,4,5,6) \in \mathtt{DPath}(\Gamma)$ and increment it to be $(1,2,3,4,5,6,7)$. This yields $|I_u| = \lfloor (2^7 - 1)/7 \rfloor = 18$ for $u \neq 7$ and $|I_7| = 19$.

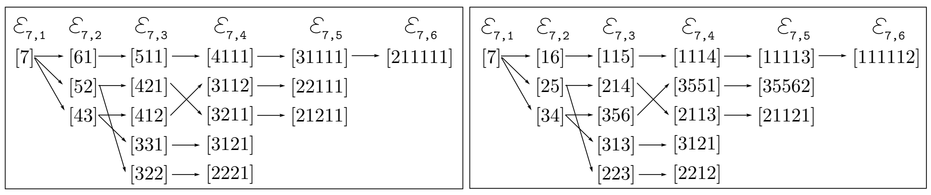
$\mathcal{E}_{7,1}$ $\mathcal{E}_{7,2}$ $\mathcal{E}_{7,3}$ $\mathcal{E}_{7,4}$ $\mathcal{E}_{7,5}$ $\mathcal{E}_{7,6}$
[7] [61] [511] [4111] [31111] [211111]
[52] [421] [3112] [22111]
[43] [412] [3211] [21211]
[331] [3121]
[322] [2221]
$\mathcal{E}_{7,1}$ $\mathcal{E}_{7,2}$ $\mathcal{E}_{7,3}$ $\mathcal{E}_{7,4}$ $\mathcal{E}_{7,5}$ $\mathcal{E}_{7,6}$
[7] [16] [115] [1114] [11113] [111112]
[25] [214] [3551] [35562]
[34] [356] [2113] [21121]
[313] [3121]
[223] [2212]

Fig. 3. Diagram of chains induced by $A \cup B$ in step 1(left) and its conversion after step 2(right). Recall that the direction of $\to$ is opposite to the way functions are mapped.

$$\mathbf{p}_{1,1} = (1,2,3,4,5,6,7)$$

$$\left\{\begin{array}{l}
\mathbf{p}_{1,2} = (2,3,4,5,6,7) \\
\mathbf{p}_{1,3} = (3,4,5,6,7,1) \\
\mathbf{p}_{1,4} = (4,5,6,7,1,2) \\
\mathbf{p}_{1,5} = (5,6,7,1,2,3) \\
\mathbf{p}_{1,6} = (6,7,1,2,3,4) \\
\mathbf{p}_{1,7} = (7,1,2,3,4,5)
\end{array}\right\} = \langle (1,1,1,1,1) \rangle \setminus \{(1,2,3,4,5,6)\}$$

$$\left\{\begin{array}{l}
\mathbf{p}_{2,1} = (1,4,2,7,6) \\
\mathbf{p}_{2,2} = (2,5,3,1,7) \\
\mathbf{p}_{2,3} = (3,6,4,2,1) \\
\mathbf{p}_{2,4} = (4,7,5,3,2) \\
\mathbf{p}_{2,5} = (5,1,6,4,3) \\
\mathbf{p}_{2,6} = (6,2,7,5,4) \\
\mathbf{p}_{2,7} = (7,3,1,6,5)
\end{array}\right\} = \langle (3,5,5,6) \rangle \quad
\left\{\begin{array}{l}
\mathbf{p}_{3,1} = (1,3,4,5,7) \\
\mathbf{p}_{3,2} = (2,4,5,6,1) \\
\mathbf{p}_{3,3} = (3,5,6,7,2) \\
\mathbf{p}_{3,4} = (4,6,7,1,3) \\
\mathbf{p}_{3,5} = (5,7,1,2,4) \\
\mathbf{p}_{3,6} = (6,1,2,3,5) \\
\mathbf{p}_{3,7} = (7,2,3,4,6)
\end{array}\right\} = \langle (2,1,1,2) \rangle$$

$$\left\{\begin{array}{l}
\mathbf{p}_{4,1} = (1,4,5,7) \\
\mathbf{p}_{4,2} = (2,5,6,1) \\
\mathbf{p}_{4,3} = (3,6,7,2) \\
\mathbf{p}_{4,4} = (4,7,1,3) \\
\mathbf{p}_{4,5} = (5,1,2,4) \\
\mathbf{p}_{4,6} = (6,2,3,5) \\
\mathbf{p}_{4,7} = (7,3,4,6)
\end{array}\right\} = \langle (3,1,2) \rangle \quad
\left\{\begin{array}{l}
\mathbf{p}_{5,1} = (1,3,5,6) \\
\mathbf{p}_{5,2} = (2,4,6,7) \\
\mathbf{p}_{5,3} = (3,5,7,1) \\
\mathbf{p}_{5,4} = (4,6,1,2) \\
\mathbf{p}_{5,5} = (5,7,2,3) \\
\mathbf{p}_{5,6} = (6,1,3,4) \\
\mathbf{p}_{5,7} = (7,2,4,5)
\end{array}\right\} = \langle (2,2,1) \rangle$$

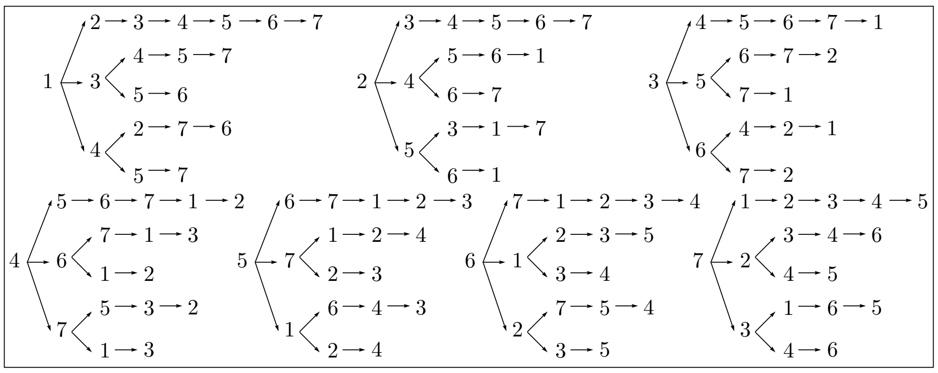

Fig. 4. `DPath`(Γ) (upper part) and its differential path representation (lower part) of SKDP for optimal KPS.

References

1. S. G. Akl and P. D. Taylor, "Cryptographic Solution to a Problem of Access Control in a Hierarchy", ACM Transactions on Computer Systems, Vol. 1, No. 3, pp. 239-248, 1983.
2. J. Anzai, N. Matsuzaki and T. Matsumoto, "Quick Group Key Distibution Scheme with Entity Revocation", Asiacrypt 1999, LNCS 1716, pp. 333-347, 1999.

3. T. Asano, "A Revocation Scheme with Minimal Storage at Receivers", ASIACRYPT 2002, LNCS 2501, pp.433-450.
4. N. Attrapadung, K. Kobara, H. Imai, "Broadcast Encryption with Short Keys and Transmissions", ACM Workshop on Digital Rights Management, October 2003.
5. R. Blom, "An Optimal Class of Symmetric Key Generation Systems", EUROCRYPT 1984, LNCS 209, pp. 335-338.
6. C. Blundo, L. F. Mattos, D. R. Stinson. "Trade-offs Between communication and Storage in Unconditionally Secure Schemes for Broadcast Encryption and Interactive Key Distribution",CRYPTO'96, LNCS 1109 pp. 387-400.
7. D. Boneh and A. Silverberg, "Applications of Multilinear Forms to Cryptography", preprint, 2002. Available from http://eprint.iacr.org.
8. G. C. Chick and S. E. Tavares, "Flexible Access Control with Master Keys", Crypto 1989, LNCS 435, pp. 316-322, 1990.
9. Y. Desmedt, V. Viswanathan,"Unconditionally Secure Dynamic Conference Key Distribution", IEEE, ISIT'98.
10. Y. Dodis and N.Fazio, "Public Key Broadcast Encryption for Stateless Receivers", ACM Workshop on Digital Rights Management, November 2002.
11. Y. Dodis and N. Fazio, "Public Key Broadcast Encryption Secure against Adaptive Chosen Ciphertext Attack", PKC '03, LNCS 2567, 2003.
12. Y. Dodis, N. Fazio, A. Kiayias, M. Yung, "Fully Scalable Public-Key Traitor Tracing", To appear in the proceeding of PODC 2003.
13. A. Fiat, M. Naor, "Broadcast Encryption", CRYPTO 1993, LNCS 0773, pp. 480-491.
14. C. Gentry, A. Silverberg, "Hierarchical ID-Based Cryptography", ASIACRYPT 2002, LNCS 2501, pp.548-566.
15. E. Gafni, J. Staddon, Y.L.Yin, "Efficient Methods for Integrating Traceability and Broadcast Encryption", CRYPTO 1999, LNCS 1666, pp. 372-387.
16. D. Halevi,A. Shamir "The LSD Broadcast Encryption Scheme",CRYPTO 2002, LNCS 2442, pp. 47-60.
17. R. Kumar, S. Rajagopalan, A. Sahai, "Coding Constructions for Blacklisting Problems without Computational Assumptions", CRYPTO 1999, LNCS 1666, pp. 609-623.
18. K. Kurosawa, T. Yoshida, Y. Desmedt, M. Burmester, "Some Bounds and a Construction for Secure Broadcast Encryption", ASIACRYPT'98, LNCS 1514, pp. 420-433.
19. M. Luby, J. Staddon, "Combinatorial Bounds for Broadcast Encryption", EUROCRYPT 1998, LNCS 1403, pp. 512-526.
20. T. Matsumoto, H. Imai,"On the Key Predistribution System: A Practical Solution to the Key Distribution Problem", CRYPTO'87, 185-193.
21. D. Naor, M. Naor and J. Lotspiech,"Revocation and Tracing Schemes for Stateless Receivers",CRYPTO 2001, LNCS 2139, 41-62.
22. M. Naor and B. Pinkas, "Effcient Trace and Revoke Schemes", Financial Cryptography FC 2000, LNCS1962, pp.1-20.
23. D.R.Stinson,"On Some Methods for Unconditionally Secure Key Distribution and Broadcast Encryption", Designs, Codes and Cryptography 12 (1997), 215-243.
24. D. Wallner, E. Harder and R. Agee, "Key Management for Multicast: Issues and Architectures", IETF NetworkWorking Group, Request for Comments: 2627, available from ftp://ftp.ietf.org/rfc/rfc2627.txt, 1999.
25. C. K. Wong, M. Gouda and S. S. Lam, "Secure Group Communications Using Key Graphs", ACM SIGCOMM'98, 1998.