“Tree-Based Symmetric Key Broadcast Encryption”

(Thesis Defense)

Sanjay Bhattacherjee

ISI, Kolkata; ENS-Lyon

Supervisor: Prof. Palash Sarkar

ISI, Kolkata

Date: 8^{th} October, 2015

ENS DE LYON

Outline

Symmetric Key Encryption

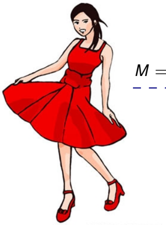
Alice

Bob

Symmetric Key Encryption

Alice

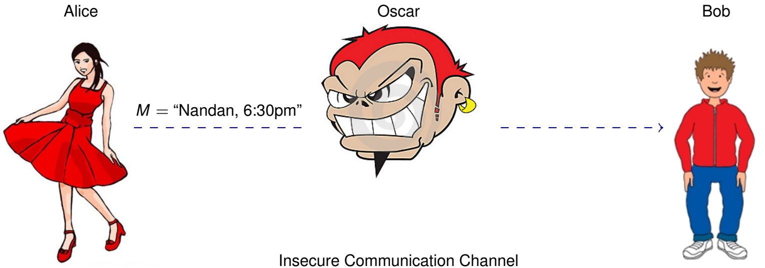
Bob



M = “Nandan, 6:30pm”

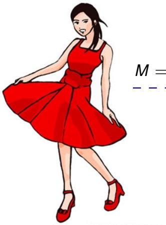
Insecure Communication Channel

Symmetric Key Encryption



Symmetric Key Encryption

Alice



Oscar

Bob

M = "Nandan, 6:30pm"

Insecure Communication Channel

K $\quad$ Enc $\qquad\qquad$ Dec $\quad$ K

Symmetric Key Encryption

Alice

Oscar

Bob

M = "Nandan, 6:30pm"

Insecure Communication Channel

M = "Nandan, 6:30pm"

K

Enc

Dec

K

Symmetric Key Encryption

Alice

Oscar

Bob

M = “Nandan, 6:30pm”

Insecure Communication Channel

M = “Nandan, 6:30pm”

$K \dashrightarrow$ $C \leftarrow E(M, K)$

K

Enc

Dec

Symmetric Key Encryption

Alice

Oscar

Bob

M = “Nandan, 6:30pm”

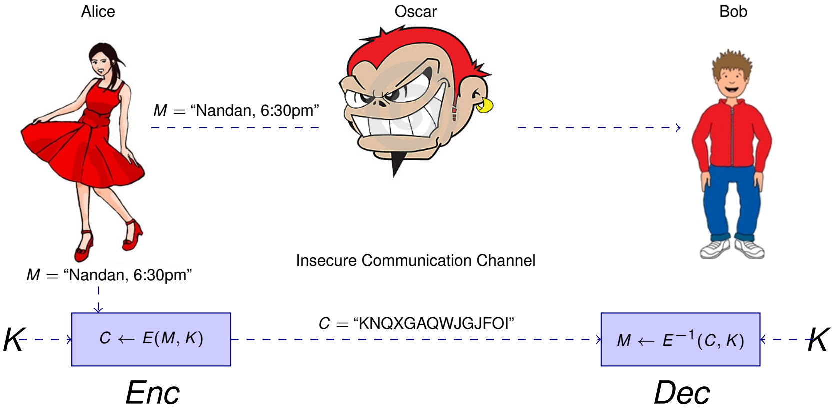
Insecure Communication Channel

M = “Nandan, 6:30pm”

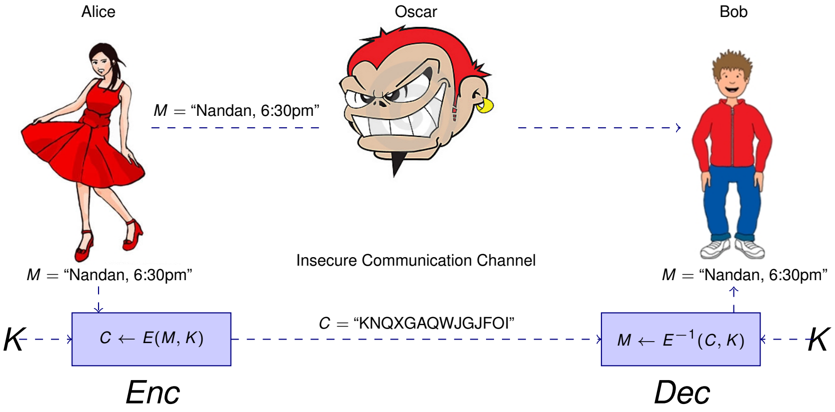
K $\quad$ $C \leftarrow E(M, K)$ $\quad$ C = “KNQXGAQWJGJFOI” $\quad$ K

Enc $\quad$ *Dec*

Symmetric Key Encryption

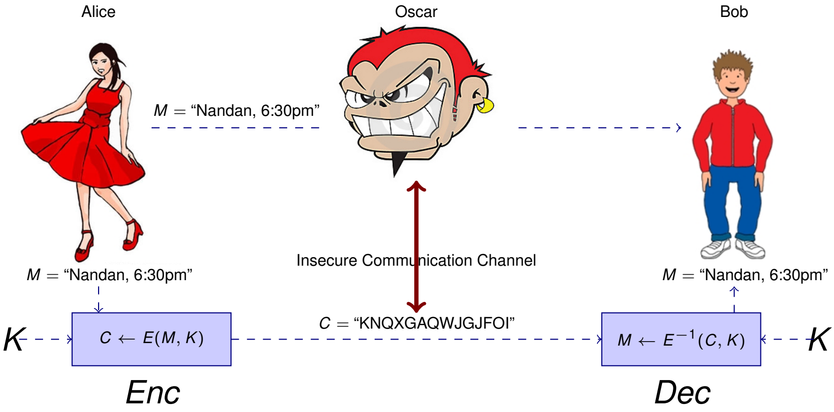


Symmetric Key Encryption



ENS DE LYON

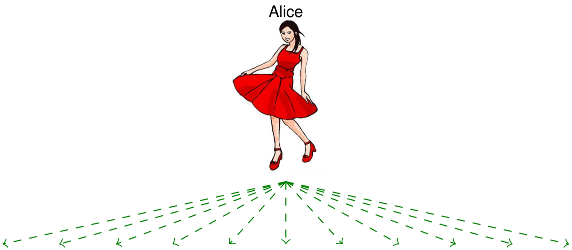
Symmetric Key Encryption



One Alice → Many Bobs?

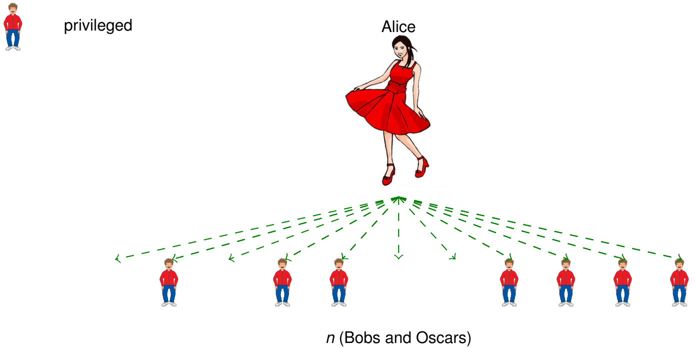
Alice

One Alice $\rightarrow$ Many Bobs?

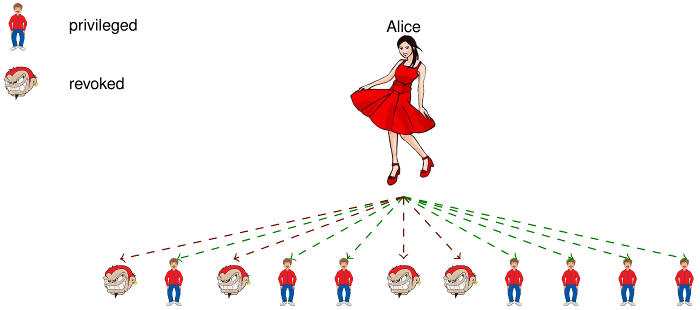


n (Bobs and Oscars)

One Alice → Many Bobs?



One Alice → Many Bobs?



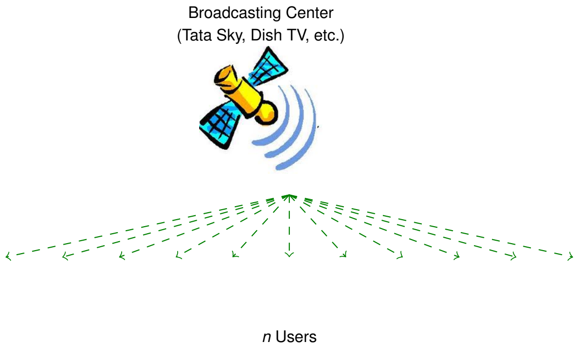
n (Bobs and Oscars)

Pay-TV center $\rightarrow$ Subscribers

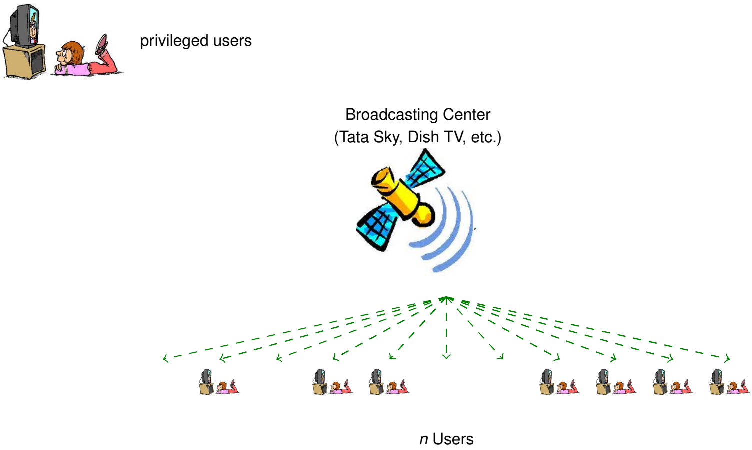
Broadcasting Center
(Tata Sky, Dish TV, etc.)

ENS DE LYON

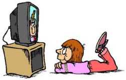
Pay-TV center $\rightarrow$ Subscribers



Pay-TV center → Subscribers



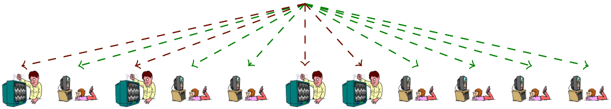
Pay-TV center → Subscribers



privileged users

revoked users

Broadcasting Center
(Tata Sky, Dish TV, etc.)



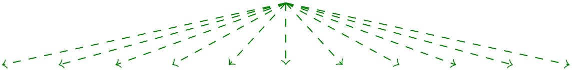
n Users

DRM in Blu-ray/DVD discs

(Copyrighted) Content Production House

DRM in Blu-ray/DVD discs

(Copyrighted) Content Production House

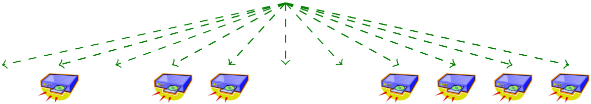


n Users

DRM in Blu-ray/DVD discs

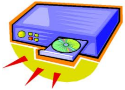
legitimate player

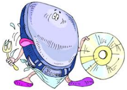
(Copyrighted) Content Production House

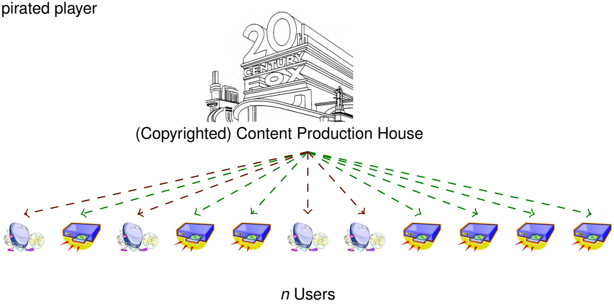


n Users

DRM in Blu-ray/DVD discs

 legitimate player

 pirated player



Basic Schemes

$\mathcal{N}$: set of all users $u_1, \ldots, u_n$; $\quad n = |\mathcal{N}|$

$\mathcal{R}$: set of revoked users; $\quad r = |\mathcal{R}|$

Basic Schemes

$\mathcal{N}$: set of all users $u_1, \ldots, u_n$; $\quad n = |\mathcal{N}|$
$\mathcal{R}$: set of revoked users; $\quad r = |\mathcal{R}|$

$$\mathcal{S} = \{S_i : S_i \subseteq \mathcal{N}\}$$

STATISTICAL INDIAN INSTITUTE
UNITY IN DIVERSITY

ENS DE LYON

Basic Schemes

$\mathcal{N}$: set of all users $u_1, \ldots, u_n$; $\quad n = |\mathcal{N}|$
$\mathcal{R}$: set of revoked users; $\quad r = |\mathcal{R}|$

$$\mathcal{S} = \{S_i : S_i \subseteq \mathcal{N}\}$$

Singleton Set Scheme

$\mathcal{S} = \{\{u_1\}, \ldots, \{u_n\}\}$

- Each user is assigned a unique key. $\quad O(1)$
- M has to be encrypted for each user in $\mathcal{N} \setminus \mathcal{R}$. $\quad O(n - r)$

Basic Schemes

$\mathcal{N}$: set of all users $u_1, \ldots, u_n$; $\quad n = |\mathcal{N}|$
$\mathcal{R}$: set of revoked users; $\quad r = |\mathcal{R}|$

$$\mathcal{S} = \{S_i : S_i \subseteq \mathcal{N}\}$$

Singleton Set Scheme

$\mathcal{S} = \{\{u_1\}, \ldots, \{u_n\}\}$

- Each user is assigned a unique key. $O(1)$
- M has to be encrypted for each user in $\mathcal{N} \setminus \mathcal{R}$. $O(n - r)$

Power Set Scheme

$\mathcal{S} = \{\{u_1\}, \ldots, \{u_1, u_2\}, \ldots, \{u_1, \ldots, u_{n-1}\}, \ldots, \mathcal{N}\}$

- Each subset of users is assigned a unique key. $O(2^n)$
- M is encrypted only once for the set $\mathcal{N} \setminus \mathcal{R} \in \mathcal{S}$. $O(1)$

Subset Cover Framework [NNL01]

1. Initiation

- Choose the collection

$$\mathcal{S} = \{S_1, \ldots, S_w\}; S_i \subseteq \mathcal{N}.$$

Subset Cover Framework [NNL01]

1. Initiation

- Choose the collection

$$\mathcal{S} = \{S_1, \ldots, S_w\}; S_i \subseteq \mathcal{N}.$$

- Assign key L_i to each $S_i \in \mathcal{S}$
  - Only $u \in S_i$ gets L_i

Subset Cover Framework [NNL01]

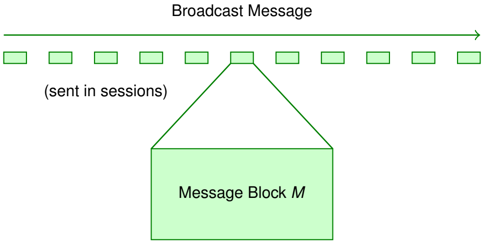
1. Initiation

- Choose the collection

$$\mathcal{S} = \{S_1, \ldots, S_w\}; S_i \subseteq \mathcal{N}.$$

- Assign key L_i to each $S_i \in \mathcal{S}$
  - Only $u \in S_i$ gets L_i

2. Encryption



Subset Cover Framework [NNL01]

2. Encryption $(M, \mathcal{R})$

For each session (with privileged users $\mathcal{N} \setminus \mathcal{R}$):

Subset Cover Framework [NNL01]

2. Encryption (M, $\mathcal{R}$)

For each session (with privileged users $\mathcal{N} \setminus \mathcal{R}$):

- Find the Subset Cover $\mathcal{S}_c = \{S_{i_1}, \ldots, S_{i_h}\}$

Subset Cover Framework [NNL01]

2. Encryption $(M, \mathcal{R})$

For each session (with privileged users $\mathcal{N} \setminus \mathcal{R}$):

- Find the Subset Cover $\mathcal{S}_c = \{S_{i_1}, \ldots, S_{i_h}\} \subset \mathcal{S}$ such that

$$\mathcal{N} \setminus \mathcal{R} = S_{i_1} \cup \cdots \cup S_{i_h}$$

- Encrypt:

Subset Cover Framework [NNL01]

2. Encryption $(M, \mathcal{R})$

For each session (with privileged users $\mathcal{N} \setminus \mathcal{R}$):

- Find the Subset Cover $\mathcal{S}_c = \{S_{i_1}, \ldots, S_{i_h}\} \subset \mathcal{S}$ such that

$$\mathcal{N} \setminus \mathcal{R} = S_{i_1} \cup \cdots \cup S_{i_h}$$

- Encrypt:
  - M with random K_s;

Subset Cover Framework [NNL01]

2. Encryption $(M, \mathcal{R})$

For each session (with privileged users $\mathcal{N} \setminus \mathcal{R}$):

- Find the Subset Cover $\mathcal{S}_c = \{S_{i_1}, \ldots, S_{i_h}\} \subset \mathcal{S}$ such that

$$\mathcal{N} \setminus \mathcal{R} = S_{i_1} \cup \cdots \cup S_{i_h}$$

- Encrypt:
  - M with random K_s;
  - K_s with L_{i_j} of each $S_{i_j} \in \mathcal{S}_c$

Subset Cover Framework [NNL01]

2. Encryption $(M, \mathcal{R})$

For each session (with privileged users $\mathcal{N} \setminus \mathcal{R}$):

- Find the Subset Cover $\mathcal{S}_c = \{S_{i_1}, \ldots, S_{i_h}\} \subset \mathcal{S}$ such that

$$\mathcal{N} \setminus \mathcal{R} = S_{i_1} \cup \cdots \cup S_{i_h}$$

- Encrypt:
  - M with random K_s;
  - K_s with L_{i_j} of each $S_{i_j} \in \mathcal{S}_c$

$$\boxed{F_{K_s}(M)}$$

ENS DE LYON

Subset Cover Framework [NNL01]

2. Encryption $(M, \mathcal{R})$

For each session (with privileged users $\mathcal{N} \setminus \mathcal{R}$):

- Find the Subset Cover $\mathcal{S}_c = \{S_{i_1}, \dots, S_{i_h}\} \subset \mathcal{S}$ such that

$$\mathcal{N} \setminus \mathcal{R} = S_{i_1} \cup \cdots \cup S_{i_h}$$

- Encrypt:
  - M with random K_s;
  - K_s with L_{i_j} of each $S_{i_j} \in \mathcal{S}_c$

$F_{K_s}(M)$	$E_{L_{i_1}}(K_s)$	$\dots$	$E_{L_{i_h}}(K_s)$

Subset Cover Framework [NNL01]

2. Encryption $(M, \mathcal{R})$

For each session (with privileged users $\mathcal{N} \setminus \mathcal{R}$):

- Find the Subset Cover $\mathcal{S}_c = \{S_{i_1}, \ldots, S_{i_h}\} \subset \mathcal{S}$ such that

$$\mathcal{N} \setminus \mathcal{R} = S_{i_1} \cup \cdots \cup S_{i_h}$$

- Encrypt:
  - M with random K_s;
  - K_s with L_{i_j} of each $S_{i_j} \in \mathcal{S}_c$

$F_{K_s}(M)$	$E_{L_{i_1}}(K_s)$	$\cdots$	$E_{L_{i_h}}(K_s)$
body	header		

Subset Cover Framework [NNL01]

3. Decryption

$F_{K_s}(M)$	$E_{L_{i_1}}(K_s)$	$\cdots$	$E_{L_{i_h}}(K_s)$

For $u \in S_{i_j}$ where $S_{i,j} \in \mathcal{S}_c$

- Find $E_{L_{i_j}}(K_s)$ in the header

Subset Cover Framework [NNL01]

3. Decryption

$F_{K_s}(M)$	$E_{L_{i_1}}(K_s)$	$\cdots$	$E_{L_{i_h}}(K_s)$

For $u \in S_{i_j}$ where $S_{i,j} \in \mathcal{S}_c$

- Find $E_{L_{i_j}}(K_s)$ in the header
- $K_s \leftarrow E^{-1}_{L_{i_j}}(E_{L_{i_j}}(K_s))$

UNITY IN DIVERSITY

ENS DE LYON

Subset Cover Framework [NNL01]

3. Decryption

$F_{K_s}(M)$	$E_{L_{i_1}}(K_s)$	$\cdots$	$E_{L_{i_h}}(K_s)$

For $u \in S_{i_j}$ where $S_{i,j} \in \mathcal{S}_c$

- Find $E_{L_{i_j}}(K_s)$ in the header
- $K_s \leftarrow E^{-1}_{L_{i_j}}(E_{L_{i_j}}(K_s))$
- $M \leftarrow F^{-1}_{K_s}(F_{K_s}(M))$

Parameters of Interest

- $|\mathcal{S}_c| = h$: header length (costliest parameter)

  Example: Pay-TV bandwidth cost

Parameters of Interest

- $|\mathcal{S}_c| = h$: header length (costliest parameter)

  Example: Pay-TV bandwidth cost

- $|I_u|$: user storage (may be costly)

  Example: High-end military receivers

Parameters of Interest

- $|\mathcal{S}_c| = h$: header length (costliest parameter)

  Example: Pay-TV bandwidth cost

- $|I_u|$: user storage (may be costly)

  Example: High-end military receivers

- Encryption time
- Decryption time

  Example: TV set-top box booting time

Applications of BE

- **Pay-TV**, CableLabs standard.
- **AACS:** Disney, Intel, Microsoft, Panasonic, Warner Bros., IBM, Toshiba and Sony.

Blu-ray Disc Manufacturer

Player Manufacturer

Applications of BE

- **Pay-TV**, CableLabs standard.
- **AACS:** Disney, Intel, Microsoft, Panasonic, Warner Bros., IBM, Toshiba and Sony.

Blu-ray Disc Manufacturer

Player Manufacturer

- Military Broadcasts
  - Global Broadcast Service (US)
  - Joint Broadcast System (Europe)

Applications of BE

- **Pay-TV**, CableLabs standard.
- **AACS:** Disney, Intel, Microsoft, Panasonic, Warner Bros., IBM, Toshiba and Sony.

Blu-ray Disc Manufacturer

Player Manufacturer

- Military Broadcasts
  - Global Broadcast Service (US)
  - Joint Broadcast System (Europe)
- File Sharing in Encrypted File Systems.
- Mailing list encryption: [BGW05] OpenPGP functions as a BE system
- Online content sharing and distribution [BBW06]
- eCommerce: trade secret broadcasts
- ...

Why NOT use Public-Key BE?

Efficiency!!!

(decryption time, hence cost)

The collection $\mathcal{S}$

$$\mathcal{S} = \{S_1, \ldots, S_w\}; S_i \subseteq \mathcal{N}$$

The collection $\mathcal{S}$

$$\mathcal{S} = \{S_1, \ldots, S_w\}; S_i \subseteq \mathcal{N}$$

- determines the header length h
  (through the cover generation algorithm)
  Subset Cover $\mathcal{S}_c = \{S_{i_1}, \ldots, S_{i_h}\} \subset \mathcal{S}$ such that

$$\mathcal{N} \setminus \mathcal{R} = S_{i_1} \cup \cdots \cup S_{i_h}$$

The collection $\mathcal{S}$

$$\mathcal{S} = \{S_1, \ldots, S_w\}; S_i \subseteq \mathcal{N}$$

- determines the header length h
  (through the cover generation algorithm)
  Subset Cover $\mathcal{S}_c = \{S_{i_1}, \ldots, S_{i_h}\} \subset \mathcal{S}$ such that

$$\mathcal{N} \setminus \mathcal{R} = S_{i_1} \cup \cdots \cup S_{i_h}$$

- determines the user storage $|I_u|$
  (through the key assignment and distribution algorithm)

The collection $\mathcal{S}$

$$\mathcal{S} = \{S_1, \ldots, S_w\}; S_i \subseteq \mathcal{N}$$

- determines the header length h
  (through the cover generation algorithm)
  Subset Cover $\mathcal{S}_c = \{S_{i_1}, \ldots, S_{i_h}\} \subset \mathcal{S}$ such that

$$\mathcal{N} \setminus \mathcal{R} = S_{i_1} \cup \cdots \cup S_{i_h}$$

- determines the user storage $|I_u|$
  (through the key assignment and distribution algorithm)
- determines the encryption and decryption time
  (through the key assignment and distribution algorithm)

Two types of $\mathcal{S}$

- Subset Difference $\{1\}, \{3\}, \{6, 7, 8\}$
- Punctured Interval $\{1, 3, 6\}, \{7, 8\}$

Dalit Naor, Moni Naor, and Jeffery Lotspiech.
Revocation and tracing schemes for stateless receivers.
In Joe Kilian, editor, *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 41–62. Springer, 2001.

Nam-Su Jho and Jung Yeon Hwang and Jung Hee Cheon and Myung-Hwan Kim and Dong Hoon Lee and Eun Sun Yoo.
One-Way Chain Based Broadcast Encryption Schemes.
In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 559–574. Springer, 2005.

Outline

Preliminaries

Background

NNL-SD: Initiation

Define $\mathcal{S}_{NNL-SD}$

Key Assignment

Key Distribution

NNL-SD: Encryption

Halevy-Shamir Layered SD

Other Related Works

Our Contributions

Paper 1: Arbitrary n; Detailed Analysis

Paper 2: Layering; Minimizing Storage

Paper 3: k-ary Generalization

Paper 4: Assured Savings on Communication

Conclusion

ENS DE LYON

Subset Difference (SD) Scheme [NNL01]

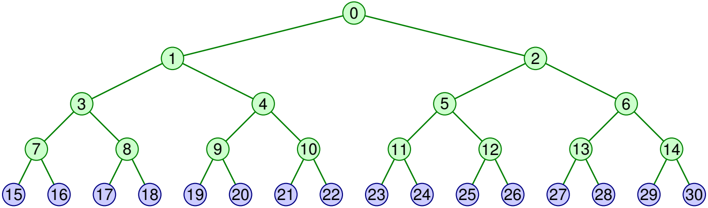
Naor-Naor-Lotspiech (2001)

- Patented
- Used in the AACS standard

Subset Difference (SD) Scheme [NNL01]

Naor-Naor-Lotspiech (2001)

- Patented
- Used in the AACS standard



Assumed $n = 2^{\ell_0}$

Collection $\mathcal{S}_{NNL-SD}$ has:

Collection $\mathcal{S}_{NNL-SD}$ has:

- For all internal nodes i

Collection $\mathcal{S}_{NNL-SD}$ has:

- For all internal nodes i
- For all corresponding nodes $j(\neq i)$ in the subtree $\mathcal{T}_i$

Collection $\mathcal{S}_{NNL-SD}$ has:

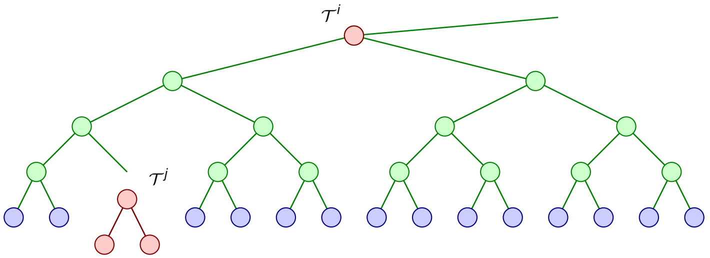
- For all internal nodes i
- For all corresponding nodes $j(\neq i)$ in the subtree $\mathcal{T}_i$

$$S_{i,j} = \mathcal{T}_i \setminus \mathcal{T}_j$$

Collection $\mathcal{S}_{NNL-SD}$ has:

- For all internal nodes i
- For all corresponding nodes $j(\neq i)$ in the subtree $\mathcal{T}_i$

$$S_{i,j} = \mathcal{T}_i \setminus \mathcal{T}_j$$



All users that are in $\mathcal{T}_i$ but not in $\mathcal{T}_j$

Key Assignment

Key for $S_{i,j}$:

Key Assignment

Key for $S_{i,j}$:

- Assign random $seed_i$ to each internal node i
- Pseudo-random generator (PRG): $G : \{0,1\}^k \to \{0,1\}^{3k}$

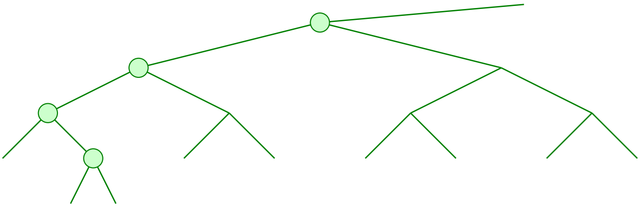
$$G(seed) = G_L(seed) || G_M(seed) || G_R(seed)$$

Key Assignment

Key for $S_{i,j}$:

- Assign random $seed_i$ to each internal node i
- Pseudo-random generator (PRG): $G : \{0,1\}^k \to \{0,1\}^{3k}$

$$G(seed) = G_L(seed) || G_M(seed) || G_R(seed)$$

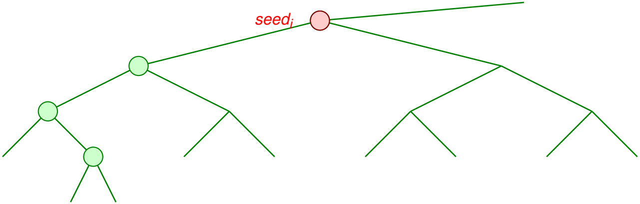


Key Assignment

Key for $S_{i,j}$:

- Assign random $seed_i$ to each internal node i
- Pseudo-random generator (PRG): $G : \{0,1\}^k \to \{0,1\}^{3k}$

$$G(seed) = G_L(seed) || G_M(seed) || G_R(seed)$$

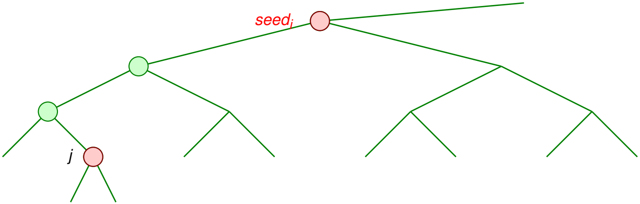


Key Assignment

Key for $S_{i,j}$:

- Assign random $\textit{seed}_i$ to each internal node i
- Pseudo-random generator (PRG): $G : \{0,1\}^k \rightarrow \{0,1\}^{3k}$

$$G(seed) = G_L(seed)||G_M(seed)||G_R(seed)$$

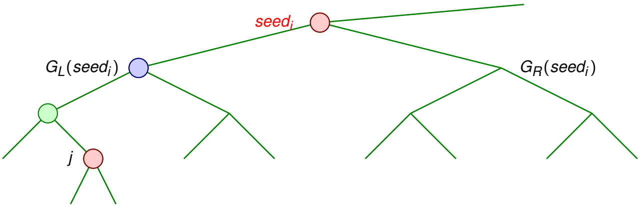


Key Assignment

Key for $S_{i,j}$:

- Assign random $seed_i$ to each internal node i
- Pseudo-random generator (PRG): $G : \{0,1\}^k \rightarrow \{0,1\}^{3k}$

$$G(seed) = G_L(seed) || G_M(seed) || G_R(seed)$$

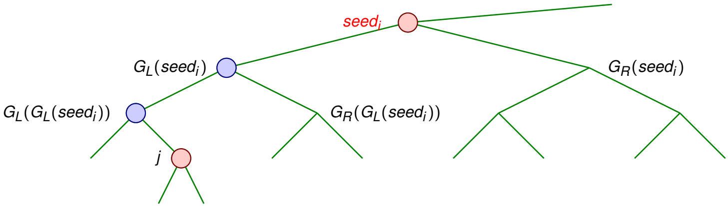


Key Assignment

Key for $S_{i,j}$:

- Assign random $seed_i$ to each internal node i
- Pseudo-random generator (PRG): $G : \{0,1\}^k \to \{0,1\}^{3k}$

$$G(seed) = G_L(seed)||G_M(seed)||G_R(seed)$$

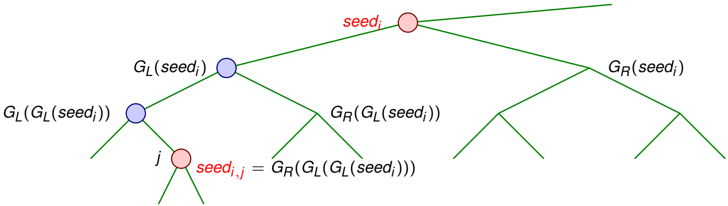


Key Assignment

Key for $S_{i,j}$:

- Assign random $seed_i$ to each internal node i
- Pseudo-random generator (PRG): $G : \{0,1\}^k \rightarrow \{0,1\}^{3k}$

$$G(seed) = G_L(seed) || G_M(seed) || G_R(seed)$$

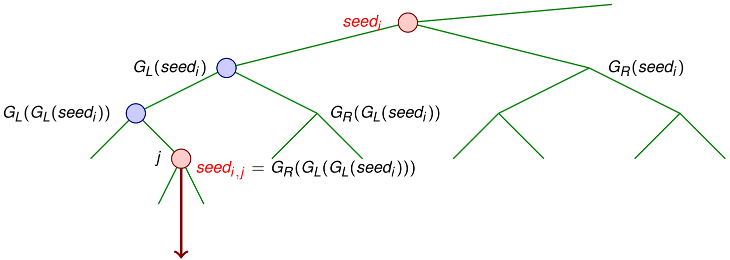


Key Assignment

Key for $S_{i,j}$:

- Assign random $seed_i$ to each internal node i
- Pseudo-random generator (PRG): $G : \{0,1\}^k \rightarrow \{0,1\}^{3k}$

$$G(seed) = G_L(seed)||G_M(seed)||G_R(seed)$$

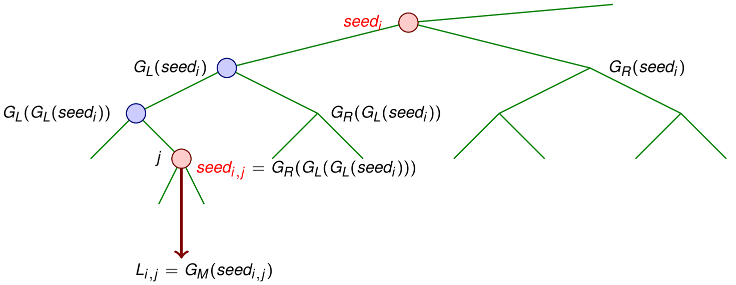


Key Assignment

Key for $S_{i,j}$:

- Assign random $seed_i$ to each internal node i
- Pseudo-random generator (PRG): $G : \{0,1\}^k \to \{0,1\}^{3k}$

$$G(seed) = G_L(seed) || G_M(seed) || G_R(seed)$$

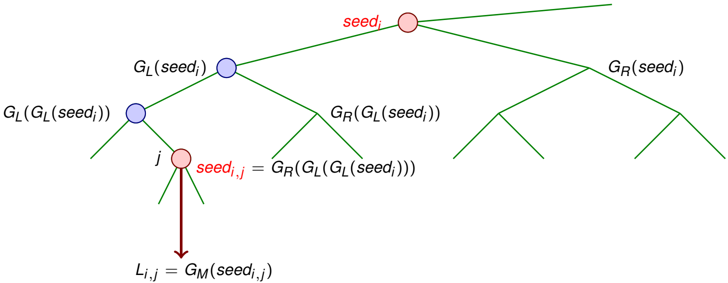


Key Assignment

Key for $S_{i,j}$:

- Assign random $\color{red}{seed_i}$ to each internal node i
- Pseudo-random generator (PRG): $\color{red}{G : \{0,1\}^k \rightarrow \{0,1\}^{3k}}$

$$\color{blue}{G(seed) = G_L(seed) || G_M(seed) || G_R(seed)}$$



Key of $S_{i,j}$: $\color{red}{L_{i,j}} = G_M(seed_{i,j})$

User Storage

User u stores: for every ancestor i (at level ℓ), the derived seeds of nodes "falling-off" from the path between i and u, derived from $seed_i$.

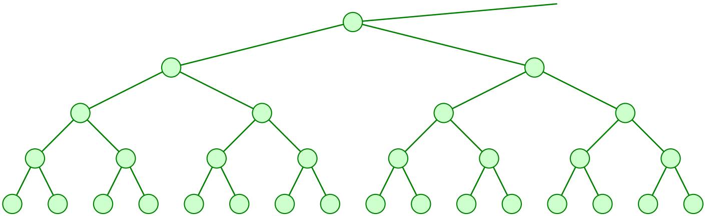


Figure: Secrets stored by u

User Storage

User u stores: for every ancestor i (at level ℓ), the derived seeds of nodes “falling-off” from the path between i and u, derived from $seed_i$.

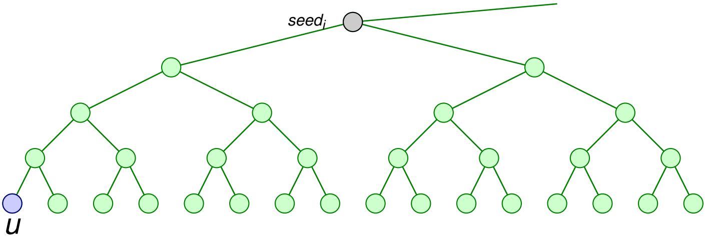


Figure: Secrets stored by u

User Storage

User u stores: for every ancestor i (at level ℓ), the derived seeds of nodes "falling-off" from the path between i and u, derived from $seed_i$.

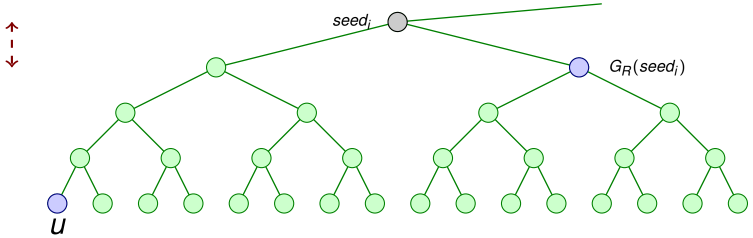


Figure: Secrets stored by u

User Storage

User u stores: for every ancestor i (at level ℓ), the derived seeds of nodes "falling-off" from the path between i and u, derived from $seed_i$.

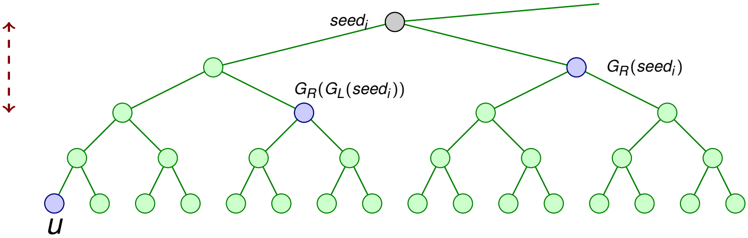


Figure: Secrets stored by u

User Storage

User u stores: for every ancestor i (at level ℓ), the derived seeds of nodes “falling-off” from the path between i and u, derived from $seed_i$.

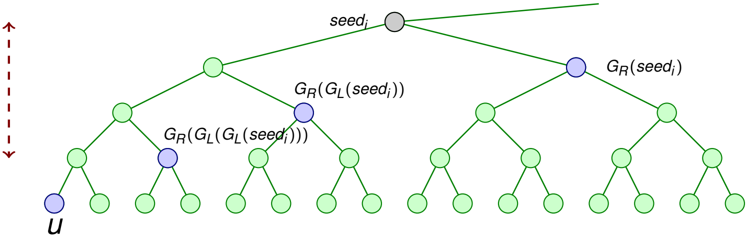


Figure: Secrets stored by u

User Storage

User u stores: for every ancestor i (at level ℓ), the derived seeds of nodes "falling-off" from the path between i and u, derived from $seed_i$.

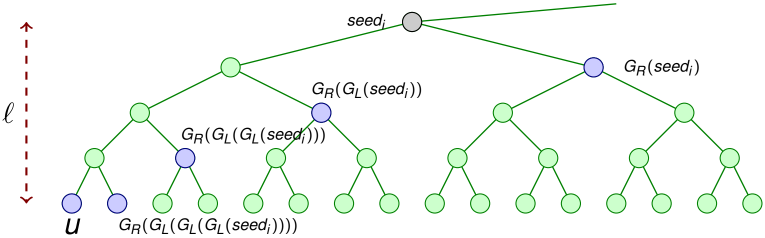
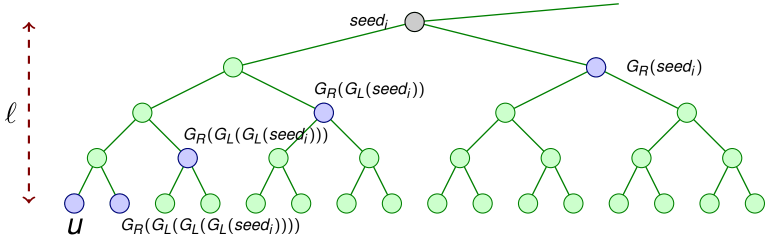


Figure: Secrets stored by u

User Storage

User u stores: for every ancestor i (at level ℓ), the derived seeds of nodes "falling-off" from the path between i and u, derived from $seed_i$.

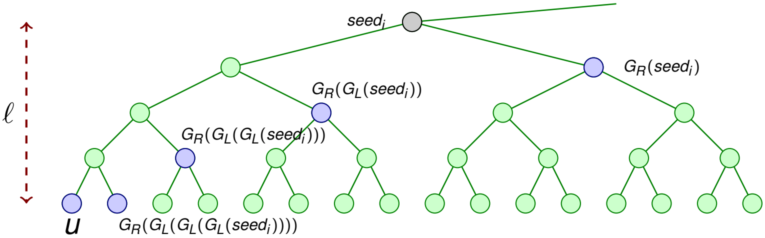


$1 +$

Figure: Secrets stored by u

User Storage

User u stores: for every ancestor i (at level ℓ), the derived seeds of nodes "falling-off" from the path between i and u, derived from $seed_i$.

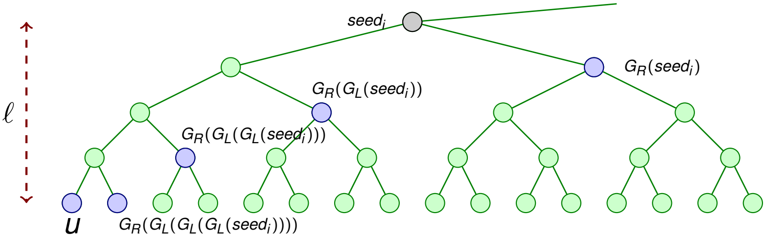


$$1 + 2 +$$

Figure: Secrets stored by u

User Storage

User u stores: for every ancestor i (at level ℓ), the derived seeds of nodes "falling-off" from the path between i and u, derived from $seed_i$.

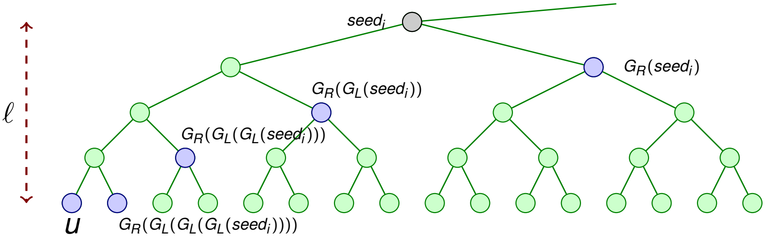


$$1 + 2 + \cdots + \ell_0 =$$

Figure: Secrets stored by u

User Storage

User u stores: for every ancestor i (at level ℓ), the derived seeds of nodes "falling-off" from the path between i and u, derived from $seed_i$.



$$1 + 2 + \cdots + \ell_0 = \frac{\ell_0(\ell_0+1)}{2}$$

Figure: Secrets stored by u

Outline

Subset Cover Finding Algorithm

Given $\mathcal{R}$, find

$$\mathcal{S}_c = \{S_{i_1,j_1}, \ldots, S_{i_h,j_h}\}$$

Subset Cover Finding Algorithm

Given $\mathcal{R}$, find

$$\mathcal{S}_c = \{S_{i_1,j_1}, \ldots, S_{i_h,j_h}\}$$

such that

$$S_{i_1,j_1} \cup \ldots \cup S_{i_h,j_h} = \mathcal{N} \setminus \mathcal{R}$$

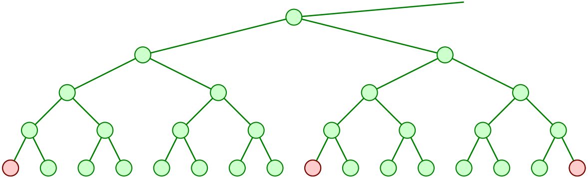
Subset Cover Finding Algorithm

Given $\mathcal{R}$, find

$$\mathcal{S}_c = \{S_{i_1,j_1}, \ldots, S_{i_h,j_h}\}$$

such that

$$S_{i_1,j_1} \cup \ldots \cup S_{i_h,j_h} = \mathcal{N} \setminus \mathcal{R}$$



$$\mathcal{S}_c = \{$$

ENS DE LYON

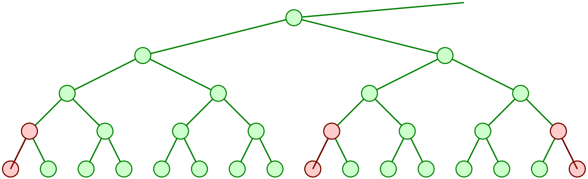
Subset Cover Finding Algorithm

Given $\mathcal{R}$, find

$$\mathcal{S}_c = \{S_{i_1,j_1}, \ldots, S_{i_h,j_h}\}$$

such that

$$S_{i_1,j_1} \cup \ldots \cup S_{i_h,j_h} = \mathcal{N} \setminus \mathcal{R}$$



$$\mathcal{S}_c = \{$$

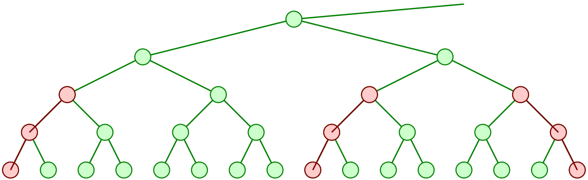
Subset Cover Finding Algorithm

Given $\mathcal{R}$, find

$$\mathcal{S}_c = \{S_{i_1,j_1}, \ldots, S_{i_h,j_h}\}$$

such that

$$S_{i_1,j_1} \cup \ldots \cup S_{i_h,j_h} = \mathcal{N} \setminus \mathcal{R}$$



$$\mathcal{S}_c = \{$$

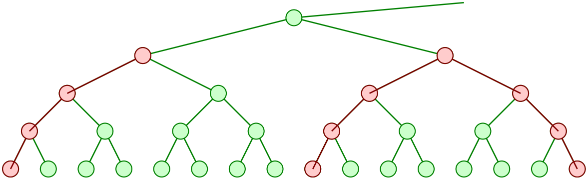
Subset Cover Finding Algorithm

Given $\mathcal{R}$, find

$$\mathcal{S}_c = \{S_{i_1,j_1}, \ldots, S_{i_h,j_h}\}$$

such that

$$S_{i_1,j_1} \cup \ldots \cup S_{i_h,j_h} = \mathcal{N} \setminus \mathcal{R}$$



$$\mathcal{S}_c = \{$$

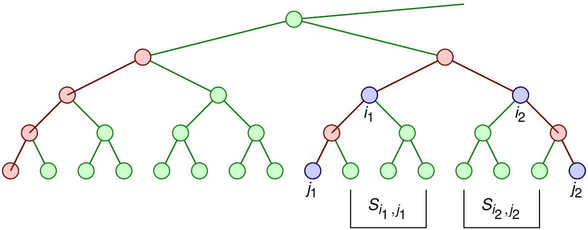
Subset Cover Finding Algorithm

Given $\mathcal{R}$, find

$$\mathcal{S}_c = \{S_{i_1,j_1}, \ldots, S_{i_h,j_h}\}$$

such that

$$S_{i_1,j_1} \cup \ldots \cup S_{i_h,j_h} = \mathcal{N} \setminus \mathcal{R}$$



$$\mathcal{S}_c = \{$$

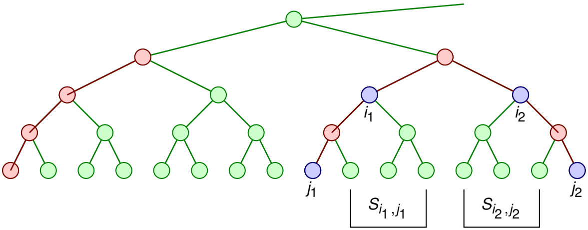
Subset Cover Finding Algorithm

Given $\mathcal{R}$, find

$$\mathcal{S}_c = \{S_{i_1,j_1}, \ldots, S_{i_h,j_h}\}$$

such that

$$S_{i_1,j_1} \cup \ldots \cup S_{i_h,j_h} = \mathcal{N} \setminus \mathcal{R}$$



$$\mathcal{S}_c = \{S_{i_1,j_1}, S_{i_2,j_2},$$

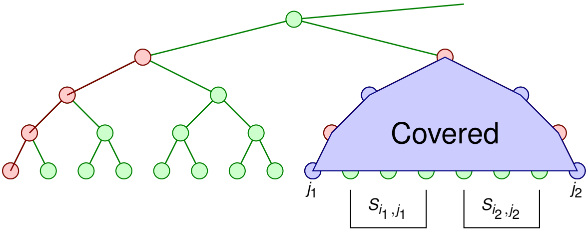
Subset Cover Finding Algorithm

Given $\mathcal{R}$, find

$$\mathcal{S}_c = \{S_{i_1,j_1}, \ldots, S_{i_h,j_h}\}$$

such that

$$S_{i_1,j_1} \cup \ldots \cup S_{i_h,j_h} = \mathcal{N} \setminus \mathcal{R}$$



$$\mathcal{S}_c = \{S_{i_1,j_1}, S_{i_2,j_2},$$

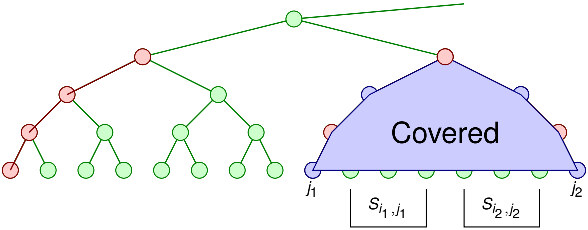
Subset Cover Finding Algorithm

Given $\mathcal{R}$, find

$$\mathcal{S}_c = \{S_{i_1,j_1}, \ldots, S_{i_h,j_h}\}$$

such that

$$S_{i_1,j_1} \cup \ldots \cup S_{i_h,j_h} = \mathcal{N} \setminus \mathcal{R}$$



$$\mathcal{S}_c = \{S_{i_1,j_1}, S_{i_2,j_2},$$

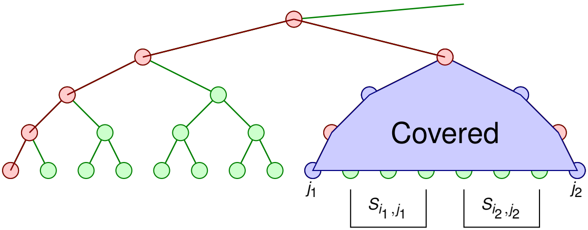
Subset Cover Finding Algorithm

Given $\mathcal{R}$, find

$$\mathcal{S}_c = \{S_{i_1,j_1}, \ldots, S_{i_h,j_h}\}$$

such that

$$S_{i_1,j_1} \cup \ldots \cup S_{i_h,j_h} = \mathcal{N} \setminus \mathcal{R}$$



$$\mathcal{S}_c = \{S_{i_1,j_1}, S_{i_2,j_2},$$

STATISTICAL INDIAN INSTITUTE
UNITY IN DIVERSITY

ENS DE LYON

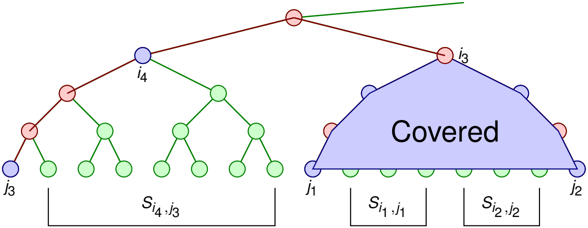
Subset Cover Finding Algorithm

Given $\mathcal{R}$, find

$$\mathcal{S}_C = \{S_{i_1,j_1}, \ldots, S_{i_h,j_h}\}$$

such that

$$S_{i_1,j_1} \cup \ldots \cup S_{i_h,j_h} = \mathcal{N} \setminus \mathcal{R}$$



$$\mathcal{S}_C = \{S_{i_1,j_1}, S_{i_2,j_2},$$

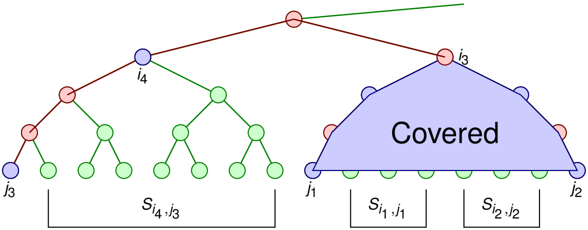
Subset Cover Finding Algorithm

Given $\mathcal{R}$, find

$$\mathcal{S}_c = \{S_{i_1,j_1}, \ldots, S_{i_h,j_h}\}$$

such that

$$S_{i_1,j_1} \cup \ldots \cup S_{i_h,j_h} = \mathcal{N} \setminus \mathcal{R}$$



$$\mathcal{S}_c = \{S_{i_1,j_1}, S_{i_2,j_2}, S_{i_4,j_3},$$

ENS DE LYON

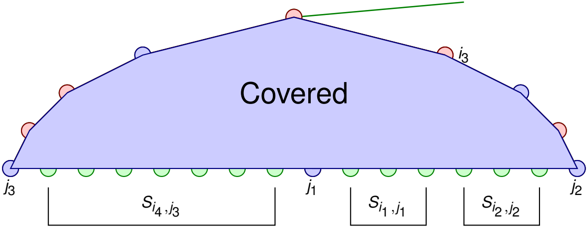
Subset Cover Finding Algorithm

Given $\mathcal{R}$, find

$$\mathcal{S}_c = \{S_{i_1,j_1}, \ldots, S_{i_h,j_h}\}$$

such that

$$S_{i_1,j_1} \cup \ldots \cup S_{i_h,j_h} = \mathcal{N} \setminus \mathcal{R}$$



$$\mathcal{S}_c = \{S_{i_1,j_1}, S_{i_2,j_2}, S_{i_4,j_3},$$

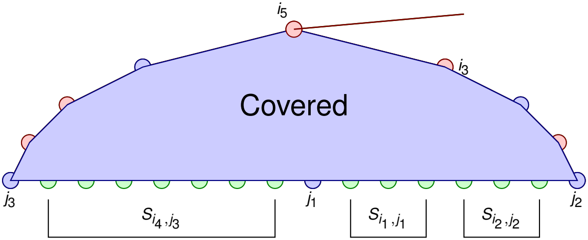
Subset Cover Finding Algorithm

Given $\mathcal{R}$, find

$$\mathcal{S}_c = \{S_{i_1,j_1}, \ldots, S_{i_h,j_h}\}$$

such that

$$S_{i_1,j_1} \cup \ldots \cup S_{i_h,j_h} = \mathcal{N} \setminus \mathcal{R}$$



$$\mathcal{S}_c = \{S_{i_1,j_1}, S_{i_2,j_2}, S_{i_4,j_3},$$

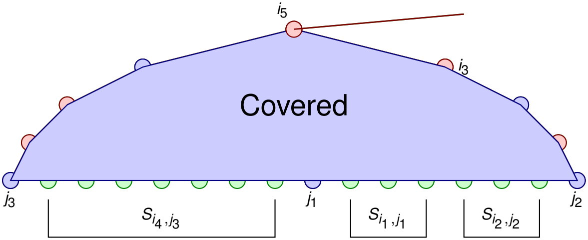
Subset Cover Finding Algorithm

Given $\mathcal{R}$, find

$$\mathcal{S}_c = \{S_{i_1,j_1}, \ldots, S_{i_h,j_h}\}$$

such that

$$S_{i_1,j_1} \cup \ldots \cup S_{i_h,j_h} = \mathcal{N} \setminus \mathcal{R}$$



$$\mathcal{S}_c = \{S_{i_1,j_1}, S_{i_2,j_2}, S_{i_4,j_3}, \ldots\}$$

NNL-SD Parameters

For n users out of which r are revoked:

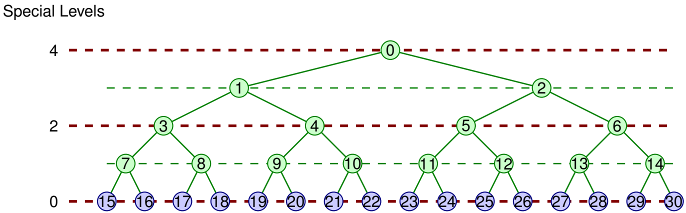
- User storage: $O(\log^2(n))$.
- Maximum header length: $2r - 1$.
- Maximum decryption time: $O(\log n)$.

Outline

ENS DE LYON

Layered Subset Difference Scheme [HS02]

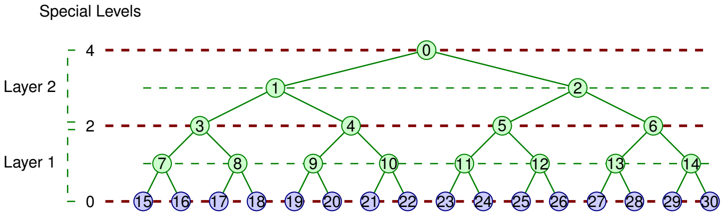
Halevy-Shamir (CRYPTO, 2002): *"special levels"*



Using layering (with special levels), $\mathcal{S}_{HS-LSD} \subset \mathcal{S}_{NNL-SD}$.

Layered Subset Difference Scheme [HS02]

Halevy-Shamir (CRYPTO, 2002): *"special levels"*



Using layering (with special levels), $\mathcal{S}_{HS-LSD} \subset \mathcal{S}_{NNL-SD}$.

Layered SD subsets

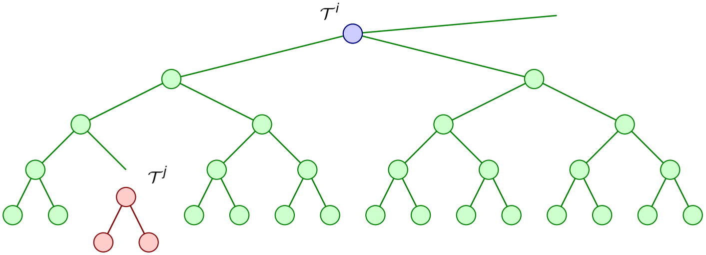
Which $S_{i,j} \in \mathcal{S}_{HS-LSD}$?

- If i is at a special level:
  for all j in $\mathcal{T}^i$, $S_{i,j} \in \mathcal{S}_{HS-LSD}$
- If i is not at a special level:
  for all j in $\mathcal{T}^i$ that are in the same layer as i, $S_{i,j} \in \mathcal{S}_{HS-LSD}$

Layered SD subsets

Which $S_{i,j} \in \mathcal{S}_{HS-LSD}$?

- If i is at a special level:
  for all j in $\mathcal{T}^i$, $S_{i,j} \in \mathcal{S}_{HS-LSD}$
- If i is not at a special level:
  for all j in $\mathcal{T}^i$ that are in the same layer as i, $S_{i,j} \in \mathcal{S}_{HS-LSD}$

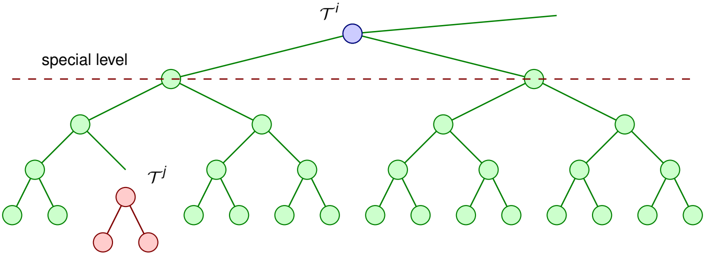


ENS DE LYON

Layered SD subsets

Which $S_{i,j} \in \mathcal{S}_{HS-LSD}$?

- If i is at a special level:
  for all j in $\mathcal{T}^i$, $S_{i,j} \in \mathcal{S}_{HS-LSD}$
- If i is not at a special level:
  for all j in $\mathcal{T}^i$ that are in the same layer as i, $S_{i,j} \in \mathcal{S}_{HS-LSD}$



Layered SD subsets

$S_{i,j} \in \mathcal{S}_{NNL-SD} \setminus \mathcal{S}_{HS-LSD}$ if

- i is not at a special level
- and i and j are not in the same layer

Layered SD subsets

$S_{i,j} \in \mathcal{S}_{NNL-SD} \setminus \mathcal{S}_{HS-LSD}$ if

- i is not at a special level
- and i and j are not in the same layer

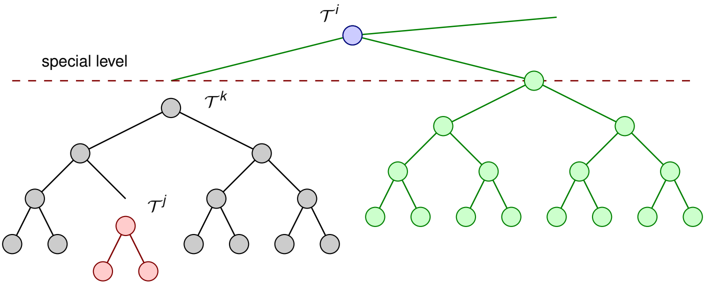
How to cover these subsets?

Layered SD subsets

$S_{i,j} \in \mathcal{S}_{NNL-SD} \setminus \mathcal{S}_{HS-LSD}$ if

- i is not at a special level
- and i and j are not in the same layer

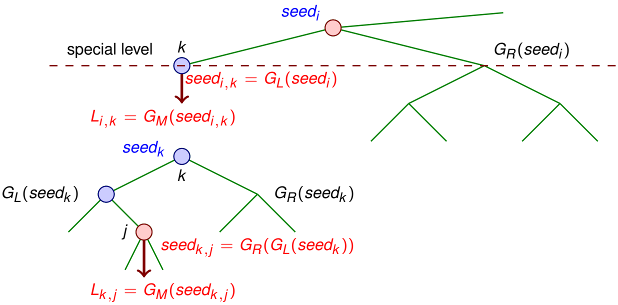
How to cover these subsets? SPLIT!!!



Subsets in $\mathcal{S}_{SD} \setminus \mathcal{S}_{LSD}$ are split into: $S_{i,j} = S_{i,k} \cup S_{k,j}$.

Layered SD Scheme

- Key for $S_{i,k}$ is $L_{i,k} = G_M(G_L(seed_i))$
- Key for $S_{k,j}$ is $L_{k,j} = G_M(G_R(G_L(seed_k)))$



LSD Parameters

NNL-SD scheme:

- User storage needed: $O(\log^2(n))$.
- Maximum Header Length: $2r - 1$.
- Decryption Time: $O(\log n)$.

HS-LSD scheme:

- User Storage needed: $O(\log^{3/2} n)$.
- Maximum header length: $4r - 2$.
- Decryption Time: $O(\log n)$.

STATISTICAL INDIAN INSTITUTE
UNITY IN DIVERSITY

ENS DE LYON

Outline

ENS DE LYON

Other SD-based Schemes

[GoodrichST04] Stratified SD

- Key assignment: Left and right preorder tree traversals
- $O(\log n)$ storage; $O(n)$ decryption time
- Double header length

[FukushimaKTS08] 3-ary tree SD

"However, in a general a-ary tree with $a \geq 4$,... our hash chain approach fails... Thus, the construction of a coalition resistant a-ary SD method with reasonable communication, computation, and storage overhead is an open issue."

[WangYL14] Balanced Double SD

- Published after I submitted my thesis
- We have better results now

Analysis of SD scheme

[ParkB06]

- Generating function for $N(n, r, h)$
- Mean header length: “complex to compute and difficult to gain insight from”

[EagleOPR08]

- Small standard deviations

[MartinMW09]

- Maximum header length

Outline

Preliminaries

Background

- NNL-SD: Initiation
  - Define $\mathcal{S}_{NNL-SD}$
  - Key Assignment
  - Key Distribution
- NNL-SD: Encryption
- Halevy-Shamir Layered SD
- Other Related Works

Our Contributions

- Paper 1: Arbitrary n; Detailed Analysis
- Paper 2: Layering; Minimizing Storage
- Paper 3: k-ary Generalization
- Paper 4: Assured Savings on Communication

Conclusion

ENS DE LYON

Complete Tree SD (CTSD) Scheme

Question: What happens when $n \neq 2^{\ell_0}$?
Answer: Add dummy users to get to the next power of two.

- If the dummy users are considered revoked, then the effect on the header length is disastrous.
- If the dummy users are privileged, the situation is better but, there is still a measurable effect on the header length.

Solution: Use a complete binary tree.

- "Completes" (and also subsumes) the NNL-SD scheme to work for any number of users.
- Conceptually simple; working out the details is a bit involved.

CTSD Scheme: Header Length Analysis

(n, r)-revocation

A choice of r revoked users out of total n users

For each (n, r)-revocation,

$$h \in \{1, \dots, h_{max}\}$$

$N(n, r, h)$

$\#(n, r)$-revocations for which the the header length is h.

CTSD Scheme: Header Length Analysis

(n, r)-revocation

A choice of r revoked users out of total n users

For each (n, r)-revocation,

$$h \in \{1, \ldots, h_{max}\}$$

$N(n, r, h)$

$\#(n, r)$-revocations for which the the header length is h.

How to compute $N(n, r, h)$?

The only known method would

- enumerate all possible $\binom{n}{r}$ (n, r)-revocations
- run the cover finding algorithm for each
- count the number of (n, r)-revocations leading to a header of size h.

Recurrence relation for $N(n, r, h)$

- $N(\lambda_i, r_1, h_1) = T(\lambda_i, r_1, h_1) + \sum_{j \in \mathrm{IN}(i)} T(\lambda_j, r_1, h_1 - 1)$ where $\mathrm{IN}(i)$ is the set of all internal nodes in the subtree $\mathcal{T}^i$ excluding the node i.
- $T(\lambda_i, r_1, h_1) = \sum_{r'=1}^{r_1-1} \sum_{h'=0}^{h_1} N(\lambda_{2i+1}, r', h') \times N(\lambda_{2i+2}, r_1 - r', h_1 - h')$ where λ_{2i+1} (respectively λ_{2i+2}) is the number of leaves in the left (respectively right) subtree of $\mathcal{T}^i$.

$T(\lambda_i, r_1, h_1)$	$r_1 < 0$	$r_1 = 0$	$r_1 = 1$	$2 \leq r_1 < n$	$r_1 = n$	$r_1 > n$
$h_1 = 0$	0	0	0	0	1	0
$h_1 \geq 1$	0	0	0	from rec.	0	0
$N(\lambda_i, r_1, h_1)$	$r_1 < 0$	$r_1 = 0$	$r_1 = 1$	$2 \leq r_1 < n$	$r_1 = n$	$r_1 > n$
$h_1 = 0$	0	0	0	0	1	0
$h_1 = 1$	0	1	n	from rec.	0	0
$h_1 > 1$	0	0	0	from rec.	0	0

Table: Boundary conditions on $T(n, r, h)$ and $N(n, r, h)$.

Computing $N(n, r, h)$

Dynamic Programming:

- $N(n, r, h)$ can be computed in $O(r^2h^2 \log n + rh \log^2 n)$ time and $O(rh \log n)$ space.
- $N(n, r, h)$ for all possible h can be computed in $O(r^4 \log n + r^2 \log n)$ time and $O(r^2 \log^2 n)$ space.
- $N(n, r, h)$ for all possible r and h can be computed in $O(n^4 \log n + n^2 \log^2 n)$ time and $O(n^2 \log n)$ space.
- $N(i, r, h)$ for $2 \leq i \leq n$ and all possible r and h can be computed in $O(n^5 + n^3 \log n)$ time and $O(n^3)$ space.

The combinatorics behind the cover generation algorithm was well captured!
(for n ~125)

ENS DE LYON

Using $N(n, r, h)$: Maximum Header Length

Theorem

The maximum header length in the CTSD method for n users is $h_{max} = \min(2r - 1, \lfloor \frac{n}{2} \rfloor, n - r)$.

- For the NNL-SD scheme, the bound of $2r - 1$ was known.
- Complete (refined) picture:
  - if $r \leq n/4$, $h_{max} = 2r - 1$;
  - if $n/4 < r \leq n/2$, $h_{max} = n/2$; and
  - for $r > n/2$, $h_{max} = n - r$.

ENS DE LYON

Using $N(n, r, h)$: More analysis

n_r

The value of n for which the header length of $2r - 1$ is achieved with r revoked users.

- Obtained a complete characterization of n_r.

Generating Function

- Similar to that of [PB06]

Probabilities and Expectation

- For n ~125
- Compute probabilities of $h \in \{1, \ldots, h_{max}\}$
- Compute expected value $H_{n,r}$

ENS DE LYON

Expected Header Length

Random experiment

Select a random subset of revoked users $\mathcal{R}$ from $\mathcal{N}$ (Select a random (n, r)-revocation).

Event: Node i generates a subset $S_{i,j}$

- $X^i_{n,r} = 1$ if $S_{i,j} \in \mathcal{S}_c$ for some j;
- $X^i_{n,r} = 0$ otherwise.

$$h = X^0_{n,r} + X^1_{n,r} + \cdots X^{n-1}_{n,r} = \sum_{i=0}^{n-1} X^i_{n,r}$$

$H_{n,r}$: expected header length for (n, r)-revocations.

$$H_{n,r} = \sum_{i=0}^{n-1} E[X^i_{n,r}] = \sum_{i=0}^{n-1} \Pr[X^i_{n,r} = 1]$$

$H_{n,r}$ for all SD based schemes

This technique has been useful for other SD-based schemes:

$$H_{n,r} = \sum_{i=0}^{n-1} \Pr[X^i_{n,r} = 1]$$

For the NNL-SD scheme:
Computing $H_{n,r}$ requires $O(r \log n)$ time and $O(1)$ space.

$H_{n,r}$ for the NNL-SD Scheme

Theorem: For all $n \geq 1$, $r \geq 1$, the expected header length $H_{n,r} \uparrow H_r$, as n increases through powers of two, where

$$H_r = 3r - 2 - 3 \times \sum_{i=1}^{r-1} \left(\left(-\frac{1}{2} \right)^i + \sum_{k=1}^{i} (-1)^k \binom{i}{k} \frac{(2^k - 3^k)}{(2^k - 1)} \right).$$

r	2	3	4	5	6
H_r/r	1.25	1.25	1.2455	1.2446	1.2448

ENS DE LYON

Outline

Preliminaries

Background

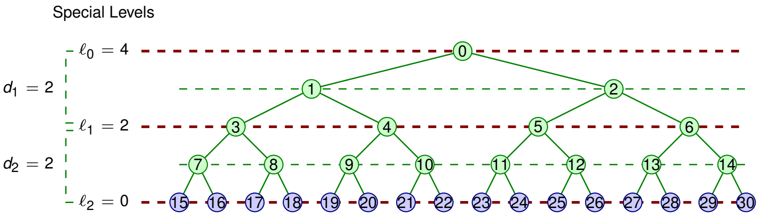
- NNL-SD: Initiation
  - Define $\mathcal{S}_{NNL-SD}$
  - Key Assignment
  - Key Distribution
- NNL-SD: Encryption
- Halevy-Shamir Layered SD
- Other Related Works

Our Contributions

- Paper 1: Arbitrary n; Detailed Analysis
- Paper 2: Layering; Minimizing Storage
- Paper 3: k-ary Generalization
- Paper 4: Assured Savings on Communication

Conclusion

Halevy-Shamir LSD Scheme



[HS02]: "The root is considered to be at a special level, and in addition we consider every level of depth $k \cdot \sqrt{\log(n)}$ *for* $k = 1 \ldots \log(n)$ *as special (wlog, we assume that these numbers are integers)."*

$n = 2^{\ell_0}$ with $\ell_0 = 4, 9, 16, 25$ only?

Layering Strategy

A choice of special levels is called a layering strategy.

General layering strategy $\boldsymbol{\ell}$

- Layering strategy $\boldsymbol{\ell} = (\ell_0, \ldots, \ell_e)$:
  - has $e + 1$ special levels
  - $\ell_0 > \ell_1 > \ldots > \ell_{e-1} > \ell_e = 0$.
- Layering strategy $\boldsymbol{d} = (d_1, \ldots, d_e)$
  - $d_i = \ell_i - \ell_{i-1}$ is a layer length
  - In general, the layer lengths need not be (almost) equal.

STATISTICAL INDIAN INSTITUTE

UNITY IN DIVERSITY

ENS DE LYON

Extending the HS Scheme

Residual bottom layer

Write $\ell_0 = d(e-1) + p$ where $1 \leq p \leq d$. Then the special levels are

$$\ell_0, \ell_0 - d, \ell_0 - 2d, \ldots, \ell - d(e-1), 0.$$

Balanced layering or extended-HS (eHS)

Write $\ell_0 = d(e-1) + p = (e-d+p)d + (d-p)(d-1)$. Define the layer lengths from the top to be

$$(\underbrace{d, \ldots, d}_{e-d+p}, \underbrace{d-1, \ldots, d-1}_{d-p}).$$

STATISTICAL INDIAN INSTITUTE

UNITY IN DIVERSITY

ENS DE LYON

Layering Strategy and User Storage

Layering strategy: $\boldsymbol{\ell} = (\ell_0, \ldots, \ell_e)$

$$\text{storage}_0(\boldsymbol{\ell}) \quad = \quad \sum_{i=0}^{e-1} \ell_i + \frac{1}{2} \sum_{i=0}^{e-1} (\ell_i - \ell_{i+1})(\ell_i - \ell_{i+1} - 1).$$

$$\text{storage}_0(\ell_0, \ell_1, \ldots, \ell_e)$$
$$= \quad \ell_0 + \frac{(\ell_0 - \ell_1)(\ell_0 - \ell_1 - 1)}{2} + \text{storage}_0(\ell_1, \ldots, \ell_e).$$

Storage Minimal Layering

$\mathrm{SML}_0(\ell_0)$

A layering strategy which minimizes the user storage among all layering strategies.

$\#\mathrm{SML}_0(\ell_0)$

User storage required by $\mathrm{SML}_0(\ell_0)$.

$$\begin{aligned}
\#\mathrm{SML}_0(\ell_0) &= \min_{1 \le e \le \ell_0} \#\mathrm{SML}_0(e, \ell_0); \\
\#\mathrm{SML}_0(e, \ell_0) &= \min_{(\ell_0, \ldots, \ell_e)} \mathrm{storage}_0(\ell_0, \ell_1, \ldots, \ell_e)
\end{aligned}$$

Dynamic programming algorithm to compute $\#\mathrm{SML}_0(\ell_0)$: $O(\ell_0^3)$ time and $O(\ell_0^2)$ space.

Root at a Non-Special Level

[HS02]: "The root is considered to be at a special level, and ..."

Making root level ℓ_0 non-special:

- $\text{storage}_1(\ell) = \text{storage}_0(\ell) - \ell_1$.
  Hence, user storage decreases.
- $\Pr[X^0_{n,r} = 1]$ is small.
  Hence, negligible increase in the expected header size.

$\text{SML}_1(\ell_0)$: SML with non-special root.
$\#\text{SML}_1(\ell_0)$: corresponding user storage.

Examples of SML

Suppose there are 2^{28} users, i.e., $\ell_0 = 28$
(a good estimate as per the CableLabs website)

Scheme Name	Layering ℓ	Storage $\|I_u\|$
NNL-SD:	(28,0)	406
eHS:	(28,22,16,10,5,0)	146
SML_0:	(28,21,15,10,6,3,1,0)	140
SML_1:	(22,16,11,7,4,2,0)	119

STATISTICAL INDIAN INSTITUTE
UNITY IN DIVERSITY

ENS DE LYON

Other Results

Complete Tree LSD scheme

Maximum Header Length

- $h_{max} = \min\left(4r - 2, \left\lceil \frac{n}{2} \right\rceil, n - r\right)$ if root is non-special.
- $h_{max} = \min\left(4r - 3, \left\lceil \frac{n}{2} \right\rceil, n - r\right)$ if root is special.

Expected Header Length:

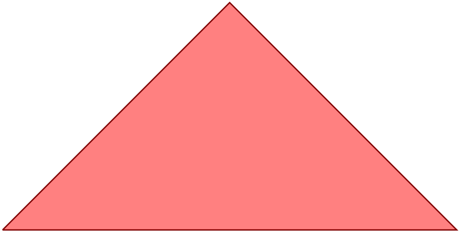
- The splitting of subsets complicates the analysis.
- $O(r \log^2 n)$ time and $O(1)$ space.

Constrained Minimization

- For a given r, the contribution of level $\ell_{max} = \ell_0 - \log_2 r$ to the header is maximum.
- As $r \uparrow$, $\ell_{max} \downarrow$. Hence,
  - Depending on the application, fix a value of r_{min} and set $\ell_{max} = \ell_0 - \log_2 r_{min}$.
  - Let $\boldsymbol{\ell} = \{\ell_{max}, 0\}$.

Constrained Minimization

- For a given r, the contribution of level $\ell_{max} = \ell_0 - \log_2 r$ to the header is maximum.
- As $r \uparrow$, $\ell_{max} \downarrow$. Hence,
  - Depending on the application, fix a value of r_{min} and set $\ell_{max} = \ell_0 - \log_2 r_{min}$.
  - Let $\boldsymbol{\ell} = \{\ell_{max}, 0\}$.

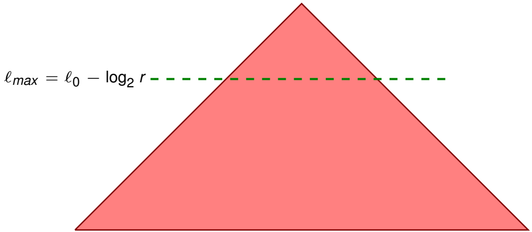


Result: $H_{n,r}$ close to that of NNL-SD, but, with lower user storage.

ENS DE LYON

Constrained Minimization

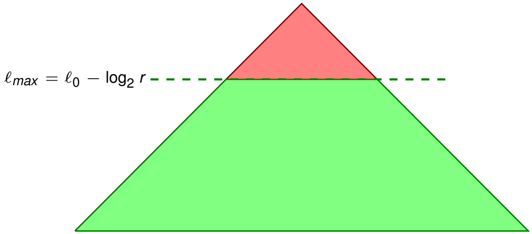
- For a given r, the contribution of level $\ell_{max} = \ell_0 - \log_2 r$ to the header is maximum.
- As $r \uparrow$, $\ell_{max} \downarrow$. Hence,
  - Depending on the application, fix a value of r_{min} and set $\ell_{max} = \ell_0 - \log_2 r_{min}$.
  - Let $\boldsymbol{\ell} = \{\ell_{max}, 0\}$.



Result: $H_{n,r}$ close to that of NNL-SD, but, with lower user storage.

Constrained Minimization

- For a given r, the contribution of level $\ell_{max} = \ell_0 - \log_2 r$ to the header is maximum.
- As $r \uparrow$, $\ell_{max} \downarrow$. Hence,
  - Depending on the application, fix a value of r_{min} and set $\ell_{max} = \ell_0 - \log_2 r_{min}$.
  - Let $\boldsymbol{\ell} = \{\ell_{max}, 0\}$.



Result: $H_{n,r}$ close to that of NNL-SD, but, with lower user storage.

A CML Example

$n = 2^{28}$ and $r_{\min} = 2^{10}$.

Scheme	Layering ℓ	$\lvert I_u \rvert$	$H_{n,r}$ (normalized with NNL-SD)
NNL-SD:	(28,0)	406	(1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00)
eHS:	(28,22,16,10,5,0)	146	(1.69, 1.63, 1.64, 1.67, 1.69, 1.72, 1.73, 1.74, 1.75, 1.75)
CML:	(23, 18,0)	219	(1.14, 1.08, 1.04, 1.03, 1.01, 1.01, 1.00, 1.00, 1.00, 1.00)

Header lengths for 10 equispaced values of r from 2^{10} to 2^{14} normalized by the header length of the NNL-SD scheme.

Outline

ENS DE LYON

k-ary tree SD

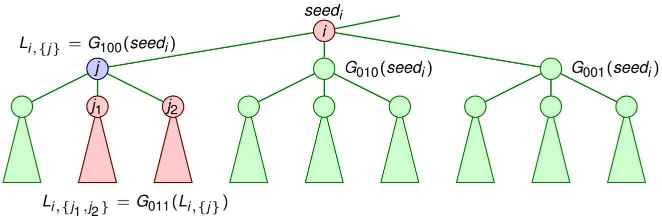


Figure: Key of $S_{i,\{j_1,j_2\}}$ is $G_{000}(L_{i,\{j_1,j_2\}}) = G_{000}(G_{011}(G_{100}(seed_i)))$.

User storage

$$1 + (2^{k-1} - 1) \sum_{\ell=1}^{\ell_0} \ell = 1 + \frac{\ell_0(\ell_0 + 1)}{2} (2^{k-1} - 1)$$

... reduced using additional tree structure
(constructed using cyclotomic cosets mod $2^k - 1$)

ENS DE LYON

k-ary tree SD: Results

- Why k-ary trees?
  - $|\mathcal{S}| \uparrow \implies (H_{n,r} \downarrow, \ |I_u| \uparrow)$

k-ary tree SD: Results

- Why k-ary trees?
  - $|\mathcal{S}| \uparrow \implies (H_{n,r} \downarrow, \ |I_u| \uparrow)$ always?

k-ary tree SD: Results

- Why k-ary trees?
  - $|\mathcal{S}| \uparrow \implies (H_{n,r} \downarrow, \ |I_u| \uparrow)$ always?
  - Hierarchy of Optimization

k-ary tree SD: Results

- Why k-ary trees?
  - $|\mathcal{S}| \uparrow \implies (H_{n,r} \downarrow, |I_u| \uparrow)$ always?
  - Hierarchy of Optimization
- Header length analysis
  - $h_{max} = \min(2r - 1, \lceil n/k \rceil, n - r)$
  - Algorithm to compute $H_{n,r}$ (for $n = k^{\ell_0}$)
    $O(r \log n)$ space; $O(1)$ time
- Reducing user storage
  - Using cyclotomic cosets modulo $2^k - 1$
  - An additional tree structure $T^{(k)}$

k-ary tree SD: Results

- Why k-ary trees?
  - $|\mathcal{S}| \uparrow \implies (H_{n,r} \downarrow, |I_u| \uparrow)$ always?
  - Hierarchy of Optimization
- Header length analysis
  - $h_{max} = \min(2r-1, \lceil n/k \rceil, n-r)$
  - Algorithm to compute $H_{n,r}$ (for $n = k^{\ell_0}$)
    $O(r \log n)$ space; $O(1)$ time
- Reducing user storage
  - Using cyclotomic cosets modulo $2^k - 1$
  - An additional tree structure $T^{(k)}$
- Complete Tree for arbitrary number of users
- Layering
  - Storage Minimal Layering
- Header length simulation study (for $n \neq k^{\ell_0}$)

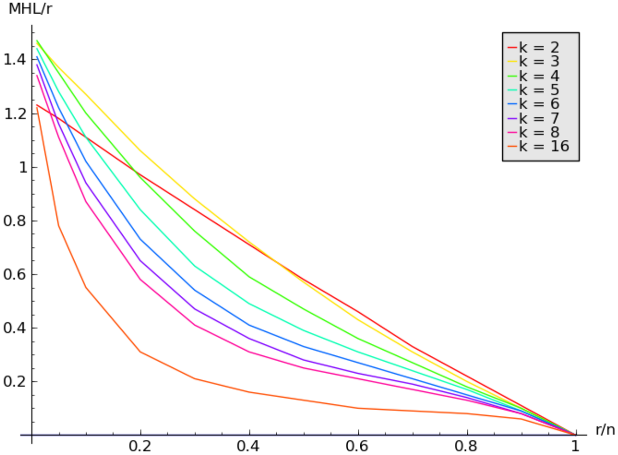
ENS DE LYON

k-ary tree SD: Header length and user storage

n	k	us_k	MHL_k/r	n	k	us_k	MHL_k/r
10^3	2	55	(1.10, 0.98, 0.72)	10^4	2	105	(1.11, 0.97, 0.71)
	3	56	(1.27, 1.06, 0.72)		3	90	(1.26, 1.07, 0.72)
	4	60	(1.21, 0.96, 0.59)		4	112	(1.20, 0.96, 0.59)
	5	90	(1.11, 0.84, 0.50)		5	126	(1.11, 0.84, 0.49)
	6	120	(1.03, 0.73, 0.42)		6	252	(1.02, 0.73, 0.41)
	7	180	(0.95, 0.65, 0.36)		7	270	(0.94, 0.65, 0.36)
	8	340	(0.86, 0.58, 0.32)		8	510	(0.86, 0.58, 0.31)
10^5	2	153	(1.11, 0.97, 0.71)	10^6	2	210	(1.11, 0.97, 0.71)
	3	132	(1.27, 1.06, 0.72)		3	182	(1.27, 1.07, 0.72)
	4	180	(1.20, 0.96, 0.59)		4	220	(1.20, 0.96, 0.59)
	5	216	(1.11, 0.84, 0.49)		5	270	(1.11, 0.84, 0.49)
	6	336	(1.02, 0.73, 0.41)		6	432	(1.02, 0.73, 0.41)
	7	378	(0.94, 0.65, 0.36)		7	648	(0.94, 0.65, 0.36)
	8	714	(0.87, 0.58, 0.31)		8	952	(0.87, 0.58, 0.31)
10^7	2	300	(1.11, 0.97, 0.71)	10^8	2	378	(1.11, 0.97, 0.71)
	3	240	(1.27, 1.06, 0.72)		3	306	(1.27, 1.06, 0.72)
	4	312	(1.20, 0.96, 0.59)		4	420	(1.20, 0.96, 0.59)
	5	396	(1.11, 0.84, 0.49)		5	468	(1.11, 0.84, 0.49)
	6	540	(1.02, 0.73, 0.41)		6	792	(1.02, 0.73, 0.41)
	7	810	(0.94, 0.65, 0.36)		7	990	(0.94, 0.65, 0.36)
	8	1224	(0.87, 0.58, 0.31)		8	1530	(0.87, 0.58, 0.31)

Table: MHL_k/r for $r = (0.1n, 0.2n, 0.4n)$.

k-ary tree SD



k	3	4	5	6	7	8	16
δ_k	0.44	0.19	0.11	0.07	0.05	0.04	< 0.01

Table: Values of the threshold δ_k.

Outline

ENS DE LYON

a-ABTSD scheme

- $\mathcal{S}_{NNL-SD} \subset \mathcal{S}_{a-ABTSD}$
  - Augment trees of height a (with $k = 2^a$ leaf nodes)

a-ABTSD scheme

- $\mathcal{S}_{NNL-SD} \subset \mathcal{S}_{a-ABTSD}$
  - Augment trees of height a (with $k = 2^a$ leaf nodes)
  - (Better?) Hierarchy of Optimization

a-ABTSD scheme

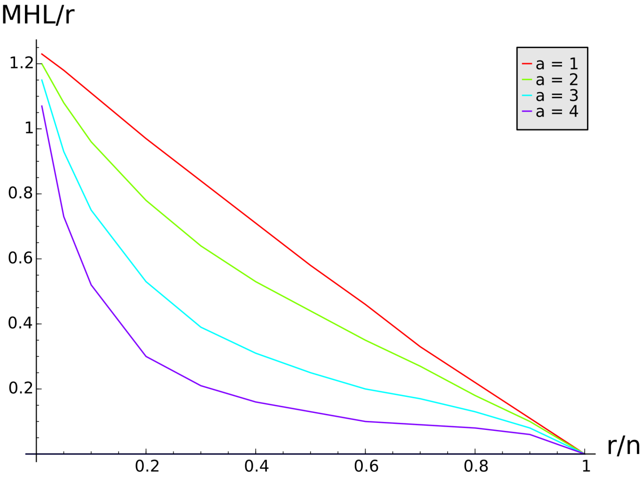
- $\mathcal{S}_{NNL-SD} \subset \mathcal{S}_{a-ABTSD}$
  - Augment trees of height a (with $k = 2^a$ leaf nodes)
  - (Better?) Hierarchy of Optimization
- Header length analysis
  - $h_{max} = \min(2r - 1, \lceil n/k \rceil, n - r)$
- Reducing user storage
  - Using cyclotomic cosets modulo $2^k - 1$
  - An additional tree structure $T^{(k)}$
- Complete Tree for arbitrary number of users
- Header length simulation study

a-ABTSD: Header length and user storage

n	a	$us_a(n)$	MHL_a/r	n	a	$us_a(n)$	MHL_a/r
10^3	1	55	$(1.11, 0.97, 0.71)$	10^4	1	105	$(1.11, 0.97, 0.71)$
	2	145	$(0.96, 0.78, 0.53)$		2	287	$(0.96, 0.78, 0.53)$
	3	1279	$(0.75, 0.53, 0.31)$		3	2757	$(0.75, 0.53, 0.31)$
	4	115247	$(0.52, 0.31, 0.16)$		4	271629	$(0.52, 0.30, 0.16)$
10^5	1	153	$(1.11, 0.97, 0.71)$	10^6	1	210	$(1.11, 0.97, 0.71)$
	2	425	$(0.96, 0.78, 0.53)$		2	590	$(0.96, 0.78, 0.53)$
	3	4233	$(0.75, 0.53, 0.31)$		3	6024	$(0.75, 0.53, 0.31)$
	4	432123	$(0.52, 0.30, 0.16)$		4	629652	$(0.52, 0.30, 0.16)$
10^7	1	300	$(1.11, 0.97, 0.71)$	10^8	1	378	$(1.11, 0.97, 0.71)$
	2	852	$(0.96, 0.78, 0.53)$		2	1080	$(0.96, 0.78, 0.53)$
	3	8902	$(0.75, 0.53, 0.31)$		3	11428	$(0.75, 0.53, 0.31)$
	4	950634	$(0.52, 0.30, 0.16)$		4	1234578	$(0.52, 0.30, 0.16)$

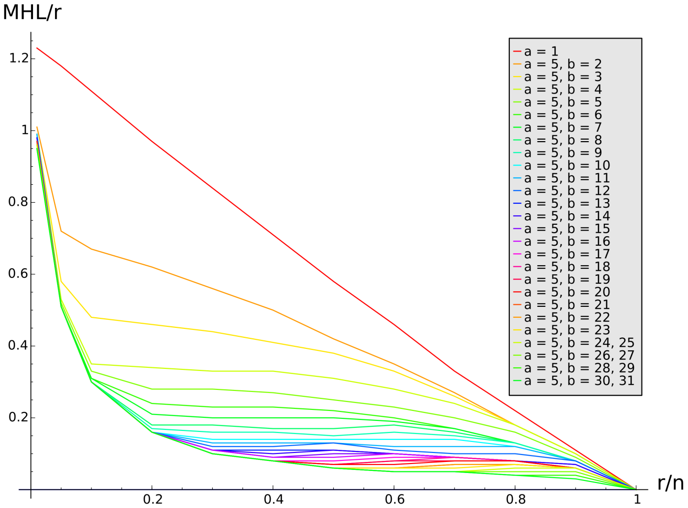
Table: MHL_a/r for three different choices of r namely, $r = (0.1n, 0.2n, 0.4n)$.

a-ABTSD performance



with $b = 2^a - 1$ and $c = \ell_0$.

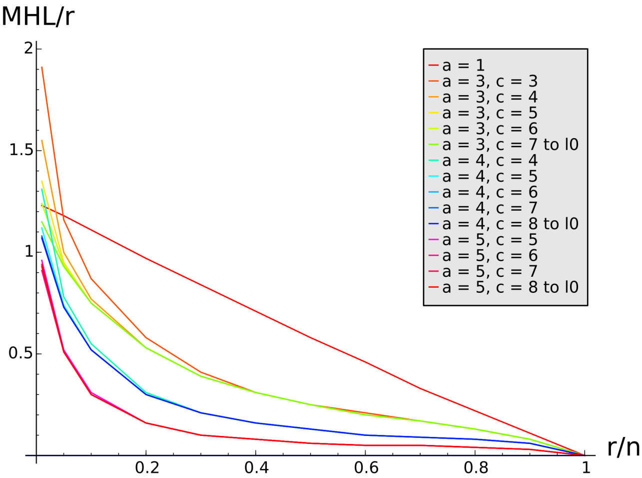
(a, b, c)-ABTSD



with $a = 5$ and $c = \ell_0$.

ENS DE LYON

(a, b, c)-ABTSD



with $b = 2^a - 1$.

Preliminaries

Background

Our Contributions

Conclusion

Implementations

Schemes:

- NNL-SD, HS-LSD and all new schemes

Analysis:

- Header length algorithms
- User storage algorithms
- ...

What this thesis is NOT about

Asymptotic Improvements

What this thesis is ALL about

Combinatorial and Probabilistic Analysis

What this thesis is ALL about

Combinatorial and Probabilistic Analysis

Obtaining Hierarchies of Optimization

Summary of Contributions

- What if $n \neq 2^{\ell_0}$?

1, 2, 3, 4 Use dummy users or complete trees?

Summary of Contributions

- What if $n \neq 2^{\ell_0}$?

1, 2, 3, 4 Use dummy users or complete trees?

- Analysis of SD-based schemes?

Summary of Contributions

- What if $n \neq 2^{\ell_0}$?

1, 2, 3, 4 Use dummy users or complete trees?

- Analysis of SD-based schemes?
  1. $N(n, r, h)$

STATISTICAL INDIAN INSTITUTE
UNITY IN DIVERSITY

ENS DE LYON

Summary of Contributions

- What if $n \neq 2^{\ell_0}$?

1, 2, 3, 4 Use dummy users or complete trees?

- Analysis of SD-based schemes?
  1 $N(n, r, h)$
  1 Generating function [PB06]

ENS DE LYON

Summary of Contributions

- What if $n \neq 2^{\ell_0}$?

1, 2, 3, 4 Use dummy users or complete trees?

- Analysis of SD-based schemes?

 1 $N(n, r, h)$

 1 Generating function [PB06]

1, 2, 3, 4 Maximum and Mean Header Lengths ($H_{n,r}$)?

  - In [PB06]: too complicated!!! (approximations)

STATISTICAL INDIAN INSTITUTE UNITY IN DIVERSITY

ENS DE LYON

Summary of Contributions

- What if $n \neq 2^{\ell_0}$?

1, 2, 3, 4 Use dummy users or complete trees?

- Analysis of SD-based schemes?

1 $N(n, r, h)$

1 Generating function [PB06]

1, 2, 3, 4 Maximum and Mean Header Lengths ($H_{n,r}$)?

  - In [PB06]: too complicated!!! (approximations)

1 Upper bound on $H_{n,r}$?

Summary of Contributions

- What if $n \neq 2^{\ell_0}$?

1, 2, 3, 4 Use dummy users or complete trees?

- Analysis of SD-based schemes?

1 $N(n, r, h)$

1 Generating function [PB06]

1, 2, 3, 4 Maximum and Mean Header Lengths ($H_{n,r}$)?

  - In [PB06]: too complicated!!! (approximations)

1 Upper bound on $H_{n,r}$?

  - $1.38r$ (sketchy proof [NNL01])
  - $1.25r$ (empirical [NNL01]) - theoretical analysis?

ENS DE LYON

Summary of Contributions

- What if $n \neq 2^{\ell_0}$?

1, 2, 3, 4 Use dummy users or complete trees?

- Analysis of SD-based schemes?

  1 $N(n, r, h)$

  1 Generating function [PB06]

1, 2, 3, 4 Maximum and Mean Header Lengths ($H_{n,r}$)?

  - In [PB06]: too complicated!!! (approximations)

  1 Upper bound on $H_{n,r}$?

  - $1.38r$ (sketchy proof [NNL01])
  - $1.25r$ (empirical [NNL01]) - theoretical analysis?

- Choice of $\mathcal{S}$: $|\mathcal{S}| \uparrow$ or $|\mathcal{S}| \downarrow$?

Summary of Contributions

- What if $n \neq 2^{\ell_0}$?

1, 2, 3, 4 Use dummy users or complete trees?

- Analysis of SD-based schemes?

  1 $N(n, r, h)$

  1 Generating function [PB06]

1, 2, 3, 4 Maximum and Mean Header Lengths ($H_{n,r}$)?

  - In [PB06]: too complicated!!! (approximations)

  1 Upper bound on $H_{n,r}$?

  - $1.38r$ (sketchy proof [NNL01])
  - $1.25r$ (empirical [NNL01]) - theoretical analysis?

- Choice of $\mathcal{S}$: $|\mathcal{S}| \uparrow$ or $|\mathcal{S}| \downarrow$?

  2 Storage minimal layering

STATISTICAL INDIAN INSTITUTE
UNITY IN DIVERSITY

ENS DE LYON

Summary of Contributions

- What if $n \neq 2^{\ell_0}$?

1, 2, 3, 4 Use dummy users or complete trees?

- Analysis of SD-based schemes?

1 $N(n, r, h)$
1 Generating function [PB06]
1, 2, 3, 4 Maximum and Mean Header Lengths ($H_{n,r}$)?
  - In [PB06]: too complicated!!! (approximations)

1 Upper bound on $H_{n,r}$?
  - $1.38r$ (sketchy proof [NNL01])
  - $1.25r$ (empirical [NNL01]) - theoretical analysis?

- Choice of $\mathcal{S}$: $|\mathcal{S}| \uparrow$ or $|\mathcal{S}| \downarrow$?

2 Storage minimal layering
2 Constrained minimization layering

ENS DE LYON

Summary of Contributions

- What if $n \neq 2^{\ell_0}$?

1, 2, 3, 4 Use dummy users or complete trees?

- Analysis of SD-based schemes?

1 $N(n, r, h)$

1 Generating function [PB06]

1, 2, 3, 4 Maximum and Mean Header Lengths ($H_{n,r}$)?

  - In [PB06]: too complicated!!! (approximations)

1 Upper bound on $H_{n,r}$?

  - $1.38r$ (sketchy proof [NNL01])
  - $1.25r$ (empirical [NNL01]) - theoretical analysis?

- Choice of $\mathcal{S}$: $|\mathcal{S}| \uparrow$ or $|\mathcal{S}| \downarrow$?

2 Storage minimal layering

2 Constrained minimization layering

3 k-ary tree SD scheme

STATISTICAL INDIAN INSTITUTE

UNITY IN DIVERSITY

ENS DE LYON

Summary of Contributions

- What if $n \neq 2^{\ell_0}$?

1, 2, 3, 4 Use dummy users or complete trees?

- Analysis of SD-based schemes?

  1 $N(n, r, h)$

  1 Generating function [PB06]

1, 2, 3, 4 Maximum and Mean Header Lengths ($H_{n,r}$)?

  - In [PB06]: too complicated!!! (approximations)

  1 Upper bound on $H_{n,r}$?

  - $1.38r$ (sketchy proof [NNL01])
  - $1.25r$ (empirical [NNL01]) - theoretical analysis?

- Choice of $\mathcal{S}$: $|\mathcal{S}| \uparrow$ or $|\mathcal{S}| \downarrow$?

  2 Storage minimal layering

  2 Constrained minimization layering

  3 k-ary tree SD scheme

  4 (a, b, c)-ABTSD scheme

STATISTICAL INDIAN INSTITUTE
UNITY IN DIVERSITY

ENS DE LYON

$|\mathcal{S}|$

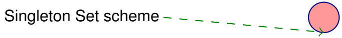
Intuition:

Choice of $\mathcal{S}$: $|\mathcal{S}| \uparrow$ or $|\mathcal{S}| \downarrow$

$|\mathcal{S}|$

Intuition:

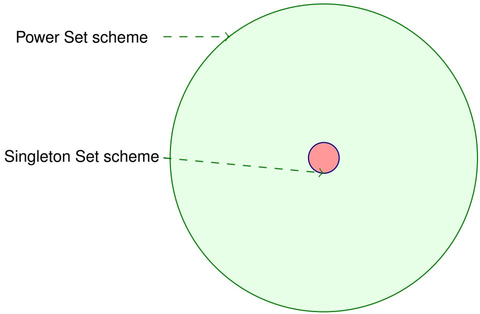
Choice of $\mathcal{S}$: $|\mathcal{S}| \uparrow$ or $|\mathcal{S}| \downarrow$



ENS DE LYON

Intuition:

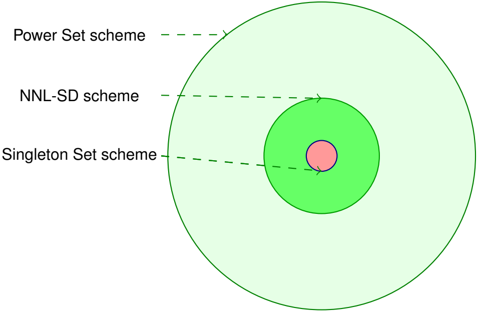
Choice of $\mathcal{S}$: $|\mathcal{S}| \uparrow$ or $|\mathcal{S}| \downarrow$



$|\mathcal{S}|$

Intuition:

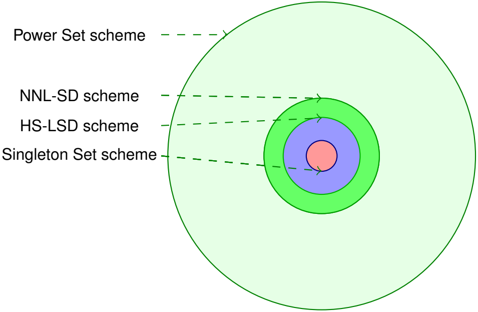
Choice of $\mathcal{S}$: $|\mathcal{S}| \uparrow$ or $|\mathcal{S}| \downarrow$



$|\mathcal{S}|$

Intuition:

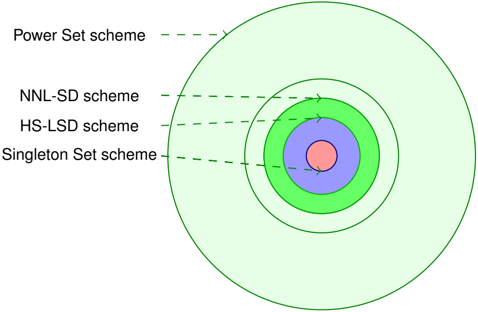
Choice of $\mathcal{S}$: $|\mathcal{S}| \uparrow$ or $|\mathcal{S}| \downarrow$



$|\mathcal{S}|$

Intuition:

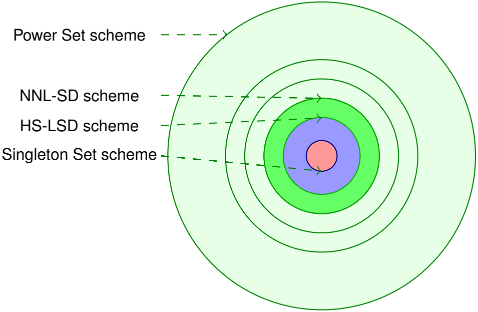
Choice of $\mathcal{S}$: $|\mathcal{S}| \uparrow$ or $|\mathcal{S}| \downarrow$



$|\mathcal{S}|$

Intuition:

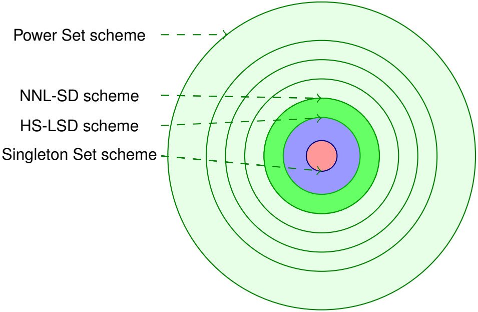
Choice of $\mathcal{S}$: $|\mathcal{S}| \uparrow$ or $|\mathcal{S}| \downarrow$



$|\mathcal{S}|$

Intuition:

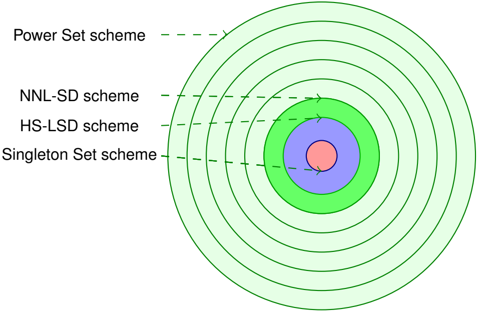
Choice of $\mathcal{S}$: $|\mathcal{S}| \uparrow$ or $|\mathcal{S}| \downarrow$



$|\mathcal{S}|$

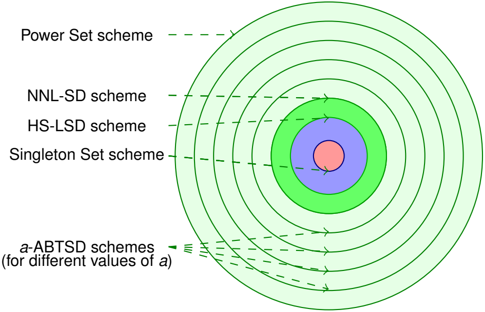
Intuition:

Choice of $\mathcal{S}$: $|\mathcal{S}| \uparrow$ or $|\mathcal{S}| \downarrow$



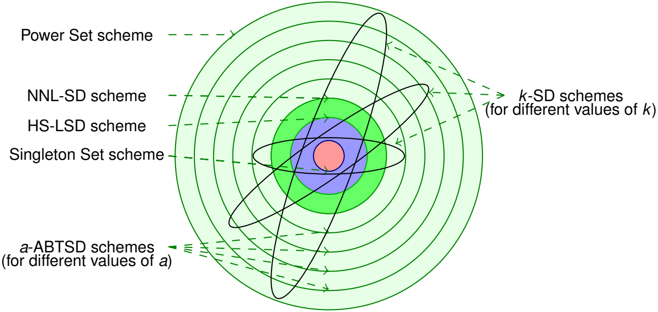
Intuition:

Choice of $\mathcal{S}$: $|\mathcal{S}| \uparrow$ or $|\mathcal{S}| \downarrow$



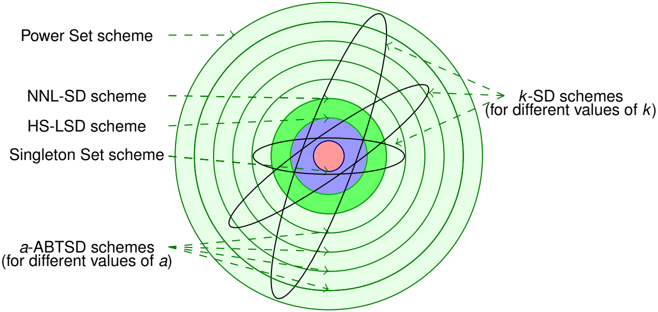
Intuition:

Choice of $\mathcal{S}$: $|\mathcal{S}| \uparrow$ or $|\mathcal{S}| \downarrow$



Intuition:

Choice of $\mathcal{S}$: $|\mathcal{S}| \uparrow$ or $|\mathcal{S}| \downarrow$



Publications

Sanjay Bhattacherjee and Palash Sarkar.
Complete tree subset difference broadcast encryption scheme and its analysis.
Des. Codes Cryptography, 66(1-3):335–362, 2013.

Sanjay Bhattacherjee and Palash Sarkar.
Concrete analysis and trade-offs for the (complete tree) layered subset difference broadcast encryption scheme.
IEEE Transactions on Computers, 63(7): 1709–1722, 2014.

Sanjay Bhattacherjee and Palash Sarkar.
Tree based symmetric key broadcast encryption.
J. Discrete Algorithms, 34: 78–107, 2015.

Sanjay Bhattacherjee and Palash Sarkar.
Reducing communication overhead of the subset difference scheme.
IEEE Transactions on Computers, to appear.

Sanjay Bhattacherjee and Palash Sarkar.
Implementations related to the above papers, `https://drive.google.com/folderview?id=0B7azs7qqqdS0UnB5aHp3WmJwcDQ&usp=sharing_eil`.
Uploaded on 13th August, 2014.

Open Questions

Schemes:

Open Questions

Schemes:

- More hierarchies of optimization?

Open Questions

Schemes:

- More hierarchies of optimization?
- Practical scheme with $h_{max} < r$

Open Questions

Schemes:

- More hierarchies of optimization?
- Practical scheme with $h_{max} < r$
- Stateless as well as forward secure?
- ...

Analysis:

Open Questions

Schemes:

- More hierarchies of optimization?
- Practical scheme with $h_{max} < r$
- Stateless as well as forward secure?
- ...

Analysis:

- Non-uniform distribution of revoked users?
- ...

ENS DE LYON

Acknowledgement

- Prof. Palash Sarkar
- Friends
- Family


A MONSTER
BIG
THANK YOU
x

Desirable Properties

Fully Collusion Resistant

Desirable Properties

Fully Collusion Resistant

Dynamic revocation

Desirable Properties

Fully Collusion Resistant

Dynamic revocation

Stateless / Stateful

STATISTICAL INDIAN INSTITUTE UNITY IN DIVERSITY

ENS DE LYON

Desirable Properties

Fully Collusion Resistant

Dynamic revocation

Stateless / Stateful

Traitor Tracing

Desirable Properties

Fully Collusion Resistant

Dynamic revocation

Dynamic joining / leaving of users

Stateless / Stateful

Traitor Tracing

Assigning seeds to users

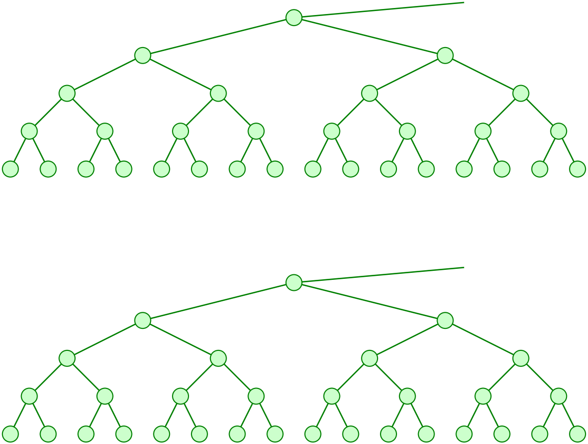


Figure: From one derived seed, keys of many subsets can be generated

Assigning seeds to users

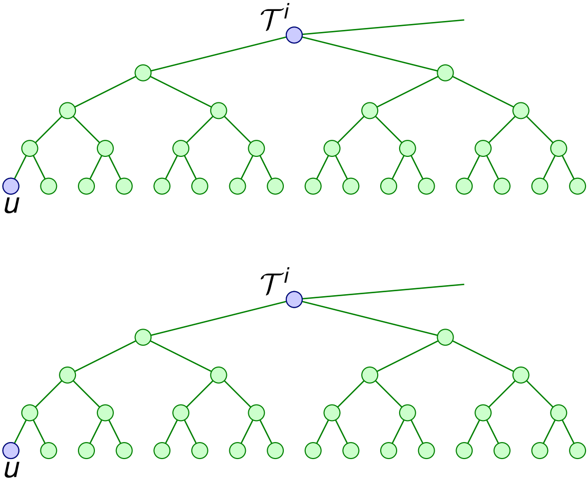


Figure: From one derived seed, keys of many subsets can be generated

Assigning seeds to users

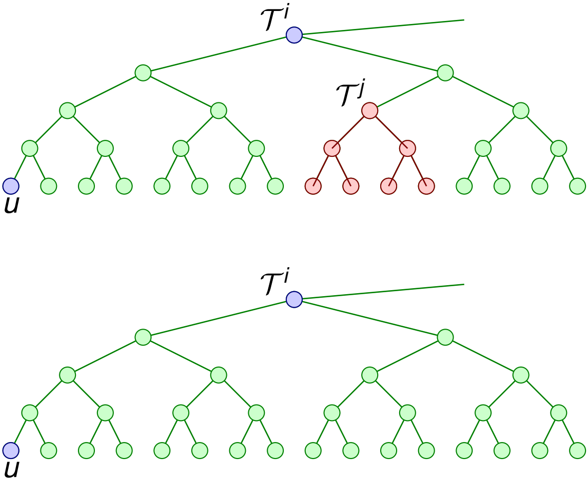


Figure: From one derived seed, keys of many subsets can be generated

Assigning seeds to users

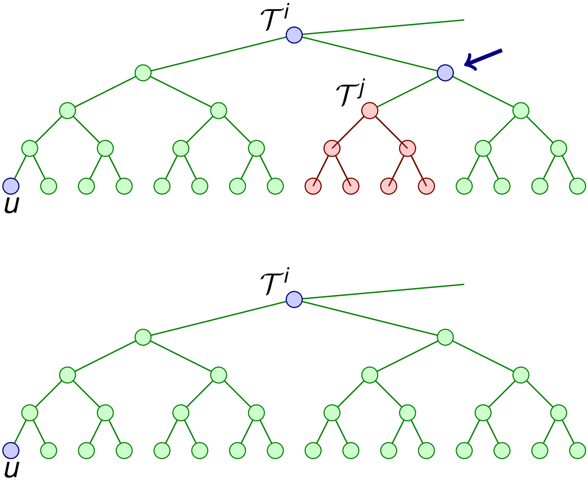
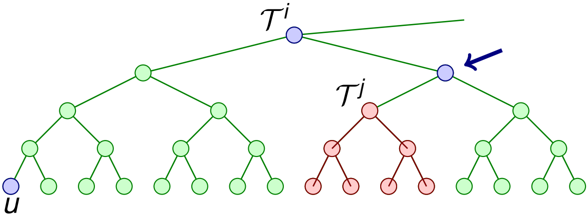


Figure: From one derived seed, keys of many subsets can be generated

Assigning seeds to users



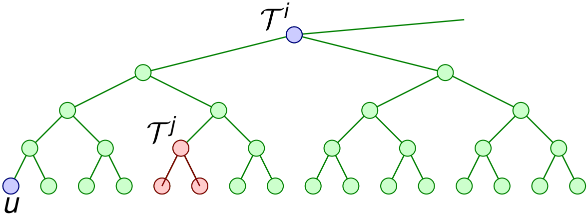


Figure: From one derived seed, keys of many subsets can be generated

Assigning seeds to users

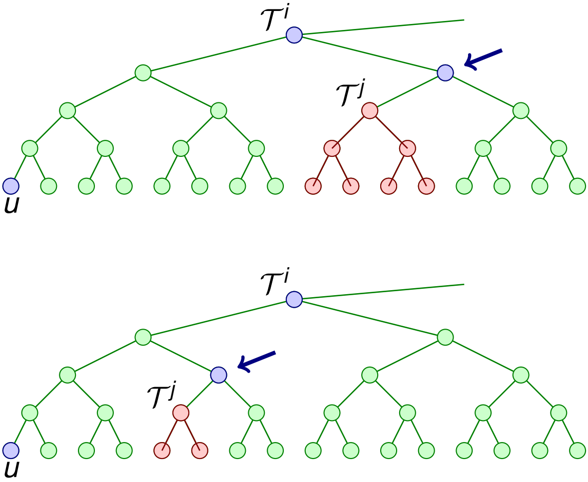
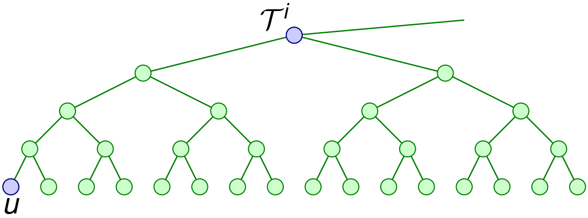


Figure: From one derived seed, keys of many subsets can be generated

Assigning seeds to users



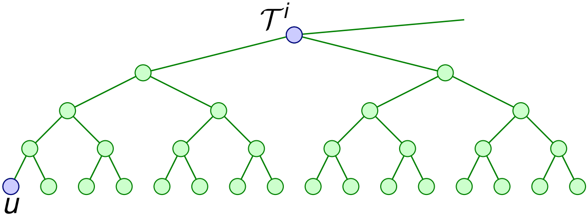


Figure: From one derived seed, keys of many subsets can be generated

Assigning seeds to users

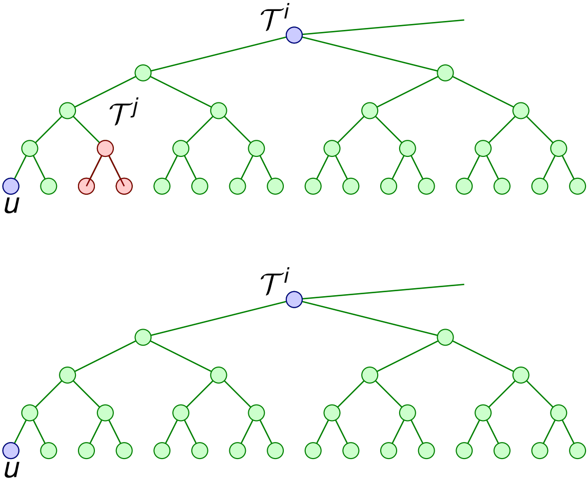


Figure: From one derived seed, keys of many subsets can be generated

Assigning seeds to users

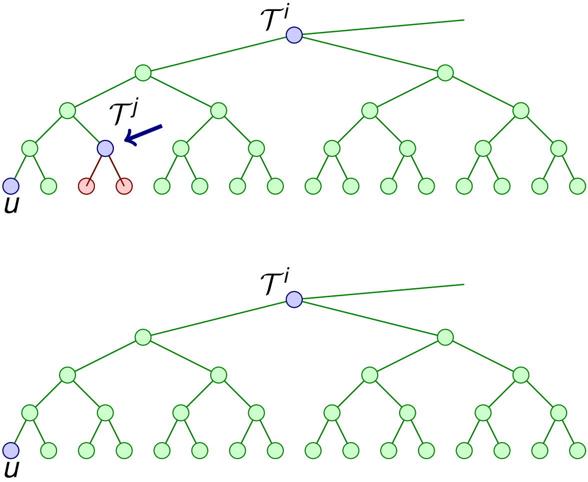


Figure: From one derived seed, keys of many subsets can be generated

Assigning seeds to users

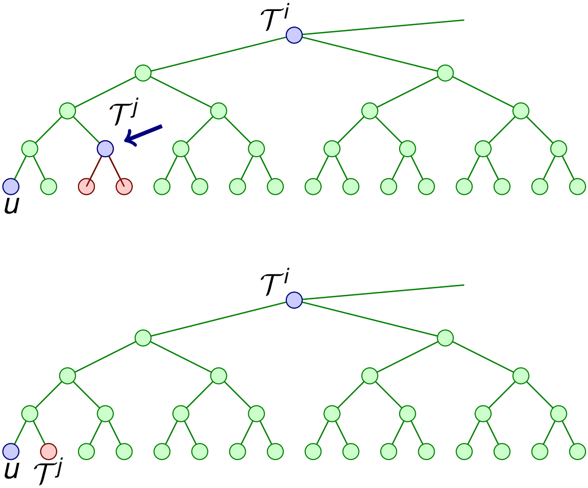


Figure: From one derived seed, keys of many subsets can be generated

Assigning seeds to users

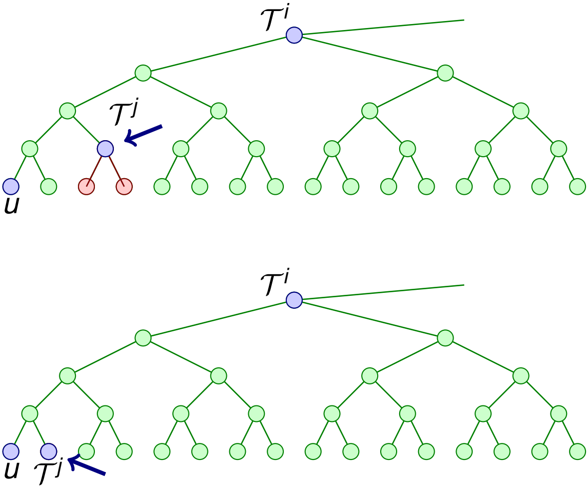


Figure: From one derived seed, keys of many subsets can be generated

ENS DE LYON